Universidad Autónoma del Estado de México
Centro Universitario UAEM Zumpango

Ingeniería en computación

DESARROLLO DE APLICACIONES DE ESCRITORIO CON PYTHON Y LA BIBLIOTECA QT

TESINA

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

Luis Enrique Rojas Desales

DIRECTOR:
Dr. Asdrúbal López Chau

Enero, 2022

Resumen

Python es un lenguaje de programación que comenzó a desarrollar en 1990, por sus características para el desarrollo de programas se a convertido en unos de los lenguajes de programación dominantes, ya que su curva de aprendizaje es muy sencilla. Python también cuenta con la programación orientada a objetos (POO), trabajando en conjunto con otras herramientas como Qt, permitiendo a los desarrolladores crear interfaces sofisticadas y de uso formal para el usuario final.

Qt es una herramienta par la creación de interfaces, ya que cuenta con la tecnología de innovación y es de código abierto, lo que permite ser manipulado por programadores para mejorar y optimizar la herramienta. qt esta dotada de una serie de librerías desarrolladas por Trolltech, creador de Qt en el año de 1991. Qt cuenta con una serie de herramientas que permiten completar las interfaces, como son QtCreator, QtDesigner, entre otras.

PyQt en la unión de los enlaces del lenguaje Python y las herramientas de Qt, permitiendo que esta herramienta se empleada en diversas plataformas como Windows, Mac OS/X, Linux, UNIX entre otros como los embebidos, como SjarpZaurus y iPAQ de la compañía de Compaq. PyQt esta con licencias GPL, licencias que permiten el uso de la versión comercial de Qt. Además PyQt esta constituido por módulos de Python, extensiones del lenguaje de C++.

Abstract

Python is a programming language that began to develop in 1990, due to its characteristics for the development of programs it has become one of the dominant programming languages, since its learning curve is very simple. Python also has object-oriented programming (OOP), working in conjunction with other tools such as Qt, allowing developers to create sophisticated and formally-used interfaces for the end user.

Qt is a tool for the creation of interfaces, since it has innovation technology and is open source, which allows it to be manipulated by programmers to improve and optimize the tool. qt is endowed with a series of libraries developed by Trolltech, creator of Qt in the year of 1991. Qt has a series of tools that allow completing the interfaces, such as QtCreator, QtDesigner, among others.

PyQt in the union of Python language links and Qt tools, allowing this tool to be used in various platforms such as Windows, Mac OS / X, Linux, UNIX, among others, such as embedded ones, such as SjarpZaurus and iPAQ of the company Compaq. PyQt is with GPL licenses, licenses that allow the use of the commercial version of Qt. In addition, PyQt is made up of Python modules, C ++ language extensions.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Anteproyecto

1.1. Introducción

El desarrollo de aplicaciones es cada vez más solicitado por empresas y se requiere que éste presente una buena calidad, ya que forman una parte fundamental del funcionamiento cotidiano de ellas. En varias universidades de nuestro país, se enseñan los lenguajes de programación C/C++ o Java como los principales lenguajes para desarrollo de aplicaciones; sin embargo, presentan la desventaja que una difícil migración hacia varias plataformas, tales como las móviles o Web. Otras opciones, como Python, tienen permiten el desarrollo más simple, y la capacidad para crear interfaces gráficas de usuario con entornos de trabajo como Qt. Python cuenta con licencia libre, código abierto y es multiplataforma. Qt se ha utilizado en proyectos tan exitosos como Skype, Google Earth. Cabe resaltar que Qt no es un lenguaje de programación nuevo, resulta bastante útil desarrollar en Qt si se tiene conocimiento de C/C++ o Java.

1.2. Planteamiento del problema

En la actualidad el desarrollo y la creación de nuevas aplicaciones se está volviendo una necesidad para las empresas, así como también una fuente de empleo para muchos de los programadores y desarrolladores. Esto porque la mayoría de las operaciones se realizan a través de computadoras o dispositivos móviles. Anteriormente los lenguajes de programación dominantes para el desarrollo de sistemas, desarrollo móvil, y Web eran C/C++ y Java. El lenguaje C++ ha sido elegido por su estructura y sintaxis, además implementa la programación orientada a objetos. Java por su parte tiene características de C++, las cuales son atractivas para el desarrollo. Además, implementar el paradigma de programación orientada a objetos. Sin embargo, la curva de aprendizaje de estos lenguajes es larga. Lenguaje C++ no está pensado para desarrollo de aplicaciones Web o para móviles. Java sí se usa en desarrollos Web y móvil. Python es un lenguaje relativamente nuevo, que tiene características muy atractivas para el desarrollo. Además, cuenta con entornos de trabajo y bibliotecas poderosas. Sin embargo, en muchas universidades de México todavía no se fomenta su uso, pese a que en el mercado laboral sí existe una demanda de desarrolladores Python [13]. En este trabajo se

explica la forma de desarrollar aplicaciones con Python, presentando algunos fundamentos y el uso de la biblioteca Qt para crear interfaces gráficas de usuario.

1.3. Justificación

La presente investigación tiene un enfoque de conocimiento y explicación detallada de las opciones que existen hoy en día para programar y diseñar GUI en Python bajo sistemas libres o privativos, con el fin de tener un amplio panorama para el desarrollo de sistemas de información, programas computacionales e incluso aplicaciones Web. Este documento pretende enseñar a alumnos de ingeniería en computación las ventajas y opciones que ofrece el lenguaje de programación Python, así como la simplicidad que es escribir el código, la interpretación y la facilidad de leer el mismo. Para los diseñadores de GUI, la biblioteca Qt es una alternativa para ofrecer a sus usuarios y clientes la oportunidad de tener interfaces más sofisticadas y estéticas, para ejecutar sus programas en una computadora, tableta o dispositivo móvil. Esto gracias a los elementos que tiene la herramienta Qt Designer para realizar interfaces gráficas. Hoy en día los programadores para aplicaciones móviles buscan que su aplicación sea ligera, para que sus usuarios la puedan ejecutar en sus dispositivos. Esta tecnología se ha convertido en uno de las fuentes de empleo más prometedoras, ya que estas aplicaciones también se pueden desarrollar para diferentes plataformas, e incluso hasta desarrollar juegos. Recordemos que el lenguaje de programación Java era líder en el desarrollo de juegos hace algunos años, y poco a poco ha ido perdiendo terreno por que existen muchos otros lenguajes como Python, que compiten con él y ofrecen características atractivas, como la sencillez.

1.4. Objetivo

1.4.1. Objetivo General

Realizar una presentación general del lenguaje de programación Python, y mostrar el proceso de desarrollo de aplicaciones multiplataforma con interfaz gráfica usando la biblioteca Qt a través de ejemplos, para fomentar en la comunidad académica de ingeniería en computación el uso de este lenguaje de programación.

1.4.2. Objetivos Específicos

Los objetivos específicos de este trabajo son los siguientes.

- Mostrar los fundamentos del lenguaje de programación Python.
- Realizar una investigación documental sobre las herramientas de desarrollo para Python y la biblioteca Qt.
- Presentar ejemplos simples pero completos sobre el desarrollo de aplicaciones con interfaz gráfica con Python.

1.5. Alcance

Este documento no pretende ser una guía de iniciación para aprender a programar, sino un apoyo para mostrar cómo desarrollar aplicaciones con interfaz gráfica con el lenguaje de programación Python y la biblioteca Qt. Los ejemplos que se presentan son simples pero completos.

Capítulo 2

Lenguaje de programación Python

Hoy en día, el uso de las computadoras como herramienta de trabajo, para enviar correos, para búsquedas en Internet, chat o juegos videojuegos, se han convertido en parte cotidiana de nuestra vida. Muchas personas contamos actualmente con una computadora o teléfono inteligente, los cuales usamos como almacenamiento y reproducción de música, fotos, videos, etc., algunos otros la usan para su trabajo como capturar datos, almacenar bases de datos, guardar historiales clínicos etc., y otros más lo utilizan para desarrollar programas que simplifiquen la realización de tareas.

En este capítulo se presentan las nociones básicas de programación en Python. Este lenguaje de programación resulta ser muy sencillo y fácil de aprender por su simplicidad y escritura.

Python es un lenguaje programación interpretado y multiplataforma, de propósito general, que ha existido desde hace un tiempo. Un lenguaje interpretado, esto se refiere a que el programa se interpreta por el código que lo conforma. Además de usarse para desarrollo Web, también es útil para desarrollar GUI de escritorio y crear videojuegos.
Guido Van Rossum, es el creador de Python, lo comenzó a desarrollar en 1990, por lo que este lenguaje estable y maduro actualmente. Python también es considerado como un lenguaje dinámico, esto significa que la compilación se realiza en tiempo de ejecución. Es orientado a objetos, significa que se ejecuta en las principales plataformas de Hardware y de los distintos sistemas operativos, por lo cual no está restringido. Programar con Python es muy productivo, ya que en cada fase del desarrollo ofrece un análisis, diseño, creación, ajustes, documentación, implementación y mantenimiento [9].

En resumen, Python tiene las siguientes características:

- De alto nivel, esto indica que leer y escribir en Python es sencillo
- Interpretado, es que no necesita un compilador para escribir y ejecutar Python, lo cual permite ahorrar tiempo en desarrollo y no es necesario compilar ni enlazar.
- Orientado a Objetos, esto es que permite a los usuarios manipular estructuras denominadas objetos para construir y ejecutar los programas.
- Python ofrece mucho mayor prueba de errores comparado con el lenguaje C.

- Python tiene incorporado arreglos de tamaño flexible y diccionarios.
- Permite trabajar los programas por módulos, estos mismos se pueden reutilizar en otros programas.

Figura 2.1: Logo de Python

En Python se puede escribir un programa o script para simplificar alguna tarea, estos funcionan mediante el interpretes de comandos. Los scripts se usan para mover y modificar archivos, es por esos que Python ofrece mucho mayor soporte y estructura para programas grandes.

2.1. Excepciones para manejo de errores

Los errores comunes pueden gestionarse en Python mediante excepciones. Cuando el intérprete se encuentra en una situación errónea específica, se lanza una excepción, donde se puede informar al usuario que se existe un problema. La excepción se interpreta, y se muestra en consola de manera que el programador lo pueda solucionar.

Existen diferentes tipos de errores como los errores de sintaxis, errores lógicos y errores de ejecución.

- Errores de sintaxis son comunes que se presenten durante la escritura del programa, estos errores se pueden detectar cuando se compila y ejecuta el programa.
- Errores lógicos son el resultado de una incorrecta expresión formulada, estos tipos de errores son más complicados de detectar cuando se trata de códigos de grandes.
- Errores de ejecución o tiempo son cuando no se contemplan todos los casos posibles para tratar una sentencia.

Las excepciones son errores que ocurren durante la ejecución del programa, estas tienen la función de informar al usuario que ocurrió un error. En Python existen las delaciones de **except** y **try** para el manejo de las excepciones. La excepción con cláusula **except** da como resultado un mensaje error en el código. La sentencia **try** declara un fragmento de código para producir una excepción. Las excepciones con una cláusula **except** en una sentencia **try** muchas veces se confunden con errores, pero no lo son. Este tipo de declaración se puede utilizar para envolver todo el código o partes del código, para encontrar e identificar errores dentro del código [26].

2.2. Fundamentos de python

Python resulta ser mas accesible y la implementación del algoritmo complejos lo realiza mucha más rápido comparado con C y C++. Python resulta ser una excelente opción para desarrollo de prototipos ya que permite la reducción de tiempo y costos. Python es similar a los lenguajes de programación Perl y Ruby. Los programas Python se almacenan en archivos o scripts. El termino de Scripts se puede para archivos de código fuente desarrollados en Python, esto suele llamarse así tanto a lenguaje como al interprete [7].

Algunas ventajas de Python son:

- Python es mucho más sencillo y practico que C y C++.
- La funcionalidad de Python es prácticamente lo mismo que C y C++.
- Python puede hacer unión con las librerías de C y C++.
- Desarrollar en Python es diez veces más rápido que en C y C++.
- Los programas de C y C++ también pueden absorber el código de Python.
- Python y C/C++ pueden trabajar de manera conjunta [4].

2.2.1. Sintaxis de Python

Con relación a la sintaxis, se caracteriza por la simplicidad del lenguaje, es otras palabras es sencillo de escribir y fácil de leer, esta característica se convierte en una ventaja importante ya que ayuda a mantener el código comprensible.

- Python posee la característica de tener tipiado dinámico, ¿qué es esto? que las variables se fijan en el momento que se asignan. La ventaja de esto es que las variables pueden cambiar su tipo sin necesidad de declarar
- Python soporta la programación orientada a objetos (POO), pero ese no es el único lenguaje de programación que permite, también permite soporta la programación funcional y la imperativa, convirtiéndose en un lenguaje con más facilidades para enseñar la programación orientada a objetos. Con todo esto Python en su sintaxis contiene los mecanismos de introspección, y soporta la implementación de herencia sencilla y múltiple.
- Una de las diferencias más sobresaliente es el uso de espacios, que en otros lenguajes se utiliza sólo para separar o identificar las sentencias de un mismo bloque. Cada sentencia no se finaliza con un punto y coma (**;**) como sucede en los lenguajes de C/C++, Java y PHP. Las sentencias en Python tienen que ir en líneas diferentes para identificarlas
- Python no hace uso de llaves {} para indicar que inicia un bloque de sentencias, tampoco el uso de palabras como **begin** o **end**, para inicia o terminar un bloque. En Python únicamente se utiliza los dos puntos (**:**) para indicar el inicio, y el final lo hace el sangrado (espacios antes de la primera palabra).

- La programación en Python se hace sencilla por el uso de estructuras de datos de alto nivel como, por ejemplo: listas, diccionarios, cadenas de texto mejor conocidas como **strings**, **tuplas** y conjuntos.
- Python también cuenta con una biblioteca estándar que abarca desde los **strings** hasta la programación criptográfica, también puede trabajar con archivos de tipo ZIP y CSV, y realizar comunicaciones por red a través de varios protocolos. Todo esto se puede hacer con Python sin tener que instalar muchas bibliotecas.
- En Python existe un recolector de basura llamado *garbage collector,* con este recolector no es necesario indicar que libere memoria, el interprete lo realizara de manera automática por que se encarga de la administración de la memoria [7].

2.2.2. Palabras clave o reservadas

Las **palabras reservadas** o clave son identificadores **reservados** que ya están predefinidos. La Tabla 2.1 se muestra. estas palabras [7, 4].

Cuadro 2.1: Palabras reservadas

Palabra clave	**Tipo**	**Descripción**	**Ejemplo**
and	Operador lógico	representación lógica «y», retorna valores que está comparando	Si a y b son verdaderos entonces la salida es verdadera. 'a' **and** 'b'
Assert	Expresión booleana	Afirma condiciones para ser evaluadas.	**assert**(Promedio > = 8), **print** "Calificación aprobatoria!"
Break	Bucles	finaliza la ejecución de un ciclo o bucle.	var = 8 **while** var > 0: var = var + 1 **if** var == 4: **break** **print** ("El valor es: " + str(var))
Class	Clase	Declara y define clases.	**class** ClaseEjemplo: declaración 1 : declaración n
Continue	Bucles	Suspende una ejecución para continuar y resumir con otro elemento	var = 10 **while** var > 1: var = var + 1 **if** var == 5: **continue** **print** ("El valor es: " + str(var))
Def	Funciones	Declara funciones	**def** Mi_Funcion_Ejemplo () : **print** "Hola" **funcion()**
Del	Función	Elimina referencias, esto solo en caso de utilizarse seguido de un objeto. En diccionario elimina elementos, en listas re-ordena los elementos hasta llegan a un espacio vació.	Nombre= {Jose, Pedro, Luis} **del** Nombre{Luis}
Elif	Condicional	Se utiliza para evaluar las instrucciones en el caso de que sea falso, en caso que la sentencia es positiva se ejecuta las instrucciones correspondientes.	**elif** var > 10: **print** "No has es 10" **else:** **print** "adios"
else	Condicional	Permite distinguir cuando se respeta una indicación y cuando no, esto se realiza en dos bloques distintos, en otras palabras, la instrucción else revisa las condiciones.	**if** (a 1 != 5): **print** "es diferente de 5" **else:** **print** "la variable es igual a 5" **print** "Adios"

Palabra clave	**Tipo**	**Descripción**	**Ejemplo**

except	Excepción	Captura todas la excepciones posibles durante una ejecución de un bloque.	**def** suma(numero)
			try:
			  return math.sqrt(numero)
			except ValueError:
			  return 'Error: no se puede calcular'

exec	Método	Permite la ejecución del código Python a patir de objetos (cadenas) globales y locales archivos.	mi_programa = **input**('iniciar':) **exec**(mi_programa)

finally	Excepción	Permite abortar una excepción.	**try:**
			  bloque_instrucciones_1
			except Excepción_1
			finally:
			  final_del_bloque

for	Bucle	Permite el recorrido de elementos de una secuencia de un objeto.	**for** a = [1,2,3] **print**('Bienvenido', end = ' ') **print**('Final')
from	Modulo	Importa elementos de uno o mas elementos del módulo	**import** modulo **from** modulo
global	Tipo de variable	Sirve para declarar variables de tipo globales, lo cual significa que se puede tener acceso en cualquier parte del código	**global** x x = 1 **def** resul (x, y): **return** x + y
if	Bucle	Permite evaluar casos	**if** color == verde: **print** 'puedes pasar'
import	Módulo	importa un módulo que pertenecen y los que no pertenecen a un paquete.	**import** modulo #importa modulo que no pertenece al paquete. **import** paquete.modulo # lo que eta dentro del paquete.
in	Operador	determina la existencia de valores dentro de las listas, tuplas y diccionarios	2 **in** [0, 1, 2, 3] >> TRUE 2 **in** [0, 1, 3, 4] >> FALSE
is	Comparador	Determina la igualdad de dos objetos. Devuelve un True si las dos variables a comparar apuntan al mismo objeto	x = [0, 1, 2, 3] y = x y **is** x >>TRUE
lambda	Función	Crear funciones de manera rápida, unicamente operadores y comprobaciones.	**lambda** argumento: resultado **def** funcion(): **return** **print** funcion() funcion = **lambda**

Cuadro 2.2: Palabra reservadas(1 continuación)

not	Operador lógico	Operador de negación «no» Devuelve un valor verdadero si se esta negando.	**not** false >> true
or	Operador lógico	representación lógica «o» y retorna los valores que esta comparando	False **or** false >> False
pass	Bucles	Se utiliza para rellenar espacios, por tal no tiene función alguna.	**while** True: **pass**
print	Salida de programa	imprime en pantalla una determinada cadena.	**for** x in **range** (0, 5): **print**(”*”, end = ””)

raise	Excepción	lanza excepciones	**try:**
			  raise Excepcion(” ”)
			except:
			  log_error()
			  raise

return	Sentencia	Pertenece a una función y regresa un valor en especifico.	**def** peri(lado):
			  suma = lado +lado1
			  return suma

try	Excepción	captura una excepción dentro del código.	**try:** excepción_de_fallo()
while	Bucles	ejecuta una excepción de código mientras la condición expresada sea verdadera.	**while** x >= 5: **print** (x)
with	Contexto	Encapsula la ejecución de un bloque de código	**with** x **as** y: z = y()

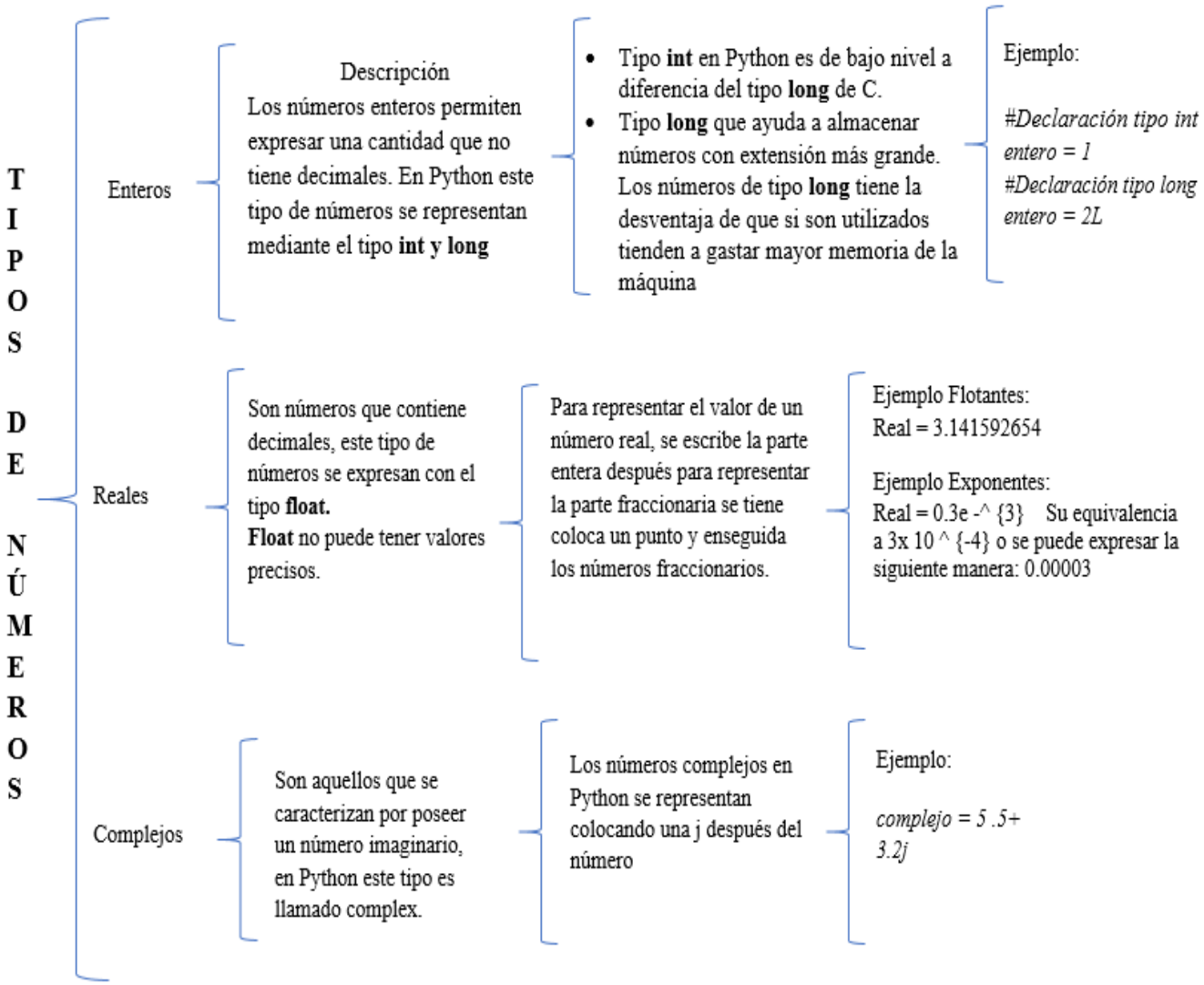
yield	Función	La función que se realiza es muy parecida al *return*, pero el valor que regresa elementos que conforman un generador, esto permite que se utilice muchas veces en una sola función	**def** funcion():
			  x = 1
			while a < funcion:
			  yield a
			  x = x + 1

Cuadro 2.3: Palabras reservadas(2)

2.3. Números

En Python se puede trabajar distintos tipos de números, entre ellos se encuentran enteros, reales (o también denominados coma flotantes), y complejos. Es importante definir que tipo de número se emplea, para evitar errores, por su parte Python tiene una ventaja de intentar decidir qué tipo de número es el más apropiado [3, 6][11].

Cuadro 2.4: Tipos de números



2.4. Operadores

En nuestra vida diaria es esencial utilizar los números y más si van acompañados por algunas operaciones matemáticas, situaciones como contar el dinero, para hacer una tarea, etc., estas situaciones nos hacen pensar y realizar cuestiones matemáticas, en programación también es usual hacer uso de las matemáticas y Python no es la excepción, Python cuenta con módulos para diferentes tipos de operaciones, como *math* y *cmath* [6].

El módulo *math* en esencia proporciona las funciones matemáticas propias y definidas por el estándar de C, *math* permite emplear funciones como hipérbolas, trigonométrica y logarítmicas, esto para números reales [23].

El módulo *cmath* está disponible para proporcionar acceso a las funciones matemáticas con números complejos, este módulo también acepta números enteros, flotantes o números complejos con argumentos [23].

2.4.1. Operadores aritméticos

Los operadores aritméticos son los más sencillos, se utilizan para realizar operaciones sencillas básicas, como suma, resta, multiplicación y división, módulo o residual y exponenciales. Python utiliza los siguientes operadores aritméticos para calcula el valor, superior, inferior o absoluto de los números siempre que se utilicen las funciones de los operadores. Ver Cuadro 2.5

Cuadro 2.5: Operadores aritméticos

Operador	Descripción	Ejemplo a=5, b=3
+	Suma	x = a + b
-	Resta	x = a - b
-	Negación	x = - a
*	Multiplicación	x = a * b
**	Exponente	x = a ** b
/	División	x = a / b
//	División entera	x = a // b
%	Módulo	x = a% b

La diferencia entre división y división entera, en la primera de devuelve el resultado de un número real y en la división entera es solo el resultado que se devuelve en la parte entera.

2.4.2. Operadores de comparación

Este tipo de operadores comparan los valores que tienen en cada lado y el resultado que puede demostrar si es falso o verdadero. A este tipo de operadores también se les llama operadores relacionales, en el Cuadro 2.6 se muestran los operadores y lo que realiza cada uno [23].

Cuadro 2.6: Operadores de comparación

Operador	Significado	Descripción
==	Igual a	Evalúa como verdadero si dos variables son iguales
!=	Distinto de	Evalúa como verdadero si dos variables son diferentes
< >	lo mismo que	Si los valores de dos operandos no son iguales, entonces la condición se convierte en verdadera.
>	mayor que	Si el valor del operando izquierdo es mayor que el valor del operando derecho, entonces la condición se convierte en verdadera.
<	menor que	Si el valor del operando izquierdo es menor que el valor del operando derecho, entonces la condición se convierte en verdadera.
> =	mayor o igual que	Si el valor del operando izquierdo es mayor o igual que el valor del operando derecho, entonces la condición se convierte en verdadera.
< =	menor o igual que	Si el valor del operando izquierdo es menor o igual que el valor del operando derecho, entonces la condición se convierte en verdadera.

2.4.3. Operadores lógicos

Los operadores que se utilizan en Python son muy conocidos ya que se utilizan en otros lenguajes de programación, en Python existen tres tipos de operadores lógicos **and**, **or** y **not.** Este tipo de operadores permiten unir varias condiciones para construir expresiones lógicas, y estas expresiones dan como resultado un booleano.

Es importante para que para los bucles se expresen estas condiciones, los tipos booleanos son además una expresión los tipos relacionales. Y recordemos que los booleanos únicamente tienen dos valores True y False, que en español se traduce como cierto y falso respectivamente [6],[5].

Cuadro 2.7: Operadores lógicos

Operador	Símbolo	Descripción	Resultado
AND	&	Que significa «Y»,	El resultado de AND es verdadero solo cuando operados lo son.
OR	|	Que significa «O»	El resultado de OR es verdadero solo cuando cualquiera de los operadores lo es.
OR	~	Que significa «NO»	El operador NOT invierte el valor del operando.

2.4.4. Operadores a nivel de bit

En todas las computadoras están presentes los bits, que son una serie de ceros y unos, con estos se representan los números. Los operadores de nivel de bit operan directamente en los bits, a continuación se muestra este tipo de operadores y su descripción.

El operador **&** o **AND** ya se había mencionado antes devuelve el valor de 1 siempre que el valor operando sea 1 y 1, en caso contrario devuelve un 0.

El operador **OR** o bien | este regresa el valor de 1 con la siguiente condición: si el primero es 1 o el segundo es 1, en caso contrario se regresa 0.

El operador **XOR** o también denominado exclusión regresa el valor de 1 si uno de los operandos es 1.

El operador **NOT** niega uno a uno cada uno de los bit, en otras palabras si existe un 0 cambia 1 y si es 1 cambia 0.

Los operadores de desplazamiento son utilizados para desplazar los bits a cualquier posición ya sea izquierda o derecha.

2.5. Cadenas

En esta apartado se demuestra la manipulación de cadenas de caracteres o también denominado ***string***. Explicar cómo manipular las cadenas de texto es fundamental, porque de eso depende que la construcción del código sea limpio y legible.

Una cadena de caracteres se define como la secuencia de elementos ordenada como letras, números o símbolos, las letras de "a" hasta la "z" mayúsculas o minúsculas, los números del "0" al "9", los símbolos son diversos, se considera también el espacio en blanco como carácter [4].

En otros lenguajes de programación para declarar que se va a utilizar un tipo cadena o string, es necesario declarar el tipo de variable, en el caso de Python no es necesario declarar, par distinguir que en Python estas usando cadenas solo se coloca la cadena entre comillas.

En Python existen varias formas de definir las cadenas, la primera es con comillas dobles ("), este tipo de comillas es muy común usarlas en otros lenguajes de programación.

```
texto = "esto es una cadena"
```

También se utilizan la comilla simple ' , dentro de las comillas simples o dobles pueden estar otros caracteres especiales o también denominados caracteres de de secuencia de escape, como para indicar una nueva línea o el de tabulación, esto caracteres especiales se representan con \. La secuencia para un salto de línea se representa con "\n".

Una cadena puede contener un carácter antes, como r o u, esto significa que la cadena que si tiene el carácter u pertenece a una codificación Unicode y si es r es para una cadena de tipo **raw**, indicada en expresiones regulares [7].

Es decir:

- Con el carácter 'r' puede obligar a secuencia de escape a que no se interpreta de manera incorrecta, como en el siguiente ejemplo: r 'Es \n cadena'. El \n no lo convierte a un salto de línea.
- Con el carácter 'u' obliga a que la cadena sea convertida a un carácter Unicode.

Las cadenas en Python son inmutables, esto quiere decir que no se pueden realizar modificaciones, en caso de querer hacer modificaciones es necesario construir o declarar una cadena nueva

Ejemplo:

```
cadena [1] = "Hola"
cadena [2] = "Hi"
```

2.5.1. Operaciones simples con cadenas

Python puede hacer operaciones y funciones comunes con las cadenas, que permiten que se manipule la cadena y devuelva cadenas como resultado. En el Cuadro 2.8 se muestran los operadores que pueden tener efecto en las cadenas.

Cuadro 2.8: Operadores en cadenas

Operador	Significado	Descripción	
+	Concatenación de cadenas	La concatenación necesita dos cadenas como operandos y resultado de la concatenación es que devuelve una cadena uniendo la segunda cadena con la primera.	Etiqueta1 = 'Hola' + ' ' + 'amigos' print (etiqueta1) >> Hola amigos
*	Repetición de cadena	Este operador acepta una cadena y un número entero y devuelve la concatenación de la cadena con las veces que el número entero se haya declarado.	Etiqueta1 = 'niños' * 2 print(Etiqueta1) >>niños niños
%	Sustitución	Este operador acepta una cadena y expresiones, estas expresiones pueden ser una o varias. Las expresiones se colocan entre paréntesis y separadas por comas. El valor que devuelve donde las marcas de formato se sustituyen por el resultado de la evaluación de las expresiones, las expresiones pueden ser %d, %f, etc.	Etiqueta1 = 2 Print 'número %d y número %d' % (0,Etiqueta1) >>número 0 y número 2

Las funciones en cadenas son posibles en el Cuadro 2.9 se muestra las funciones que pueden utilizarse en las cadenas.

Cuadro 2.9: Funciones en cadenas

Función	Descripción
int	Esta función recibe una cadena con secuencia de dígitos y devuelve un número entero
float	Recibe contenido la cual tiene una descripción flotante y devuelve el valor en flotante
str	Recibe valores flotantes o enteros y devuelve una secuencia con representación de los valores recibido en forma de caracteres
ord	Recibe cadenas compuestas por un único carácter y lo devuelve en código ASCII
chr	Recibe enteros de 0 al 255 y devuelve el valor en entero de acuerdo al código ASCII

Se puede manipular cadenas mediante sus métodos propios de las cadenas. En el siguiente Cuadro 2.10 se muestra algunos de los métodos[10].

Cuadro 2.10: Métodos para cadenas

Método	Descripción
a.lower( )	Convierte los caracteres a minúsculas
a.upper( )	Devuelve los caracteres convertidos a mayúsculas
a.capwords( )	Devuelve cadenas en las que la letra inicial la convierte en mayúscula.

2.5.2. Unicode

El unicode es un sistema para detectar los caracteres de todos lo lenguajes. Esto significa que representa cada carácter, letra, número en cuatro bytes, los número van a representar a un carácter y este debe de ser único, por lo tanto existen un número para cada carácter y viceversa.

A partir de versión 2 de Python se implementa el unicode para el reconocimiento de caracteres. La ventaja de poder utilizar unicode es que se puede utilizar un número ordinal para cada carácter, tanto en textos antiguos como en textos modernos.

El UTF-32 es una tabla de codificación unicode, que utiliza cuatro bytes por carácter. Esta toma un número de 4 bytes y el carácter lo representa con el mismo número. Existen muchos unicode que en su mayoría no se usan, como el UTF-16 este codifica 2 bytes en cada carácter del 0 al 65535 y cuando se requiere utilizar caracteres raros para representarlos más allá de sus parámetros. UTF-16 no es exactamente el ideal, esto debido que la mayoría de sus caracteres son ASCII.

UTF-8es un sistema de codificación de longitud variable y tiene la ventaja de que los caracteres pueden utilizar diferente número de bytes. Para aumentar su ventaja UTF-8 tienen los mismos bytes

que ASCII. Los caracteres raros utilizan cuatro bytes, los caracteres latino como la "ñ" y "ü" utilizan dos bytes y los caracteres chino utilizan tres bytes. La desventaja es que cuanto más larga es la cadena, más tiempo se llevará para encontrar la cadena en un carácter específico y se tendrá que ir codificando y decodificando entre bytes y caracteres [12].

2.5.3. Secuencias de escape para cadenas

Las sentencias de escape se utilizan para avisar que lo que sigue es un carácter especial. En el Cuadro 2.11 se observa las secuencias de escape(en este link encontraras más información `https://docs.python.org/2/howto/unicode.html`). La barra invertida se llama carácter de escape o también denominado carácter especial, este indica que el siguiente carácter tiene una acción diferente a las comunes por ejemplo la letra «n» significa «new line», es decir nueva línea. Un salto de línea indica que es un único carácter [10].

Cuadro 2.11: Secuencias de escape

Secuencia de escape	Resultado en pantalla	Función
\\	\	Carácter de barra invertida
\'	'	Comilla simple
\"	"	Comillas dobles
\a	BELL	Carácter de campana (BELL)
\b	BS	Retroceso o espacio hacia atrás
\f	FF	Avance de página
\n	LF	Salto de línea
\N{name}	Sólo unicode	Carácter name en la base de datos unicode
\r	CR	ASCII retorno de carro
\t	TAB	ASCII tabulación horizontal
\uxxxx	Sólo unicode	Carácter con 16 bits hexadecimal y un valor xxxx
\Uxxxxxxxx	Sólo unicode	Carácter con 32 bits hexadecimal y un valor de xxxxxxxx
\v	\v	ASCII tabulación vertical (VT)
\ooo	ooo	Carácter con valor octal
\xhh	hh	Carácter con valor hexadecimal

2.5.4. Longitud de cadena

En este apartado mencionaremos la longitud de una cadena, para esto la función se emplea es ***len***, que Inglés es ***length***, qué significa longitud. Lo que realiza esta función es contar el número de caracteres que contiene una cadena, en otras palabras devuelve en forma de número los caracteres que forman la cadena [10].

La forma de utilizar esta función en muy sencilla y de manera directa. Observa cómo es esto:

```
leng("Hola")
```

```
#el resultado es un número
4
```

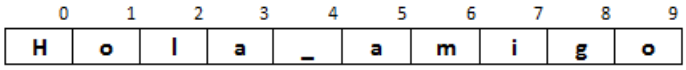
Hay cadenas que pueden tener un valor nulo o que devuelven un valor en 0, a este tipo de cadenas se les denomina cadena vacía o de longitud cero. es importante mencionar que pudiera confundirse una cadena vacía con un espacio en blanco, pareciera ser iguales pero no lo son. El espacio en blanco está considerado como un carácter y cuenta con una longitud de 1. Observa el siguiente ejemplo en donde se diferencia una cadena vacía y un espacio en blanco [10].

```
#cadena vacía
 len("")
#valor que regresa es:
0
#espacio en blanco
len(" ")
#valor que regresa es:
1
```

2.5.5. Indexación

El operador de indexación permite ingresar a uno de los caracteres de la cadena, esto se coloca dentro de un corchete. Se considera como índice el primer elemento de la cadena y este índice empieza en cero, en la figura 2.2

Figura 2.2: Indexación de cadenas

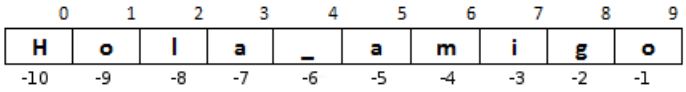


El primer carácter de la cadena tiene el índice 0 y ***len*** (a) caracteres, entonces el último carácter debe de tener ***len***(a)-1, es porque se quiere acceder a ***len***(a), Python devolverá un mensaje de error indexado porque se sale de rango de parámetros válidos.

La cadena "Hola amigo" ocupa 10 casillas, y también hay que recordar que las sentencias de escape están codificadas como caracteres simple y expresan un solo carácter, aunque pareciera que ocupa dos.

Se pueden utilizar índices negativos con un significado especial, esto es, los valores negativos pueden acceder a los caracteres de derecha a izquierda. El último carácter de la cadena tiene el índice -1, el penúltimo el -2 y así sucesivamente, observa el siguiente ejemplo de la Figura . Con este método es más simple acceder a los caracteres de la cadena, se puede considerar como que se tiene un doble índice.

Figura 2.3: Ejemplo indexación



2.6. Conjuntos

Los conjuntos son ampliamente utilizados en lógica y matemáticas, y desde luego en los lenguajes de programación se explota sus propiedades, en Python las propiedades de los conjuntos da oportunidad de crear códigos más eficientes y claros. Un conjunto es una colección de objetos que no es ordenada, por defecto Python tiene estos tipos de datos, por ejemplo: listas, tuplas y diccionarios.

Para expresar un conjunto se colocan los elementos entre llaves, como se observa el el siguiente ejemplo.

```
c = {2, 4, 6, 8}
```

Python acepta diversos tipos de por ejemplo:

```
r = {Falso, 1.5, None, True, "hola mundo", (2,3)}
```

Python distingue entre los tipos de operación, para generar un conjunto vacío se crea la instancia de la clase **set**, y se obtener objetos a partir de estos conjuntos.

```
x = set()
```

Un set puede convertirse en listas y viceversa, en el caso de los elementos duplicados, estos se unificados.

Existen elementos mutables de los elementos, ejemplo el método el **add( )** y **discard( )** permiten agregar y quitar elementos indicándose como argumentos[17].

```
x = {0, 1, 2, 3, 4, 5, 6}
x.add(7)
x.remove(0)
Y el resultado de esto es:
x = {1, 2, 3, 4, 5, 6, 7}
```

La palabra reservada **in** tiene el poder de definir si un elemento pertenece a un conjunto, ver el siguiente ejemplo:

```
0 in {1, 2, 3}
Flase
2 in {1, 2, 3}
True
```

Para eliminar elementos se utiliza la función **clear( )** ejemplo de esta función, observa lo siguiente.

```
x = {1, 2, 3}
x.clear()
x
set()
```

Para eliminar los elementos del conjunto de uno por uno se utiliza el método **pop( )**, este método devuelve elementos de forma aleatoria, se realiza de esta manera ya que los elementos no están ordenados. **KeyError** es una excepción del método pop() y **remove( )**, este se lanza cuando detecta que está en el conjunto o el conjunto se encuentra vació.

En ocasiones es necesario conocer cuántos elementos están dentro de el conjunto, y para agilizar este proceso se cuenta con la función **len( )**, que permite contar los elementos del conjunto, observa el ejemplo [17].

```
len({a, b, c, d, e})
5
```

2.6.1. Los operadores principales de los conjuntos

Los conjuntos son una colección de valores que no poseen un orden ni valores repetidos, también se puede ver como una asociativa, lo que en programación es **array.** Los diversos tipos de datos se pueden implementar en los conjuntos, además de poder utilizar las operaciones como unión, inserción y diferencia.

La **unión** es definida como el conjunto de elementos que se encuentran en uno u otro, o en su defecto en ambos. la unión se realiza con el carácter **|** el valor que regresa es el que contiene al menos uno de los conjuntos en cuales se realiza la operación [17, 14].

```
x = {a, b, c}
y = {b, c, d}
x | y
{a, b, c, d}
```

La inserción opera de manera similar, pero utilizando el operador **&** y el valor que regresa es un nuevo conjunto que tiene los elementos que se encuentran en ambos conjuntos. Observa el ejemplo, tomando como referencias los conjuntos del ejemplo de unión[17].

```
x & b
{b, c}
```

La diferencia de dos conjuntos, que en este caso son x, y la diferencia está en que los elementos que se encuentran en $\boldsymbol{x}$ no se encuentran en $\boldsymbol{y}$, en Python se emplea el símbolo **-** (menos) que es un operador de resta simple.

```
x = {a, b, c, d}
y = {a,d}
{b, c}
```

2.6.2. Más operaciones

Una de las fabulosas estructuras de Python son los subconjuntos, la teoría de los subconjuntos nos dice "A es subconjunto de B cuando A está incluido en B", en otras palabras quiere decir que B contiene a A.

En Python esto se puede definir con el método **issubset( )**. Observa el ejemplo.

```
x = {a, b, c, d}
y = {b, c}
y.issubset(a)
True
```

Esto se puede expresa inversamente lo que quiere decir que las dos relaciones son todo un conjunto y que al mismo tiempo un subconjunto.

La diferencia simétrica se puede entender como una unión exclusiva, esto quiere decir que un nuevo conjunto contiene elementos que pertenecen a alguno de los conjuntos dentro de la operación, pero no pueden ser ambos. en Python esto se expresa de la siguiente manera:

```
x = {a, b, c, d}
y = {c, d, e, f}
x.symmetric_difference(b)
{a, b, e, f}
```

Otra operación son los conjuntos disconexo, esto es que los conjuntos no comparten elementos entre sí, es decir si su intersección es vacía.

2.6.3. Conjuntos inmutables

La clase **set** tiene un implemento similar pero inmutable llamado **frozenset**. Es decir funciona con todas las operaciones antes mencionadas excepto los elementos de **add()** y **discard()**, ya que estas alteran sus elementos. Esto permite que los conjuntos empleen claves en los diccionarios[17].

Un *hash* es un número entero que representa al objeto. la probabilidad de que un *hash* esté dentro un objeto distinto, es muy alta. En los números enteros un *hash* aparece de manera aleatoria.

2.7. Comentarios en Python

Al igual que otros lenguajes de programación los comentarios sirven para explicar el código y se pueda explicar partes del código. Los comentarios son solo unos mensajes que coloca el desarrollador para que el código se entienda. Estos comentarios son ignorados cuando es ejecutado el programa ya que no se considera parte del código. Colocar comentarios en el código resulta ser una tarea tediosa, pero resulta ser una práctica muy buena, ya que con el tiempo se puede olvidar para que sirve cierta línea de código.

En Python los comentarios pueden ser de dos maneras:

- Colocando el símbolo almohadilla o numeral (#) antes del código, por ejemplo

- #print (“saludos”)

- Colocar triple comilla doble al principio y al final del código, esto se puede emplear en una sola línea o abarcar más de una línea. Por ejemplo:

  “”“numero = int (self.aut.lineEditInsertar.text())

  resul = 1

  for i in range(numero): “”“

En este último existe una contrariedad, ya en algunas referencias se puede encontrar que los comentarios únicamente se pueden considerar con las #, y el uso de las comillas dobles es solo para cadenas con múltiple línea.

2.8. Funciones

En Python es posible crear funciones dentro de una estructura de una función más extensa. En otras palabra la función interna es propiedad de la función externa. Así mismo las funciones son objeto de una primera clase, lo que significa que estas clases pueden contener atributos y pueden asignarse a una variable. A esto se le denomina variables de alto nivel y que pueden retornar valores.

Es posible crear funciones a través de la palabra clave **lambda**, esta función es mayormente utilizada para definir las funciones de manera completa, un ejemplo de esta función es cuando de manda a llamar la función y no como la declaración. La función **lambda** puede definir una función anónima pero solo como expresión y se puede definir dentro de la misma línea de la llamada de la función [2].

Una función es una sentencia de sentencia que ejecuta una operación deseada, las funciones de Python se definen por la instrucción **def**, la sintaxis para definir una función en Python es la siguiente:

```
def nombre_funcion (parámetros de la función):
sentencias o algoritmos
```

Como en toda estructura de Python el nombre de la función está seguida de paréntesis, el algoritmo que realizará la función va dentro de los paréntesis indentado con 4 espacios y finaliza con dos puntos (:) Las funciones en Python, no funcionan por sí solas, en el ejemplo anterior se muestra como se declara una función pero que esta función se ejecute es necesario invocar a esta función. En el siguiente ejemplo se muestra la invocación de la función [8]:

```
def nombre_funcion (parámetros de la función):
sentencias o algoritmos
nombre_funcion()
```

Las funciones permiten devolver valores, esto se almacenan en una variable indicando y asignado que en esa variable se almacena el valor de la función. Observa el siguiente ejemplo:

```
def nombre_funcion ():
return "valor que regresa"
```

```
mensaje = nombre_funcion()
print mensaje
```

2.8.1. Parámetros

Las condiciones para declarar una función y a sus parámetros son las mismas que se utilizan para dar un nombre a una variable, pero se pueden utilizar letras, dígitos y el carácter de subrayado. Cuando se usan letras no deben de emplearse palabras reservadas, tampoco la primera letra del nombre puede contener un número. Cuando declara una función y una variable se debe de tener cuidado ya que no se puede nombre a la función y ala variable del mismo modo, en Python esto se debe definir e identificar de manera única para cada función y cada variable.

Los parámetros son los valores que esperan recibir de las funciones, esto sólo puede suceder cuando se invoca dicha función. Las funciones pueden ser una o varias las cuales se pueden separar por comas y se definen en los paréntesis como si fueran variables, mismas que se van a utilizar dentro de la función.

2.8.1.1. Parámetros de variables locales

Las variables de ámbito local son las que se utilizar en las funciones y estos mismo parámetros son los que utiliza la función. En otras palabras son variables locales que únicamente la función podrá tener acceso [27]. En el siguiente ejemplo se puede observar las variables locales:

```
def funcion(nombre, apellido):
        NombreCompleto= nombre, apellido
        print NombreCompleto
```

Es importante conocer que no se puede acceder a estas variables locales fuera de la función, si esto pasa se genera un error.

Cuando una función es llamada, es necesario pasar sus argumentos en el mismo orden en el que los espera, pero también es posible evitar esto, utilizando como keywords. La palabra clave se distinguen si utilizas mayúsculas o minúsculas.

En Python se puede agregar valores por defecto, también llamados parámetros por omisión, esto significa que la función podrá ser llamada con algunos de sus parámetros, no precisamente todos los que espera.

2.8.1.2. Parámetros arbitrarios

Los parámetros arbitrarios es cuando la función espera recibir un número de forma desconocida o arbitraria, esto llegan en forma de tupla a la función. Se pueden declarar argumentos arbitrarios en la funciones, para identificar que son parámetros arbitrarios se les coloca un asterisco (*), en el siguiente ejemplo se puede observar [10]:

```
# función con parámetros arbitrarios
def funcion(fijo, *arbitrario):
```

```
        print fijo
#los parámetros arbitrarios se expresan como tuplas
for parametro in arbitrario:
        print parametro
funcion("arbitrario1","arbitrario2")
```

2.8.1.3. Parámetros con palabras clave

Los parámetros arbitrarios siempre deben de seguir a los fijos, esto para que la función espere recibir ambos parámetros. También posible recibir y obtener valores arbitrarios como pares con keywords expresado de la siguiente manera:

```
palabra_clave=valor
```

Para este tipo de casos al parámetro se le colocan dos asteriscos (**), observa el siguiente ejemplo[10]:

```
# función con parámetros arbitrarios
def funcion(fijo , *arbitrario , **kwords):
        print fijo
for parametro in arbitrario:
        print parametro
# argumentos arbitrariosde tipo clave se recorren como si fueran diccionarios
for clave in kwords:
        print " valor de", clave , "es", kwords[clave]
funcion("arbitrario1","arbitrario2", clave1 = "valor 1", clave2 = "valor 2")
```

2.8.2. Flujo de ejecución

Puede existir gran cantidad de funciones programadas dentro de un programa, cuando esto sucede la cantidad de funciones para utilizar es muy grandes y puede tornarse complicado cuál de todas utilizar. Para evitar este enredo de funciones es importante documentar cada función para determinar qué tarea realiza, cuáles son los parámetros que recibe y que es lo que devuelve.

Una función es útil para realizar una o varias instrucciones ya sea con el mismo parámetro o con distintos [26].

2.9. Listas

En Python para agrupar texto compuestos existen varios elementos, entre ellos se encuentran las listas, estas pueden ser similares a las cadenas o *strings*. Una lista en Python es un conjunto de valores de manera ordenada por medio de listas, esto quiere decir que los valores contiene un índice. A los valores de la lista se les denomina elementos y esto pueden ser de cualquier tipo [26].

Una lista en Python es considerada como una estructura de datos con características especiales como almacenar cualquier tipo de valor, por ejemplo enteros, cadenas e inclusive funciones [10].

Para representar las listas en Python es muy sencillo, solo hay que colocar los valores de la lista dentro de unos corchetes y separados con comas. A igual que las cadenas es importante mencionar que el primer elemento de la lista tiene la posición 0. En la siguiente ejemplo se muestra una de listas de números y cadenas.

```
# Ejemplo de listas
n = [2, 4, 6, 8, 10]
#El resultado que devuelve la variable n es
n
[2, 4, 6, 8, 10]
```

Las listas también pueden contener y almacena cadenas, de modo que las cadenas es necesario agregar una referencias (o también llamados punteros en otros lenguajes de programación) a los elementos. Los elementos de la lista no siempre pueden ser del mismo tipo, observa el siguiente ejemplo [10].

```
#ejemplo de listas con números enteros
[1, 2, 3, 4]
#ejemplo de listas con cadenas de texto
["hola", "dia", "noche"]
```

Las listas con números enteros en sencillo construirlas y son las más comunes, en Python existe una forma de crear este tipo de listas con la función ***range*** se puede generar números enteros.

La función ***range*** puede recibir uno, dos o hasta tres argumentos numéricos de tipo entero. El tercer argumento se le conoce como paso o ***step***. Observa el siguiente ejemplo en el cual se muestra la función ***range***.

```
#función range con un parámetro
range (5)
[0, 1, 2, 3, 4]
#función range con dos parámetros
range (1, 6)
[1, 2, 3, 4, 5, 6]
```

Anteriormente se mencionó que en la función ***range*** puede aceptar tres argumentos, esto significa que se creará una lista con un rango de números y el tercer parámetro indicará el espacio entre los números. En el siguiente ejemplo se muestra cómo se usan las listas con tres parámetros al igual que un ejemplo [10].

```
#se definen dos parámetros enteros 'a' y 'b'
#a representa el primer elemento de la lista
#b representa el último elemento de la lista
#c será el espacio entre 'a' y 'b'
#Se define la función range con los parámetros dentro

range(a,b,c)
range(0, 10, 2)
# el resultado es una secuencia de números enteros con espacio de 2
[0,2,4,6,8]
```

Lo anterior también funciona con números enteros negativos:

```
rango_lista(-3, 3)
[-3, -2, -1, 0, 1, 2, 3]
```

Existen también las llamadas listas vacías, como el caso de las cadenas también puede existir un elemento vació, para representar este tipo de listas solo se coloca los paréntesis ().

Para añadir elementos a una lista, se coloca el operador + (más), y este crea una nueva lista a partir de las que ya existen. Recordemos que una lista puede contener cualquier número de elementos, ya que el límite de tamaño depende de la memoria que se tenga disponible. Para el caso de la nueva lista se agrega una nueva variable a la lista. A este paso de agregar una nueva lista se realiza en dos pasos, primero la concatenación y luego la asignación, esta última puede ocupar mucha memoria de manera temporal [12].

Una lista se puede considerar como una clase, cuando se crear una lista realmente estás iniciando una clase, ya que las listan también poseen métodos, mismos que sirven para realizar acciones sobre la lista. Existen métodos que pueden ayudar a la lista.

2.9.1. Métodos en listas

- ***append()***, este método añade un nuevo al final de la lista, este elemento es único.
- ***extend()*** obtiene un parámetro, una lista y agrega a la lista original cada uno de sus elementos.
- ***insert(i,x)*** inserta un elemento a la lista, este elemento debe ser único, donde i representa el índice y x el elemento que se va a insertar.
- ***count()*** devuelve el número de veces que aparece un valor en específico del parámetro que se encuentra en la lista .
- ***remove()*** elimina el primer elemento de la lista cuyo valor es x. Si no existe el elemento devuelve un error.
- ***index()*** regresa el índice en la lista el primer elemento, el cual tendrá el valor de x. Si no existe el elemento devuelve un error.

- ***pop([i])*** quita el elemento que se encuentra en una posición señalada en la lista. Si a este método no se le especifica un índice, este método elimina y regresa el último elemento de la lista. Lo que se encuentre dentro de los corchetes es la firma del método.
- ***sort()*** Regresa el número de veces que x aparece en la lista
- ***reverse()*** Invierte los elementos de la lista [12].

Ahora se muestra un ejemplo en donde se muestran la mayoría de los métodos

```
#Declaramos una lista con números enteros y utilizaremos el método count
lista = [1, 2, 2, 3, 4, 4]
print lista.count(1), lista.count(2), lista.count(3), lista.count(4)
#el resultado del conteo de los elementos dentro de la lista es:
1 2 1 2

#En el siguiente ejemplo se emplea el método insert y append
lista.insert(2, 0)
lista.append(5)
lista
[1, 2, 0, 2, 2, 3, 4, 4, 5]

#Método index
lista.index(4)
2

#Método remove
lista.remove(0)
lista
#El resultado es:
[1, 2, 2, 2, 3, 4, 4, 5]

#método reverse
lista.reverse()
lista
#el resultado es:
[5, 4, 4, 3, 2, 2, 2, 1]

#Método sort
lista.sort()
listas
```

```
#el resultado es:
[1, 2, 2, 2, 3, 4, 4, 5]
```

2.9.2. Acceso a la lista

Para acceder a los elementos de una lista, los elementos se encuentran dentro de los paréntesis, lo que esta expresado dentro de ellos especifica el índice y este siempre empieza en cero. Los **[]** corchetes, cuando se encuentran del lado izquierdo de una expresión, cambia uno de los elementos de la lista. Se puede considerar como índice cualquier expresión entera [1].

Existen excepciones dentro de las listas, por ejemplo cuando se requiere leer o modificar un elemento que en la lista no existe, el intérprete de Python devolverá un error.
Existen también los índices negativos, al igual que las cadenas el índice de las cadenas con número negativos comienza desde el final de la cadena. En las lista es lo mismo, los índices negativos comienzan en el último elemento de la lista, es decir se cuenta de atrás, empezando por el último elemento de la lista [2].

2.9.3. Longitud de lista

Como ya se ha mencionado antes en la sección de cadenas, es posible conocer la longitud de una cadena, lo mismo se puede hacer en la lista, para conocer la longitud de la lista se emplea la función **len**, esta función devuelve el tamaño de la lista a la cual se le asigna.

- Se puede tener acceso a los elementos de la lista se emplea la notación que las cadenas, recordemos que la notación va desde 0 a la longitud -1 de la lista
- Para que se pueda obtener una sublista de la lista original, es necesario utilizar la notación de rango.
- Es posible obtener el elemento de una lista colocando el índice del elemento que se desea.

2.9.4. Conmutar listas

Las listas son secuencias mutables, Python para lograr esto tiene operaciones que permiten cambiar, agregar y quitar los valores

Existen manera de añadir elementos a una lista anteriormente se mencionó los métodos que pueden tener las listas, algunos de estos métodos pueden hacer que se agreguen elementos a la lista, como son: ***append(), insert(), extend()***. Recordemos que una lista puede tener elementos de cualquier tipo y al mismo tiempo cada elemento puede ser de un tipo diferente [27].

- Para cambiar un elemento de la lista, primero se selecciona el elemento de la lista, mediante su índice y se le asigna un nuevo valor. Observa el siguiente ejemplo.

```
lista[1, 2, 3, 4, 5]
lista[2] = 8
#El cambio se realizó en el índice 2 y el elemento que cambió fue el número 3 por el
lista
[1, 2, 8, 4, 5]
```

- Ahora vamos a ver como se agrega un nuevo a elemento a la lista mediante el método ***append()***. Recordemos que este método agrega elementos al final de la lista. Observa el siguiente ejemplo

```
lista[1, 2, 3, 4, 5]
lista.append[6]
#los elementos de la lista son:
lista
[1, 2, 3, 4, 5, 6]
```

- El método ***insert()***, agrega nuevos elementos al alista mediante la indicación del índice. Esto quiere decir que el nuevo elemento insertado se desplaza una posición al resto de la lista. Las listas no toman en cuenta si los datos que se insertan de forma repetida, si se requiere que se restrinja esto, es necesario especificarlo en el código. Observa el ejemplo en donde se muestra el método ***insert()*** y lo que hace en la cadena [27].

```
Lista[10, 20, 40, 50]
lista.insert(2, 30)
lista
[10, 20, 30, 40, 50]
```

- Para eliminar un elemento de la lista se emplea el método ***remove()***. Recordemos que este método elimina un valor de la lista. Cuando hay valores repetidos el método ***remove()***, eliminar el primer elemento que se encuentra en su recorrido empezando por el índice 0 hasta recorrer toda la lista. En el siguiente ejemplo mostraremos un ejemplo donde existen valores repetidos [27].

```
lista[1, 2, 3, 3, 4, 5, 6, 6]
lista.remove(3)
lista
[1, 2, 3, 4, 5, 6, 6]
```

Puede darse el caso que el valor a eliminar no exista y en este caso Python devolverá una excepción de error.

2.10. Tuplas

En esta sección se hablará de una estructura de datos bastante parecida a una lista. Una tupla es una estructura ordenada de objetos. Podemos decir que una tupla es muy similar a una lista, pero con la diferencia que una tupla es inmutable, es decir una vez creada el contenido de la tupla no se puede modificar, pero es posible sobrescribir una tupla o dicho en otras palabras se volvería a crear una tupla.

Las tuplas al igual que las cadenas contienen índices para poder hacer referencia un elemento. Una diferencia importante de la lista y la cadena es que en la lista se puede modificar los elementos, pero en la cadena esto no es posible.

Se puede considerar que una tupla es una lista, los valores dentro de ella son separados por comas, incluso los valores pueden o no estar dentro de unos paréntesis, aunque en realidad la construcción de una tupla lo hace la coma (,) no el paréntesis; pero recordemos que el interprete de Python los considera parte de la sintaxis y eso nos permitirá leer el código con más claridad [17].

Para ver cómo se construye una tupla observa el siguiente ejemplo:

```
tupla = ("x", "y", "z")
```

Para construir una tupla con un solo valor es necesario colocar una coma al final, esto para que Python intérprete que se trata de una tupla, en caso contrario se estaría declarando una cadena.

2.11. Diccionarios

Entre los diversos tipos de estructuras de datos se encuentra el diccionario de datos, pero analicemos que es un diccionario como tal en la vida cotidiana, un diccionario es una herramienta donde se consultan palabras, términos o diferentes tipos de información. Ahora bien este propósito y definición de un diccionario lo llevamos hasta el lenguaje de programación en Python.

Un diccionario es considerado un contenedor asociado a una clave, en otras palabras es una estructura y un tipo de dato que tiene características especiales que nos permite guardar o almacenar cualquier tipo de valor como cadenas, listas, números e incluso funciones, permitiendo recordar ese elemento con una clave. Un diccionario lo podemos imaginar como una agenda telefónica. El acceso a los elementos guardados dentro de un diccionario es muy sencillo, e incluso es mucho más rápido que si se busca en una lista [4, 7].

La manera en la que se agregan nuevos elementos es la siguiente:

```
Ejemplo_Diccionario{Luis}= "0101200569"
```

Analicemos este ejemplo:

- Para declarar un diccionario se utiliza llaves {}, entre las llaves se encuentran los valores clave, pueden ser varios y se separan con comas
- El acceso al elemento es sencillo y rápido
- Existe una clave para que hacer referencia con el elemento a guardar, con esta clave se pueden acceder al elemento guardado.

- Cuando existe una clave su valor se actualiza en el registro existente.
- Si no existe una clave, entonces se crea un registro nuevo.
- En un diccionario no es necesario tener un orden específico, el orden de almacenamiento lo hace el intérprete de Python [4].

La flexibilidad de un diccionario hace que Python internamente representa el diccionario como una tabla ***hash***. Las tablas ***hash*** son un tipo de función que busca en un bloque de datos, este bloque puede ser de longitud variable o fijo. Las tablas permiten que facilitar el acceso y aumenta dinámica mente el número de los elementos.

2.11.1. Añadir, insertar y borrar en un diccionario

Anteriormente se indico como se declara un diccionario y como se estructura un diccionario, ahora vamos a mostrar cómo añadir elementos al diccionario.

Primero creamos un diccionario con valores:

```
dicc = {"x" : 2, "y" : 4, "z" : 6}
```

Teniendo un diccionario de datos con valores vamos a obtener el valor de un elemento por la clave, para esto se utiliza la siguiente sentencia: [7]

```
dicc{"x"}
2
```

Vamos a modificar el valor del elemento "x", ¿como hacemos eso?, muy sencillo tenemos que acceder por la clave:

```
dicc{x} = 10
```

Añadir un nuevo elemento al diccionario es igual de simple como modificar un valor a un elemento, la manera correcta de añadir es colocando el elemento clave entre las llaves y su valor después del signo igual, y veamos como esto de añadir un nuevo elemento:

```
dicc {a} = 2
```

Existen tres métodos que permiten realizar varias veces sobre el diccionario, en lenguajes de programación se llama iterar, los métodos son: ***items(), values() y keys()*** [7].

- El método ***items()*** permite el acceso a tanto a claves como a los valores
- El método ***values()*** se encarga de devolver los valores.
- El método ***keys()*** devuelve las claves del diccionario.

Cuando se está iterando sobre un diccionario es necesario hacer uso de un bucle **for**, esto permite que el tercer método ya no sea llamado cuando existe un bucle.

En el siguiente ejemplo se muestra un uso de método ***keys()*** junto con otra función ***list()***, esto nos devolverá el resultado con todas la claves del diccionario.

La función ***list()***, muestra una lista [7].

```
list(dicc.keys())
```

De la misma manera es posible hacer uso del método ***values()*** junto con la función ***list()***, esto nos dará como resultado los valores que tiene el diccionario. Veamos un ejemplo de este tipo, en el cual el resultado será una lista con las claves de cada uno de los elementos:

```
list(dicc.items())
# mi resultado es:
[(a = 2, x = 10, y = 4, z = 6)]
```

Existe una función integrada para eliminar un valor del diccionario, esta función se llama ***del()***. Esta función opera de la siguiente manera, la función ***del()***, necesita un elemento para eliminar el valor del diccionario, observa el ejemplo:

```
del(dicc[y])
```

Otro método que sirve para eliminar elementos del diccionario, es el método ***pop()***, su funcionamiento es similar al que se explicó en la sección de listas. Otra función que puede ser utilizada en el diccionario es la de ***len()***, recordemos que esta función mide la longitud para el caso de las cadenas, para el caso del diccionario devuelve el número de elementos contenidos en el diccionario.

Es posible comprobar si existe una clave dentro de los elementos del diccionario, para esto se emplea el operador in, que permite indicar si existe un valor dentro del elemento, para esto el resultado mostrado por el operador es "True" para el caso de existir un valor y "False" para el caso de no exista. El ejemplo es el siguiente [7]:

```
"a" in dicc
#el resultado es:
True
```

2.11.2. Comprensión y ordenación de diccionarios en Python

La comprensión es útil para iniciar un diccionario con un determinado valor, y tomando como referencia las diferentes claves que existen en el diccionario. Esto funciona de forma similar en las listas, se puede crear un diccionario con la iteración de una lista, observa el siguiente ejemplo, donde

se crea un diccionario con la iteración de una lista:

```
{a: a+1 for a in (1, 2, 3)}
{1: 2, 3: 4, 4: 5}
# iniciamos el contador con valor de 1
clave_elemento: 1
for clave_elemento in ["a", "b", "c"]
{"x": 1, "y": 1, "z": 1}
```

La ordenación es posible en el diccionario, recordemos que para el caso de las listas el método para ordenar era utilizar el *sort()*, pero para el caso del diccionario no existe este método, para ordenar una lista de las claves de un diccionario se usa la función *sorted()*. Observa el siguiente ejemplo donde se muestra cómo se usa esta función integrada [7].

```
sorted(dicc)
# el resultado es
["x", "y", "z"]
```

2.12. Herramientas para control de flujo en Python

Las herramientas de control de flujo son un conjunto de bloques de código en el cual se pueden agrupar instrucciones de manera que se ejecuten de forma controlada.

Pero antes de adentrarnos el tema de esta sección, se recomienda algo que es muy importante al momento de utilizar estas herramientas de control, como lo es la indentación.

En un lenguaje de programación la indentación es conocida como los espacios dentro de una sentencia, o también conocido el mundo informático como la sangría. Para el caso de Python tener la indentación es muy recomendable y obligatoria, recordemos que Python es un lenguaje que por su estructura es muy fácil de leer, esto es gracias a la indentación dentro del lenguaje, en este caso de define que la indentación es obligatoria para el lenguaje Python. Con la finalidad de proporcionar mayor legibilidad al código ya que también de eso depende su estructura.

La indentación para Python es de cuatro espacios en blanco, esto indica que las instrucciones que se coloquen después de estos espacios pertenecen o son parte de una misma estructura de control.

Otro elemento que se considera al momento de hablar de estructuras de control es el *encoding* o también llamado codificación. El *encoding* es una directiva que se coloca en el inicio del archivo de Python. Se coloca de esta manera para indicar al sistema la codificación que se utilizara en el archivo [4]].

2.12.1. Sentencia if

Los fragmentos de código son más útiles cuando existen las sentencias condicionales, permitiendo que nuestro código que tenga un orden al momento de ejecutarse y al mismo tiempo que la sentencia

condicional permitirá que el programa se comporte de diferentes formas, esto último dependerá del código.

La sentencia **if** es una condicional que permite evaluar la condición. La sintaxis de la condicional **if** es: [17]

```
if condición :
        código para ejecutar la condición
```

- **if** (en Inglés, que significa "sí") es la sentencia condicional
- seguido de la condición, seguido por dos puntos (:)
- Y por último en una nueva línea identada, se coloca el código que se ejecutará en caso de la condición se cumpla.

Ahora realicemos un ejemplo:

Supongamos que en un programas al usuario se le pide insertar datos, ese datos se va a almacenar en una variable llamada "Datos". Ya tengamos el valor insertado por el usuario lo vamos a evaluar con la condición **if**, vamos a colocar la siguiente condición: Si el valor insertado es mayor que 1, mandame un mensaje en pantalla indicando que el dato ingresado es mayor que 1.

Ahora observemos esto en un pequeño fragmento de código.

```
Datos = int(input("Escriba un número: "))
if Datos < 0:
        print("Es un número mayor a 1")
```

En otros lenguajes de programación utilizan las llaves para determinar que existe un flujo de condiciones. En Python esto no es necesario, ya que Python pretende que los programadores aprenda a usar la indentación de manera natural y que esto beneficia al lectura del código.

2.12.2. sentencia if...else

Esta sentencia es un poco más complicada, ya que permite que se ejecuten otras instrucciones, es decir se pueden ejecutar más instrucciones después de haberse cumplido la primera condición, o cuando no se cumple la primera condición [6].

El **else** significa "si no", por lo tanto la estructura de la condición **if...else** queda de la siguiente manera:

```
if condición:
        se colocan las instrucción y se ejecuta en caso de que la
        condición sea cierta. Esta instrucción puede tener varias líneas
else:
        se colocan las instrucciones para que se ejecuten si la
        primera condición  no cumple.
```

Analicemos el ejemplo anterior:

- La primera condición con **if** es la que se evalúa, recordemos que se debe terminar con dos puntos (:).
- Después viene el bloque de instrucciones, esto cuando la condición se cumple o dicho en otras palabras cuando la condición es verdadera. Para que Python reconozca bloques de código para una condición es importante que lleve la sangría o tabulación, recordemos que este espacio está conformado por cuatro espacios.
- Después viene la línea de orden **else**, indicando que el bloque de instrucciones que se encuentra dentro de **else** se ejecutará porque no la condición no se cumple, o dicho en otras palabras cuando la condición sea falsa. **Else** también debe terminar con dos puntos(:), como el caso del **if.** Hay que diferenciar algo muy importante, **else** finaliza con los dos puntos, y el **if** lleva condición y finaliza con dos puntos.
- Para finalizar recordar que el último bloque de instrucciones pertenecen al de **else.**

Ahora realizaremos un ejemplo para mostrar cómo sería la estructura de **if...else** con código

```
numero = int(input("Escriba un número "))
if numero < 5:
        print("El número introducido es menor que 5")
else:
        print("El número introducido es mayor que 5")
```

Otra opción es escribir un programa con dos **if** en lugar de **if... else,** pero está opción no es muy recomendable. Al colocar dos bloques con **if**, Python se ve obligado a evaluar dos condiciones en un mismo bloque de código. Mientras que con **if...else** únicamente se evalúa una sola condición, además de que en en programas muy grandes es mucho más simple usar el bloque **if...else.**

Otra ventaja de usar **if...else**, es que solo se escribe una sola condición para todo el bloque de instrucciones.

2.12.2.1. Alternativa if... elif

Todavía falta una condición que considerar dentro de este apartado y se trata del **elif**, es una contracción de **else if.** La construcción **elif** se lee como "Si no", es decir evalúa la condición del **if**, si esta es verdadera se ejecuta las instrucciones y después continúa ejecutando el código que está antes del condicional, si no se cumple entonces se evalúa la condición **elif.** Cuando se cumple esta última condición se ejecutan las instrucciones del código posterior al condicional; puede existir más de un **elif**, por lo tanto se continua ejecutando en el orden de ejecución. Cuando no se cumplen las condiciones de **if** y **elif**, entonces entra en acción el conjunto de instrucciones que tenga el tenga el bloque de **else** [6].

Esta estructura es una alternativa del control de condiciones, en Python la orden de **elif** se escribe así:

```
if condición1:
        bloque de instrucciones 1
elif condición 2:
        bloque de instrucciones 2
else:
        bloque de instrucciones 3
```

Esto se lee así:

- Si la condición 1 se cumple, entonces ejecuta el bloque de instrucciones 1.
- Si no se cumple la primera condición, pasa a la segunda condición, si se cumple la condición 2, ejecuta el bloque de instrucciones 2.
- Si no se cumplen la condición 1 ni la 2, entonces ejecuta el bloque de instrucciones 3.

Es importante conocer que para este tipo de estructura se pueden escribir tantos **elif** como sean necesarios, pero el bloque **else** solo una vez para evaluar lo que las condiciones anteriores no cumplen.

El orden en el que se escribe estas estructuras es muy importante, esto dependerá de las condiciones de cada programa. Por ejemplo:

1. Cuando los casos a evaluar son mutuamente excluyentes. Ejemplo de estos son obtener valores negativos, evaluar valores entre un rango de datos definidos o evaluar datos a partir de un rango.
2. Cuando unos casos incluyen a otros. Ejemplo de estos son: cuando queremos obtener múltiplos de cualquier serie que se desee (2,4,6 ...etc.), incluso si no es múltiplo.

Si la condición que se declara en el fragmento de **elif** cumple con todas la posibilidades, para este caso no sería necesario declarar un bloque de instrucciones **else** [6].

2.12.3. Bucle while

Las sentencias **while** permite ejecutar un bloque de código con un cierto número de veces, mientras que las condiciones ejecutan instrucciones de código que dependen de sus condiciones. La sentencia **while** significa "mientras", este bucle se ejecuta mientras un bloque de instrucciones cumple con una condición, es decir mientras una condición tenga el valor de "True" [6].

La sintaxis **while** es la siguiente:

```
while condicion:
        cuerpo del bucle
```

En Python cuando se llega a un bucle **while**, se evalúa la condición que tiene el bucle, y si es verdadera se ejecuta lo que contenga dentro de ese bucle, después de ejecutarse se repite el proceso una y otra vez mientras que la condición sea verdadera. Cuando la condición del bucle es falsa, no se ejecuta el bucle y se continúa ejecutando el resto las instrucciones del programa.

En un bucle **while** las variables que se utilizan son variables de control, estas se definen antes del bucle, estas variables se emplean dentro del bucle y se pueden modificar en el bucle.
Veamos un ejemplo de bucle while, este ejemplo se retoma de la referencia [6]:

```
edad = 0
while edad < 18:
        edad = edad + 1
        print "Felicidades, tienes "+ str(edad)
```

Analicemos este ejemplo:

- Se declara una variable edad y tiene el valor de 0.
- En la siguiente línea está el bucle **while** en el cual se tiene la condición de que mientras la edad sea menor que 18.
- En la siguiente línea se ejecuta lo que la condición indica, es se va aumentar 1 a la variable edad y se imprime el mensaje de la siguiente línea.
- En la línea de **print** se utiliza un operador + y una función ***str***, esto para crear una cadena, esto para crear una concatenación de un número y una cadena.
- Hecho lo anterior, el bucle se ejecuta en 1 y se evalúa con el valor de la condición 18, esto se vuelve a ejecutar y se va aumentando el valor en la variable edad, y esto se seguirá ejecutando hasta llegar a ser igual a 18. En ese momento se dejará de ejecutar el bucle y saldrá de ahí para continuar con el resto de las instrucciones del programa.

2.12.3.1. Bucles infinitos

Existen los bucles infinitos, esto puede pasar si la condición no se declara con los parámetro o instrucciones correctas. Pero existe la posibilidad que un bucle infinito sea útil, por ejemplo en programas en los que se requiere que se escriba una palabra clave para finalizar este bucle. En casos como eso es útil un bucle infinito.

Los bucles no intencionados pueden poner al programa fuera de control y deben evitarse. Cuando esto sucede hay una manera de romper con ese bucle, pulsando la combinación de teclas Ctrl+C.

Cuando se está aprendiendo a programar este tipo de errores en los bucles es muy normal. Se recomienda que cuando se expresa una condición siempre hay que igual y comparar los valores [10].

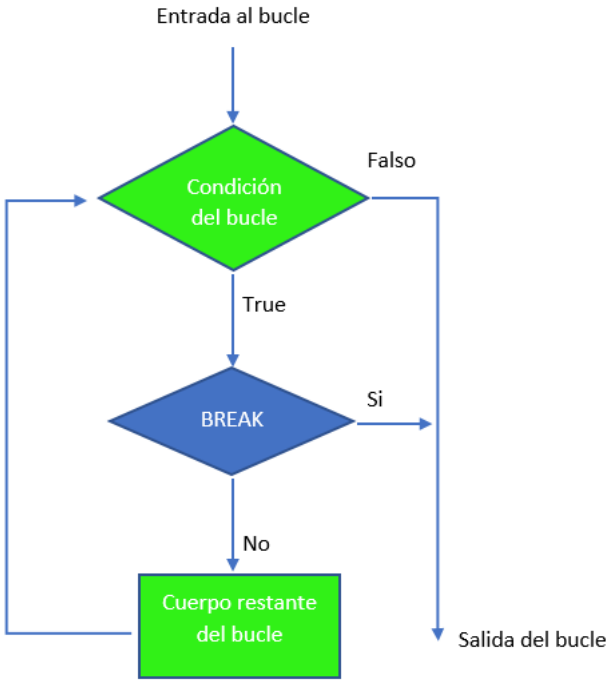
2.12.3.2. Bucles con sentencias break y continue

De manera alternativa para los bucles existe una alternativa para interrumpir un bucle, se trata de la palabra reservada **break**, permite romper un bucle aunque la evaluación de la condición de while siga siendo verdadera [4].

La sintaxis de un programa con sentencia **break** se muestra con un diagrama de flujo, observa la siguiente Figura 2.4, donde podemos observar como funciona una sentencia **break** dentro de un bucle **while.**

```
while condición:
        bloque de instrucciones para la condición
        if condición2
                break
        bloque de instrucciones del while
bloque de instrucciones fuera del bucle while
```

Figura 2.4: Diagrama de flujo de una sentencia break
Elaboración propia



La sentencia **continue** permite que pase al principio del bucle sin importar e ignorando si ha terminado de ejecutarse en ciclo actual del bucle, pasando a la siguiente iteración del ciclo [4]. El

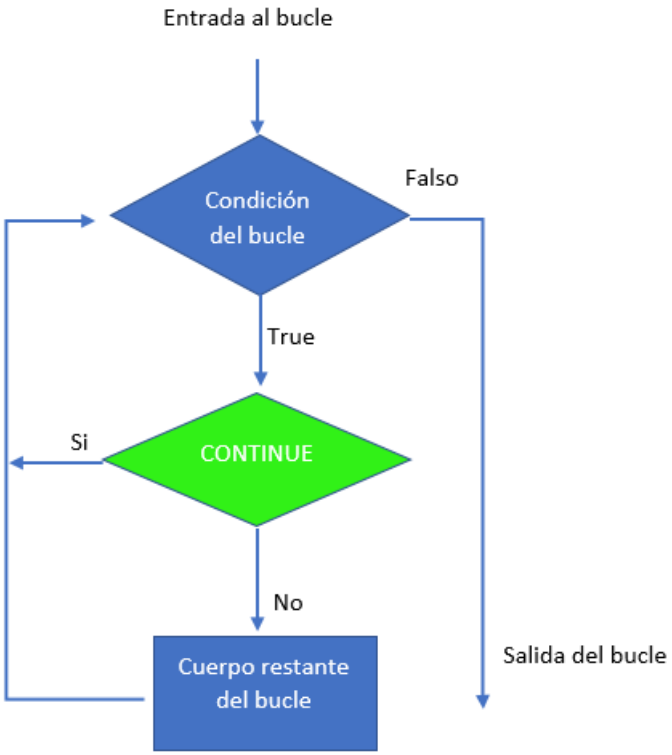
continue se puede usar al igual que en el caso del **break** en los bucles de **while** y **for** (este último lo veremos más adelante).

La sintaxis para el continue es el siguiente:

```
while condición:
        bloque de instrcuciones de la condición
        if condición2:
                continue
        instrucciones dentro del while
instrucciones fuera del while
```

En la Figura 2.5 se muestra el diagrama de flujo de **continue.**

Figura 2.5: Diagrama de flujo de continue



2.12.4. Sentencia for

Para el bucle **for** existe un tipo de bucle llamado **for-in** y se lee de la siguiente manera " Para todos los elementos de una serie, hacer.." [10].

Por lo general una sentencia **for** es denominada como bucle **for**, que es una estructura de control.

La sentencia **for** se ocupa para repetir varias veces una determinada rutina hasta llevar a cumplir la condición, se comporta de una manera más distinguida que el **while** [10].

Un bucle **for** está diseñado y pensado para recorrer listas o conjuntos, un conjunto recordemos que es una colección de valores, separados por comas y se encuentran delimitadas por llaves [21].

La sintaxis del **for** es la siguiente:

```
for variable in secuencia del bucle (lista, cadena, range, etc.):
        instrucciones del bucle
```

- La variable no es necesario definir la dentro del bucle, puede utilizarse una variable ya definida dentro del programa. Esta variable será la que recorra al bucle.
- AL igual que el **while**, el bucle **for** al final de su condición se coloca dos puntos (:).
- Las instrucciones se repiten y para cada repetición de las instrucciones se les llama iteración
- El bucle se puede ejecutar tantas veces como el elemento recorrible sea necesario.

Observemos un ejemplo del bucle **for** retomado de la referencia [21]:

```
print("Comienzo")
for i in [0,1,2]:
        print("Hola", end="")
print()
print("Final")
```

En este ejemplo la el **for** realizará un recorrido tres veces por el bucle, de manera que la cadena "Hola" se ejecutará tres veces.

Puede darse el caso que la lista o los elementos dentro del bucle no se ejecute por que no existe valores

De manera opcional el bucle **for** acepta la sentencia **else**. Su funcionamiento de la sentencia **else** en un bucle **for** es: si aparece todas las sentencias que siguen serán ejecutadas si es que no se encuentra una que provoque la salida del bucle. Para salir de un bucle, al igual que **while**, en un bucle **for** existe el **break** y **continue**. El **break** en **for** es una técnica para salir del bucle de manera inmediata. El **continue** provoca un salto a la siguiente instrucción del bucle [7].

Una variable de control es empleada antes del bucle, cuando tiene algún valor, esta variable no afecta la ejecución del bucle, al terminar el bucle, la variable de control almacena el último valor que fue asignado.

En los bucles **for** existe la posibilidad de tener dos o más bucles seguidos, utilizando la misma variable, de manera que cada uno de los bucles determina el valor de la variable, esto significa que no importa que valor tenga antes de entrar al bucle.

Algo muy importante que se usan en los bucles son los operadores en las condiciones, esto para saber qué operación se realizará en la condición. ¿Porque se utilizan los operadores en los bucles? porque siempre surge la necesidad de recorrer cierto número de veces y de manera paralela realizar la operación de un cierto número de veces hasta que la variable tenga el valor determinado.

2.12.4.1. Contadores y acumuladores en bucle for

Los contadores en muchos lenguajes de programación se emplean para hacer un conteo de las veces que ha ocurrido una condición determinado valores. Para el caso de Python ocurre lo mismo, los contadores hace la misma función, contar el número de veces que se ejecuta una condición.

Los acumuladores son los valores que se almacenan después de que se cumple cada condición. Los acumuladores se van modificando en cada iteración que se realiza en el bucle, los acumuladores por lo general siempre se declara con un valor inicial [10].

2.12.5. Bucles anidados

Un bucle anidado es cuando un bucle se encuentra dentro de otro bucle. El bucle que se encuentra dentro de otro se le llama bucle interno, y el que se encuentra afuera se le llama bucle externo. Los bucles anidados pueden contener cualquier nivel de anidamiento, es decir pueden existir más de dos bucles dentro de un externo.

En Python cuando existen estos bucles se recomienda que las variables tengan u nombre distinto en cada bucle para evitar que las variables se se repitan y se interpreten en más de un bucle.

La sintaxis de un bucle anidado en la siguiente:

```
for variable1 in(números, cadenas, etc.):
        for variable2 in (números, cadenas, etc.):
        se puede colocar una instrucción para imprimir
```

Ahora se explicará como funciona un bloque anidado

- Primero se ejecuta el bucle externo, se toma el primer valor de la variable1.
- Entra en el bucle interno y toma la variable de control juntos con el primer valor que encuentra.
- Después sale del bucle para ejecutar la siguiente iteración, la cual puede ser un print para visualizar los valores que ya se tomaron en cada bucle.
- Después de vuelve a ejecutar dentro del bucle interno, y remota la variable de control y toma el siguiente valor que se encuentra dentro de los paréntesis.
- El bucle terminará cuando termine de recorrer todos los parámetros dentro del paréntesis de cada bucle.

Cuando existen bucles anidados es importante prestar atención a la indentación de las instrucciones, ya que para Python esto es vital para poder identificar un bloque de instrucciones.

En los bucles cuando de utilizan listas existe el tipo *range()* para facilitar el funcionamiento del bucle, también tiene la ventaja de que durante la ejecución consume menos memoria y mejorar el desarrollo del programa.

Capítulo 3

¿Que es Qt?

En la actualidad la tecnología y las innovaciones del software hacen posible que se puedan crear distintos sistemas y adaptarlos a nuestro gusto y necesidades. Qt es una tecnología es usada en programas que utilizan interfaz gráfica, además tiene la ventaja que fue desarrollada usando software libre y de código abierto, esto quiere decir que participan en el desarrollo del proyecto de Qt la comunidad y las personas que se dedican al desarrollo de sistemas, en este caso hablaremos de Nokia y Digia. Nokia tuvo gran éxito con la fabricación de teléfonos móviles en el año de 1998 hasta el 2011, después de estos años su éxito se fue acabando cuando entraron al mercado los teléfonos inteligentes. Por otro lado Digia es una empresa de software de origen finlandés, que adquirió la tecnología de software de Qt de Nokia. Digia se hace cargo de las actividades que Qt realizaba con Nokia como son el desarrollo de productos, las licencias abiertas, el negocio comercial y los servicios.

Gracias a las licencias comerciales que Digia se encargó de trabajar, ahora Qt está bajo términos de GNU, además de seguir implementado que Qt se incorpore por completo iOS y Android, ofreciendo el servicio de soporte tal como ya existe en Windows, Mac OSX y Linux.

Seguramente para muchas personas se le hace desconocido el termino de Qt, pero tal vez muchos saben lo que es, ya que en diversos programas lo usan, sobre todo los multiplataformas. Qt trata de una serie de librerías para entornos gráficos desarrolladas por Trolltech, (anteriormente conocida como Quasar Technologies) es una empresa de software de computadoras de Olso, Noruega. Fue fundada por Eirik Chambe-Eng y Haavard Nord en el año de 1994, ellos empezaron a escribir lo que ahora es Qt en 1991, el desarrollo fue basado en código abierto, aunque no del todo era código abierto ya que no permitía ciertas acciones [22].

Figura 3.1: Logo Qt

Trolltech empezó a desarrollar Qt con licencias GLP desde el año 2000, esto para los sistemas libres y de pago para los sistemas privados, en la última década Qt ha pasado de ser un producto usado por pocos a convertirse en un producto utilizado por muchos desarrolladores open source, ya que esta tecnología presenta un futuro prometedor para la tecnología de desarrollo [5].

En la versión 1.45, Qt permanecía con código Licencia Qt Free Edition, cuyo código fue publicado pero no tenía la compatibilidad del código abierto y mucho menos la definición de software libre. Este detalle tuvo gran controversia, después de esto se lanza la versión 2.0 donde la licencia de Qt fue cambiada a una licencia de Software libre, aceptada por la Free Software Foundation la cual quedo como Q Public Licence (QPL).

El sistema de compilación de Qt contiene su propio Qmake, es una herramienta que permite la optimización de proyectos que utilizan makefiles. Qmake es una interfaz que permite la compilación en los sistemas nativos como GNU Make, Visual Studio y Xcode.

Qt por sus siglas Leido Quiut, es una bibliotecas para multiplataforma, que quiere decir esto, Qt se caracteriza por ser multiplataforma y se encuentra disponible para los siguientes sistemas operativos: GNU/Linux, BSD, MacOsX, Windows, Windows Phone, iOS, Android.

Hay tres ediciones de Qt para cada una de las plataformas

- GUI Framework: edición de nivel reducido de GUI, orientado a redes y base de datos
- GUI Framework: edición completa open Source
- GUI Framework: edición completa comercial [25].

Qt cuenta con su entorno de desarrollo integrado, o también denominado IDE. Este IDE funciona en plataformas como Windows, Linux y OS en ellos se puede tener código inteligente, editor de texto con resaltado de sintaxis, sistema de ayuda, programa depurador y analizador dinámico de programas, este último para medir el espacio de la memoria; todo esto incluye los complementos de Visual Studio de Qt.Qt desarrolla aplicaciones para diferentes requisitos de mercado, esto convierte a Qt en

un software internacional ya que se puede diseñar y adaptarse a varios idiomas, un ejemplo de estas aplicaciones son la localización esto significa que puede estar en cualquier región y tener un idioma específico [24].

Qt se ha convertido en mucho más que una GUI, ya que posee una lista de herramientas que proporcionan módulos para el desarrollo en las múltiples plataformas que se encargan del áreas de base datos, redes, tecnologías web, protocolos para la transmisión de comunicaciones como bluetooth, impresiones en PDF, procesamiento de archivos en formato XML, entre otros [24].

3.1. Qt y sus herramientas principales

Qt tiene una gran cantidad de herramientas, ejemplo de ellas son las siguientes, que más adelante se describen y se ilustran:

Qt tiene una gran cantidad de herramientas, ejemplo:

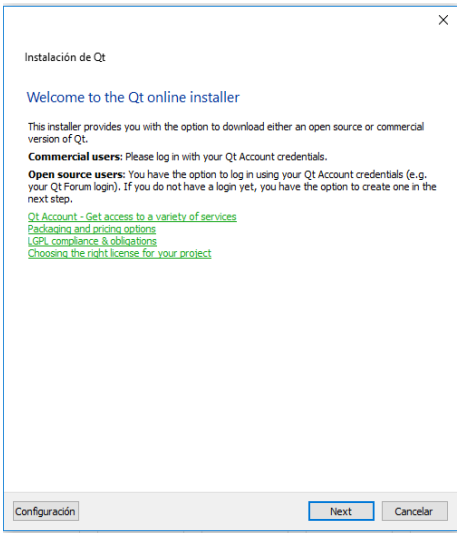
- Qt Designer: para crear diálogos e interfaces gráficas.
- Qt Linguist: herramienta utilizada para desarrollar aplicaciones de multilinguistica.
- Qt Assistant: documentación oficial.
- Qmake: un generador makefiel multiplataforma.
- Qt Creator: IDE Multiplataforma.

3.2. Instalación de Qt

Para instalar Qt se descarga desde el siguiente link https://www.qt.io/download , dentro de la página de Qt hacer clic en el icono Download, para empezar a descargar. Al finalizar la descarga ejecuta el archivo y se acepta los permisos para ejecutar el archivo.

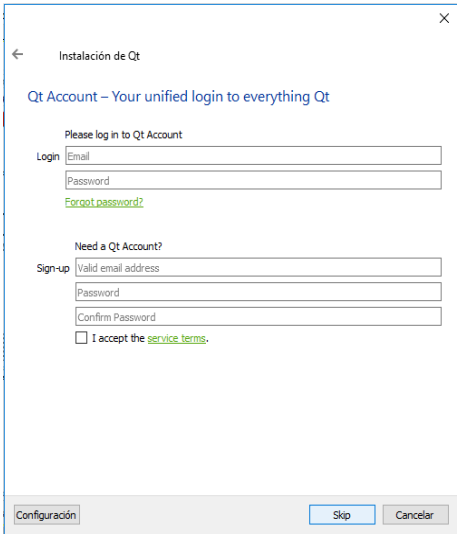
En la Figura 3.2 se muestra el asistente de instalación de Qt, clic en Next.

Figura 3.2: Asistente de instalación de Qt



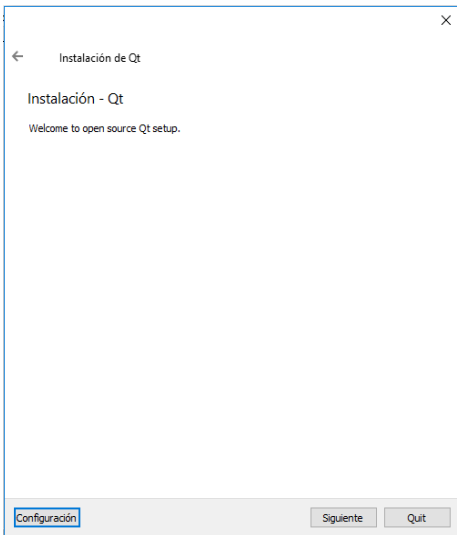
En la siguiente Figura 3.3 se da clic en Skip. Esta ventana nos pide un registro, pero esta la opción de omitir este paso.

Figura 3.3: Instalación Qt



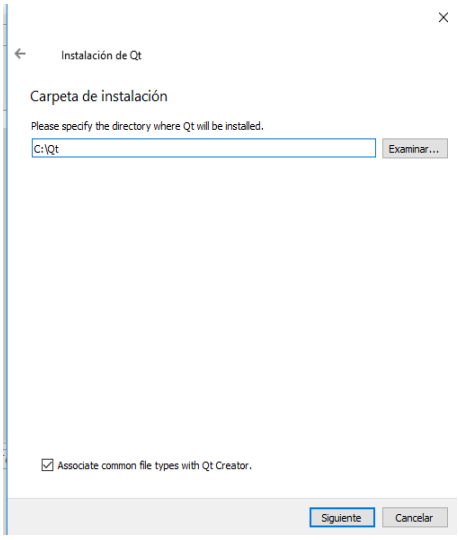
En la siguiente Figura 3.4 dar clic en Siguiente, esta es la ventana de bienvenida a la instalación.

Figura 3.4: Ventana de bienvenida a la instalación



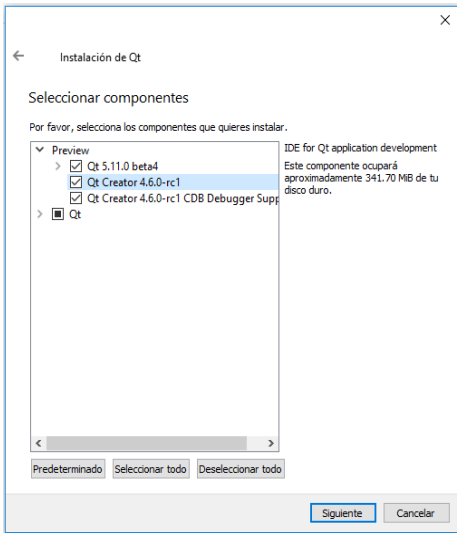
En la siguiente Figura 3.5 dar clic en Siguiente, esta es la ventana donde se indica donde se estará la ruta de instalación de Qt.

Figura 3.5: Ruta de instalación de Qt



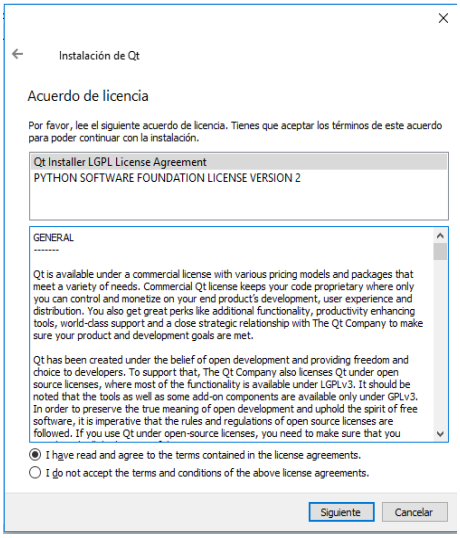
Se seleccionan los componentes que se requieren para la instalación de Qt (ver Figura 3.6), entre ellos se selecciona el elemento Qt, que a s vez tiene una sub lista que es la que vamos a elegir, es importante seleccionar los elementos necesarios por que de ellos dependen los componentes para otras aplicaciones. Después clic en Siguiente.

Figura 3.6: Componentes Qt



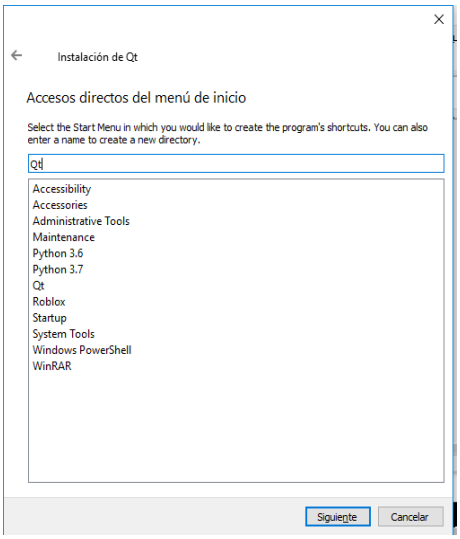
En la Figura 3.7 se muestran los acuerdos de la licencia, clic en Aceptar los términos de la licencia y después clic en Siguiente.

Figura 3.7: Acuerdos de licencia



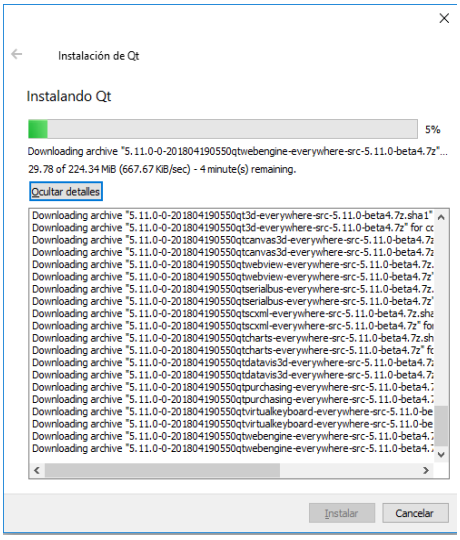
La siguiente Figura 3.8 se muestra donde se coloca el acceso para el directorio. Después clic en Siguiente.

Figura 3.8: Acceso para menú



En la Figura 3.9 se muestra el progreso de la instalación, cuando esto termine, clic en Finalizar y eso se concluye en proceso de instalación de Qt.

Figura 3.9: Progreso de la instalación

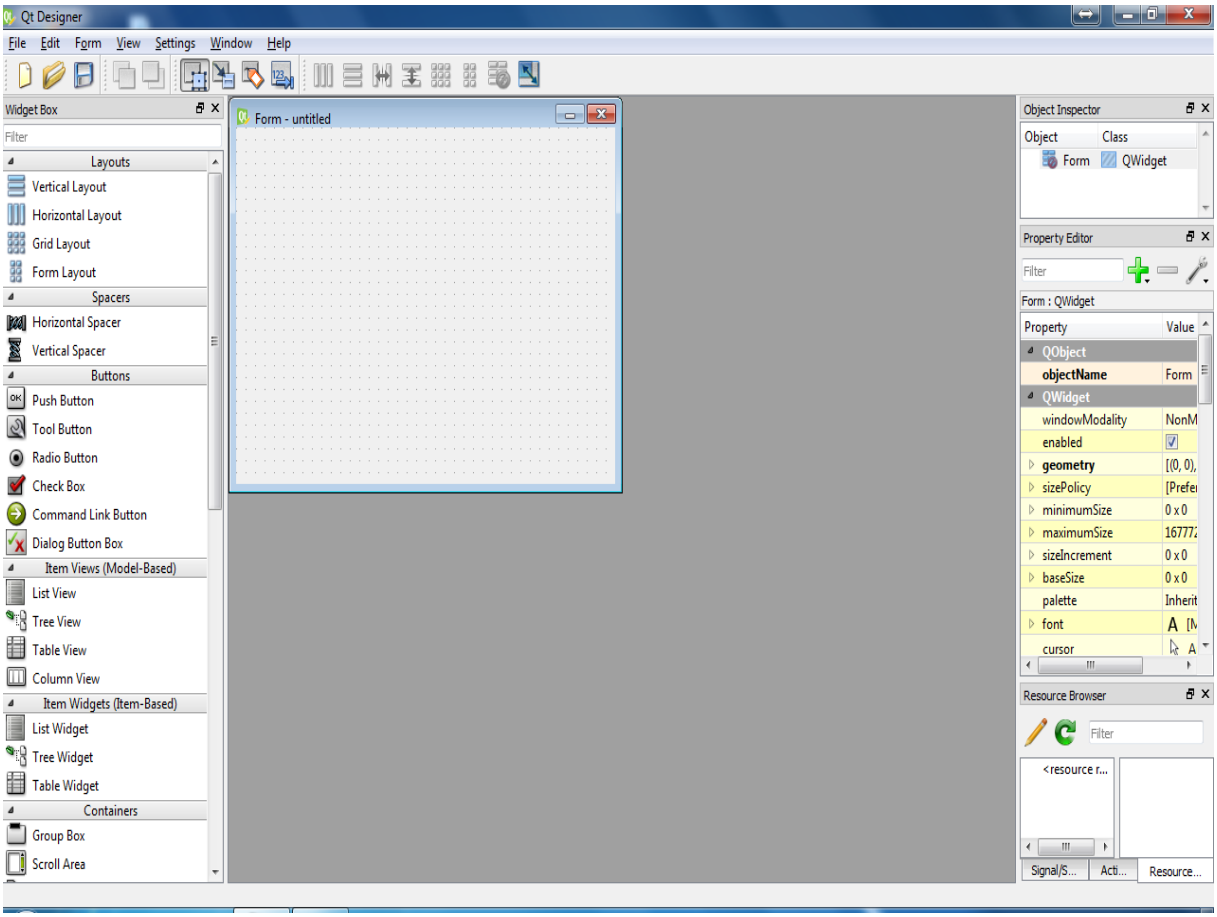


3.3. Qt Designer

Es una aplicación para crear las interfaces de usuario utilizando las librerías de Qt [20]. Qt es una herramienta de diseño para las aplicaciones de Qt. Este editor le ayudara a editar y diseñar

facilmente su GUI con tan solo arrastrar botones, cajas de texto, etiquetas etc., que se encuentran el los widgets, así como también cambiar las propiedades de estos elementos, en la Fig. 3.10 se muestra la pantalla principal y los elementos que contiene el Qt Designer.

Figura 3.10: Qt Designer



3.3.1. Qt Linguist

Qt Linguist es una guía que ofrece el soporte para los idiomas que ayuda a Qt en la traducción. Esta guía el programador se encarga de ingresar en su idioma nativo las palabras de su código fuente para la traducción, esto debe de contener la sintaxis para identificar qué palabras se tienen que traducir.

Cuando se genera la traducción, en Qt Linguist se guardan las traducciones ingresadas, permitiendo enriquecer los archivos de traducción y se prepará para futuras consultas. Las traducciones pueden tener ciertas variantes esto por la complejidad de los lenguajes humanos, se puede traducir dependiendo el contexto, dependiendo el idioma del teclado e incluso hasta las fases que contiene

variables, pueden contener cuestiones complicadas.

Los archivos de traducción son textos visibles para los usuarios y la manera en la que se crean es la siguiente:

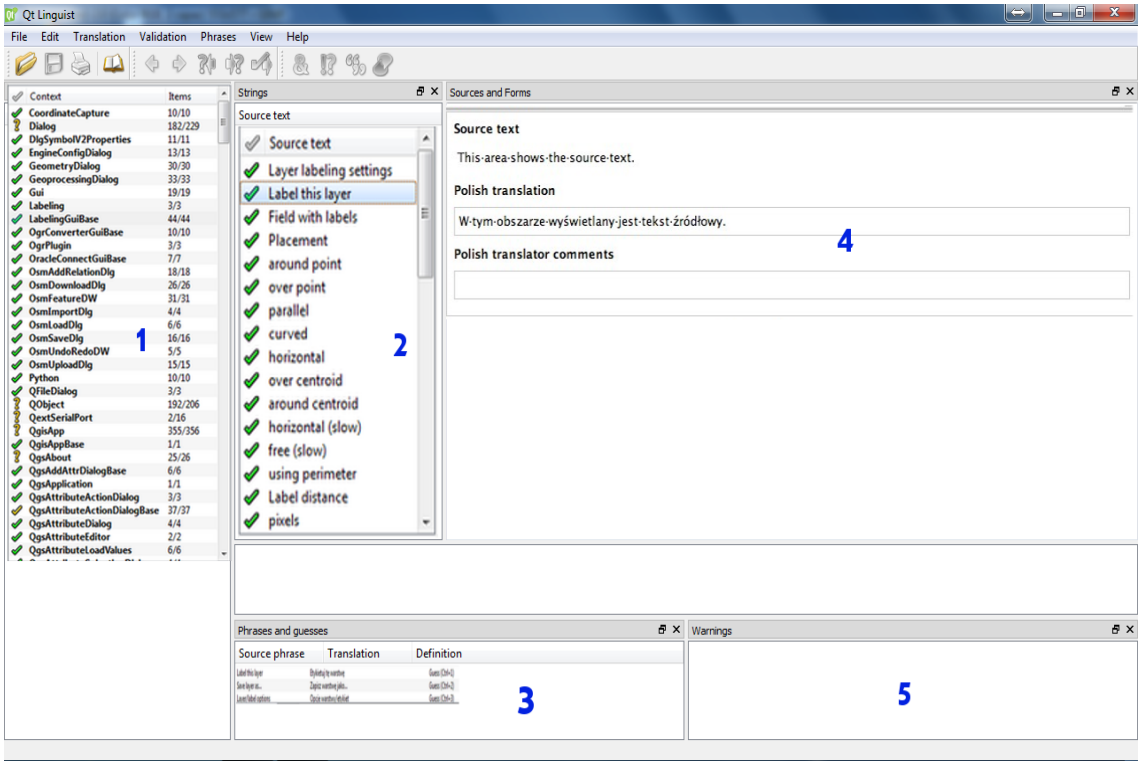
Se ejecuta un *lupdate* para generar el archivo fuente de traducción, este contiene texto visible para los usuarios, cabe mencionar que este texto aún no contiene la traducción.

Después el archivo generado se le pasa al traductor para que con Qt Linguist hagan la traducción y este ocupa cualquier texto que se haya ocupado para traducir. En caso de que se requiera una nueva incorporación de un nuevo texto se ejecuta otra vez el *lupdate*.

Qt Linguist tiene la capacidad de modificar las cadenas de traducción para la validación. las fallas que se pueden generar en la validación de pueden mostrar en la ventana de advertencias.

La ventana de Qt Linguist está dividida en sub ventas como son: Contexto, Fuente, Formulario, Cadena, Frases y conjeturas y Advertencias (en inglés llamadas Sources Text, Source code, Phrases and guesses y Warning, respectivamente.) Estas ventanas siempre pueden estar visibles, aunque también se pueden desactivar [15].

Figura 3.11: Qt Linguist



3.3.1.1. Ventana de contexto

La ventana de contexto que aparece con el número 1, enumera las cadenas para ser traducidas, el orden de la traducción es por orden alfabético, cada contexto pertenece a una subclase QObject

o también puede existir un contexto sin nombre. En la misma sub venta de contexto aparece una etiqueta en forma de ✔ esta puede ser de color verde o amarillo, si la etiqueta es de color verde ✔ significa que las cadenas de contexto han sido traducidas y pasaron la prueba de traducción. Si la etiqueta es de color amarillo ✔significa que las cadenas en el texto se han traducido pero algunas no pasaron las pruebas de validación. También existe ✔ en color gris, que significa que las cadenas traducidas todavía aparecen y por lo general ya no existen o ya no se pueden traducir. La etiqueta con un signo de interrogación (?), significa que al menos una de las cadenas no se podido traducir.

3.3.1.2. Ventana de cadenas

Esta ventana se tiene el número 2, en esta sub ventana se enumeran todas las cadenas que se han traducido en el texto que actualmente se está traduciendo. En el texto fuente hay dos widgets para ingresar la traducción de la cadena y el otro sirve para quien va a traducir ingrese comentarios.

A la izquierda del texto fuente se encuentra las columnas de Estado, Icono y Descripción, en las cuales se describen las etiquetas de color verde, amarillo y gris:

- Cuando la etiqueta es de color verde ✔ significa que la traducción de la cadena ha sido traducida y el usuario acepto la traducción.
- Cuando la etiqueta es de color ✔, significa que se acepta la traducción pero no pasa la pruebas de validación.
- Cuando la etiqueta es de color ✔, significa que la es obsoleta, ya no se puede usar.

También existe la etiqueta en forma de interrogación (?) de color verde y gris.

- Cuando es verde quiere decir, que el usuario rechaza la traducción de la cadena
- Cuando es gris significa que la cadena no tiene traducción.

Existe también otra etiqueta representada con un signo de admiración cerrado (!), este significa que la cadena no puede ser traducida por qué no hay traducción, pero al mismo tiempo esa cadena no pasa las pruebas de validación

La venta de traducción se localiza en medio de la ventana principal, es importante mencionar que cuando el programador agrega comentarios a su texto fuente estos aparecerán en la parte del área de texto donde se colocan los comentarios del desarrollador. En este mismo apartado existen dos widgets que permiten que se ingresen datos y otro para ingresar la traducción de la cadena, la ventaja de tener esto es para que en futuras traducciones se pueda leer.

3.3.1.3. Ventanas frases y conjeturas

En Qt Linguist es posible cargar libros de frases como referencias para las traducciones, funcionando como herramientas de apoyo que contiene frases fuente, frases objetivo o traducidas y definiciones opcionales. Los libros de frases son herramientas de apoyo para las traducciones y ayudan a que la coherencia se mejor. Estos libros tiene la ventaja de que sirven para evitar que se dupliquen las traducciones.

En la ventana de frases y conjeturas esta con el número 3 en la imagen 3.11, que sirve para evaluar las cadenas con uno o más de los libros de frases. Para poder utilizar las frase y conjeturas se puede hacer doble clic en la traducción y esta se copiará, para mover la venta se puede utilizar la tecla de función F10, las teclas navegación serán útiles para permitir que se pueda moverse entre la traducción. Y la tecla enter será la que nos permita copiar en el área de traducción.

3.3.1.4. Ventana fuente y formularios

Se encuentra del lado derecho de la pantalla principal, ubicada con el número 4. Esta ventana muestra las líneas de código fuente, las cuales contienen las cadenas actuales, si no existiera la cadena de origen entonces se visualiza la ruta indicando que el archivo que espera. Otra opción si el contexto muestra un origen incorrecto, esto significa que es probable que el archivo de sincronización no este entre los archivos de origen. Recordemos que para actualizar los archivos es necesario realizar un lupdate, esto solucionará los errores de origen de los archivos de sincronización.

3.3.1.5. Ventana de advertencias

Esta esta ubicada del lado derecho de la venta de frases y conjeturas y tiene el número 4 en la Figura 3.11. Esta venta de advertencias (o warnings) muestra la traducción actual en caso de que exista algún error con la pruebas de validación. Recordemos que las fallas en la traducción se pueden dar desde antes, pero para el caso de los warnings solo informan el resultado de las pruebas de validación, estas dependen de las pruebas que actualmente se encuentran activas.

3.3.1.6. Tareas comunes

Qt Linguist posee una gran capacidad para cargar y modificar múltiples archivos para la traducción de manera simultánea, es decir se puede traducir en dos idiomas distintos. Cuando se requiere dejar de traducir o abortar una traducción únicamente se presiona las teclas Ctrl + L.

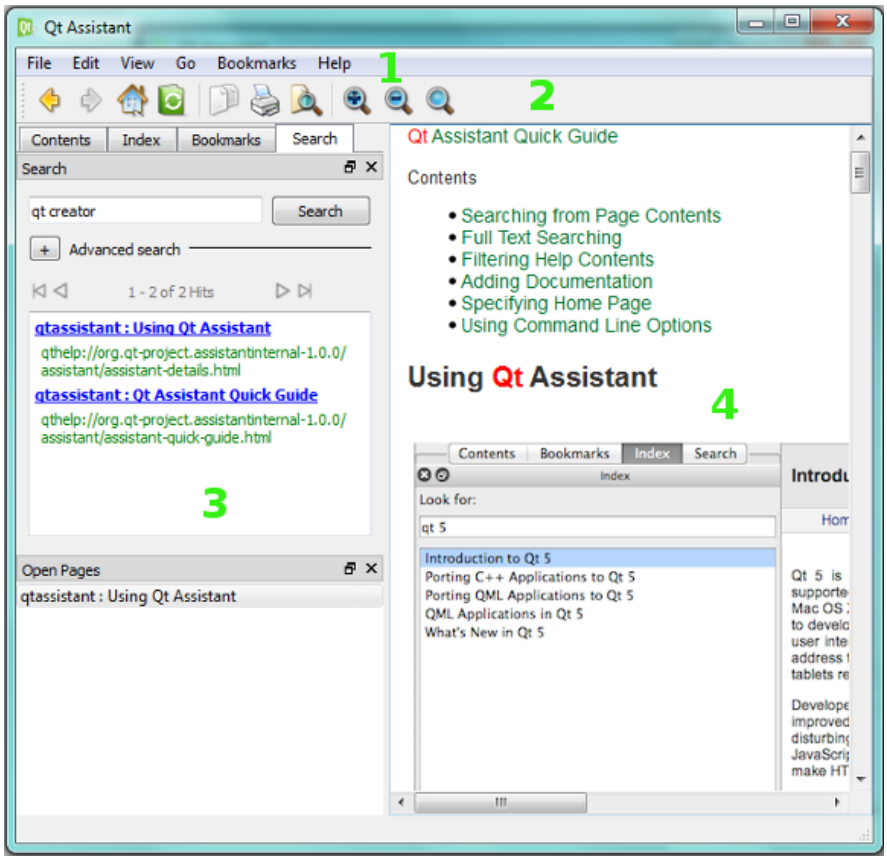
Hay frases en las que se requieren muchas traducciones y en estos casos Qt Linguist puede conocer que esa frase ya fue traducida y puede regresar a una posición anterior para intentar traducir nuevamente elegir entre alguna de las palabras candidatas posibles para la traducción.

3.3.2. Qt Assistant

Qt Assistant es un documento en línea que sirve de ayuda para las herramientas de Qt,cuando se instala qt se instalan una serie de herramientas y documentación del IDE de Qt Creator

En la Figura 3.12 se muestra las la venta principal de Qt Assistant:

Figura 3.12: Qt Assistant



Esta ventana principal está conformada por 4 partes principales que permiten la búsqueda y muestran el contenido de un documento seleccionado. La partes principales se en listan a continuación

1. Menú donde podemos encontrar File,Edit, View, Go Bookmarks y Help

2. Barra de herramientas, contiene los botones para permitir la exploración dentro de los temas consultados. En esta barra se encuentran los iconos de adelante, atrás, home, sincronizar con la tabla de contenidos, copiar, imprimir, buscar y zoom. Cuando se realiza una búsqueda se van resaltando las letras de la búsqueda.

3. Barra lateral o barra de navegación, permite ver los documentos en forma de lista, también contiene un administrador de marcadores y búsqueda. El administrador de marcadores se pueden guardar y este se agrega a la página principal, los marcadores se les puede cambiar el nombre o eliminar y estos dejan de aparecer en el menú contextual.

4. Visualizar documentos, muestra el documento que se buscá [15].

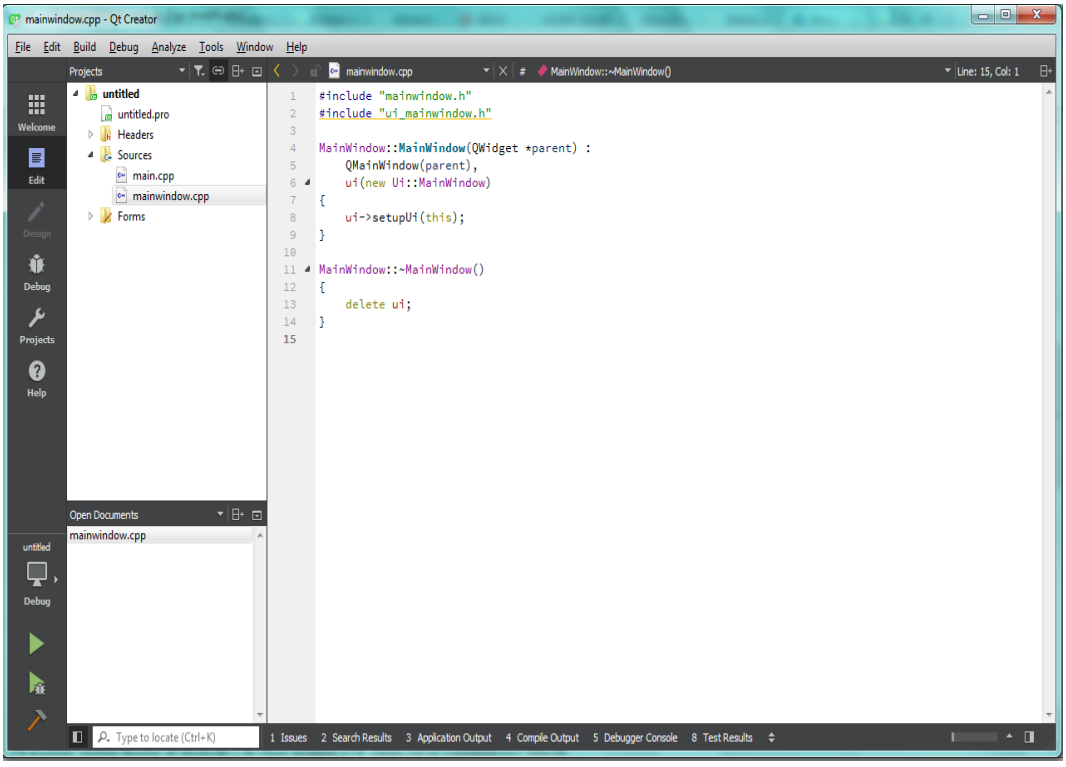
3.3.3. Qt Creator

Qt Creator es un IDE multiplataforma que permite el desarrollo de proyectos , ya que su herramienta de desarrollo y depuración cumple con las necesidades de los programadores. Qt Creator es un entorno de desarrollo de software integrado, que significa que puede soportar el desarrollo tradicional de C++ junto con las aplicaciones, así como también el desarrollo de las bibliotecas de Qt. También esta bajo licencia GPL en su versión v3 y LGPL v2. El desarrollo de Qt Creator empezó en el año 1991, durante diez años unicamente se trato de una caja herramientas para ventanas X11 y en el 2001 se empezó a dar soporte para Mac OS X [24].

Actualmente el proyecto de Qt tiene licencias tanto comerciales como no comerciales, una de las plataformas mas populares para el desarrollo de las aplicaciones es Linux, en muchas de sus distribuciones de Linux se puede obtener Qt Creator utilizando el gestor de paquetes o instalando desde linea de comando [18].

Qt se ha convertido en una herramienta de desarrollo de aplicaciones e interfaces multiplataforma. Qt integra dentro de sus paquetes, herramientas de desarrollo y de soporte tales como Qt Designer y Qt Assistant respectivamente, también librerías de desarrollo y de clases auxiliares a Qt, cuenta con un compilador Gcc MinGW.

Figura 3.13: Ventana principal de Qt Creator



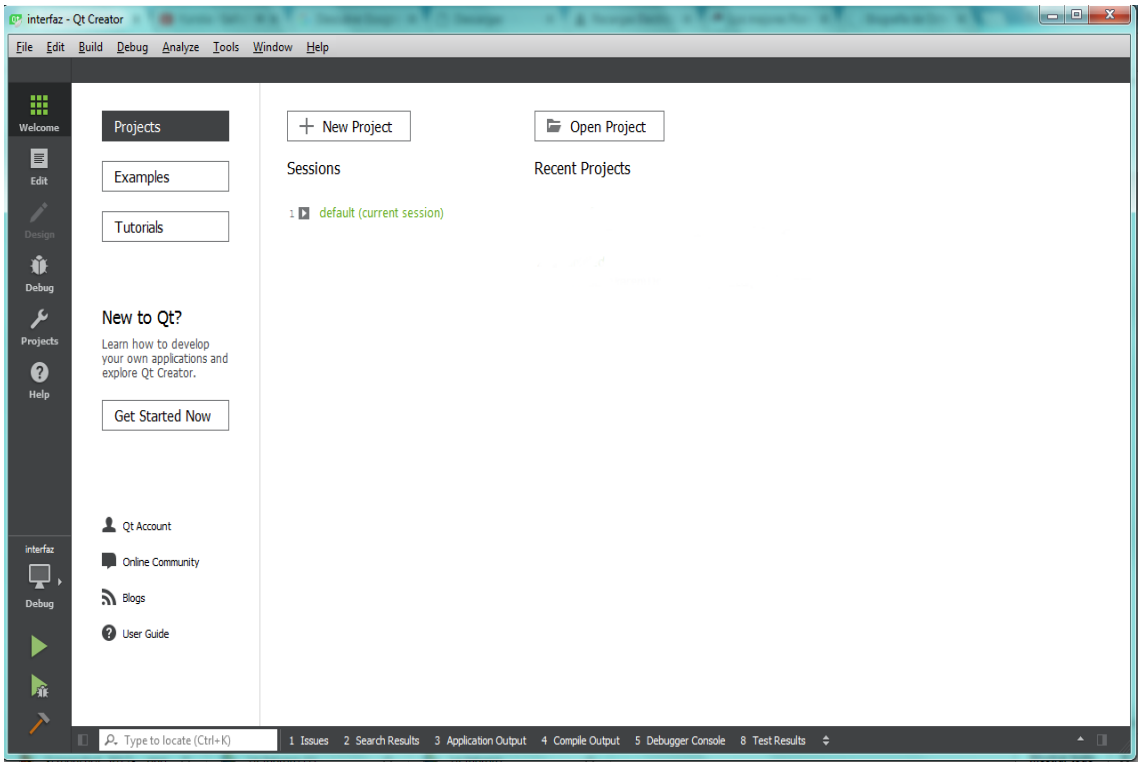
Qt Creator tiene la simplicidad, facilidad, es una herramienta productiva, por estas y las siguientes

características Qt Creator sea convertido en una herramienta de desarrollo exitosa.

- Se puede diseñar una interfaz de usuario con widgets, esto a través de Qt Desginer ya que este posee un editor integrado para widgets
- Se puede desarrollar en lenguajes como C/C++, Python, entre otros
- Para la administración de sus aplicaciones usa una herramienta para el análisis de la memoria que ocupan las aplicaciones.
- Permite crear interfaces de usuarios fluidas e intuitivas, que para el desarrollo de aplicaciones móviles le da un aspecto moderno. Esto es posible a la herramienta Qt Quick en la cual se pueden desarrollar este tipo de interfaces para dispositivos móviles, reproductores multimedia, dispositivos portátiles.
- Permite la iteración entre el diseño y el código.

En la Figura 3.14 se muestra la ventana principal de Qt Creator , en donde la pantalla principal está dedicada a la realizar las tareas de las aplicaciones que en esta se desarrollan. En la parte izquierda de la ventana principal se encuentran tres iconos los cuales son: Projects, Examples, Tutorials. El icono de Projects permite crear un nuevo proyecto o abrir uno existen. El icono de Examplos, permite encontrar algunos de los ejemplos de cuales se puede ver el código e incluso ejecutar y visualizar el proyecto. El icono de tutorial contiene videos de ayuda para la poder realizar un proyecto.

Figura 3.14: Qt Creator ventana principal



3.3.3.1. Creando un proyecto

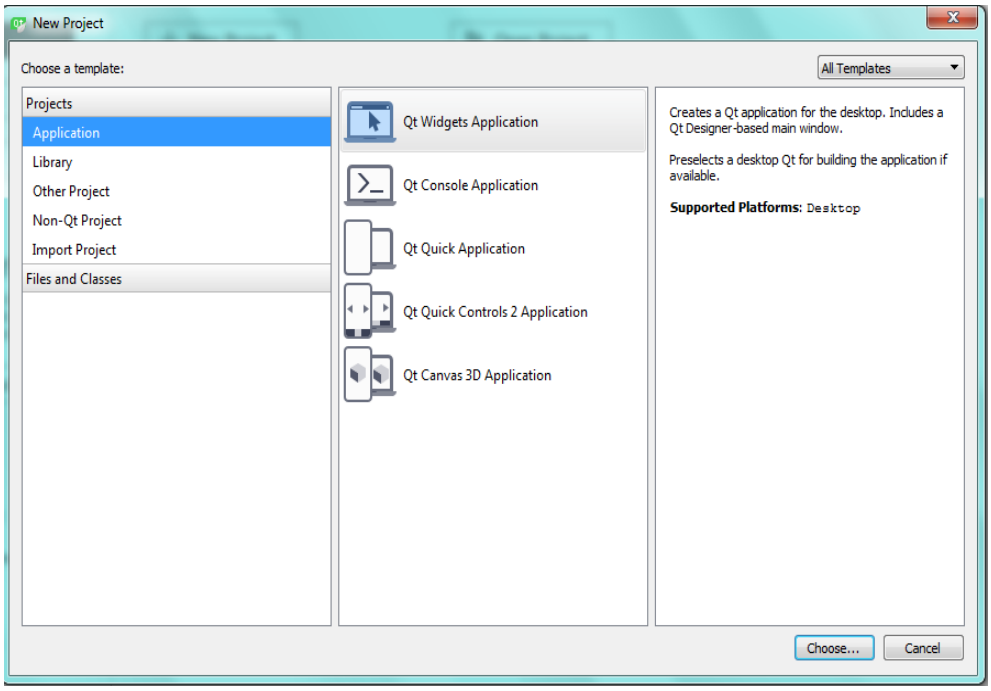
El principal objetivo de Qt Creator es cubrir las necesidades del desarrollo para los que se dedican al desarrollo y diseño de aplicaciones, para que sus proyectos cuentes con los niveles de simplicidad, uso y que sea productivo su desarrollo.

Cuando se construye y compila un proyecto en Qt Creator se necesita lo necesario para el proyecto se pueda compilar y esto se especifica en la ejecución del proyecto. Lo primero que se debe de conocer para poder realizar un proyecto es conocer qué tipo de categoría es la ideal para el tipo de proyecto que se desea crear. Cada parte de la ventana principal de Qt Creator cuenta con información requerida, misma que permite el desarrollo de un proyecto.

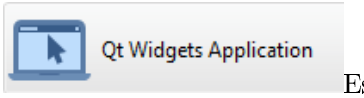
Para crea un proyecto nuevo damos clic en 

y aparecerá una venta de asistencia para elegir el tipo de plantilla que se desea crear.

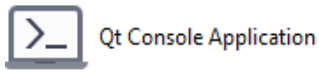
Figura 3.15: New Project

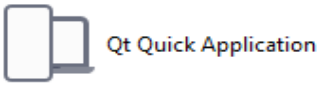


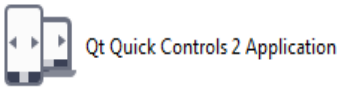
En la Figura 3.15 se muestra las opciones para crear una aplicación

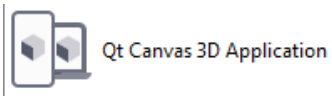
- 

Esta opción que aparece seleccionada, es de aplicaciones esta crea una aplicación para entornos de escritorio, incluyendo una ventana principal diseñada en Qt Designer.

- 

Esta opción crea un proyecto de aplicación para consola, estos proyectos contienen archivos sencillos con extensión .cpp y con la implementación de stub.
- 

permite crear aplicaciones desplegables con Qt Quick.
- 

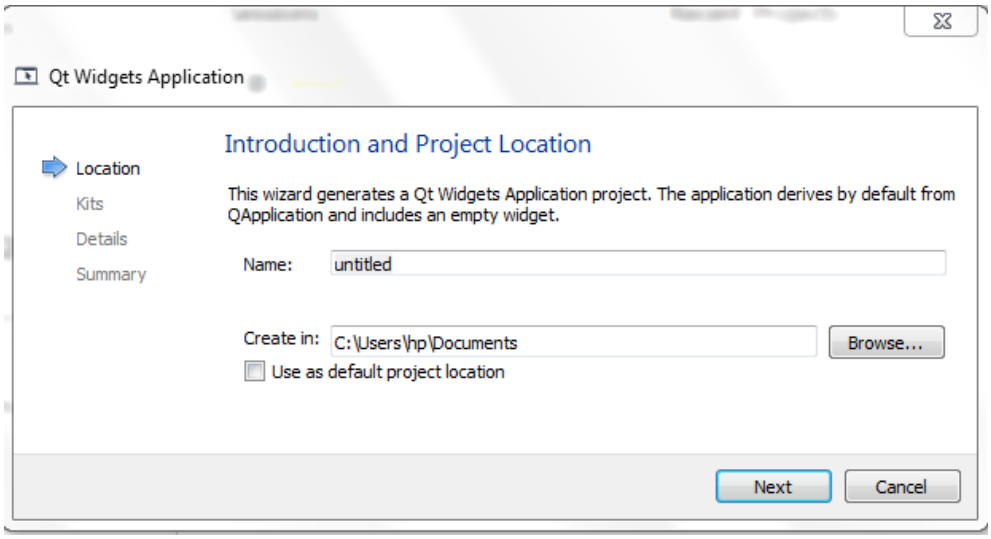
Este proyecto crea una aplicación de Qt Quick 2 desplegable, permitiendo hacer uso de los controles rápidos, estos controles se encuentran disponibles con la versión qt5.7.
- 

crea un qt canvas en 3D proporciona una forma de realizar llamadas para realizar dibujos en 3D del tipo WebGL desde Qt Quick. Esta aplicación usa fuentes de textura y utiliza OpeGL ES 2 para renderizar.

Para el caso de nuestro ejemplo se utiliza el Qt Widgets application, después de elegir el tipo de proyecto el asistente solicitará que definas las configuraciones necesarias para tu proyecto. A continuación se mostrarán las imágenes para la configuración de un proyecto.

La pantalla con el número 1 se coloca el nombre que se le asignará a nuestra aplicación y la ubicación en la cual se guardará, después de definir estos, clic en Next.

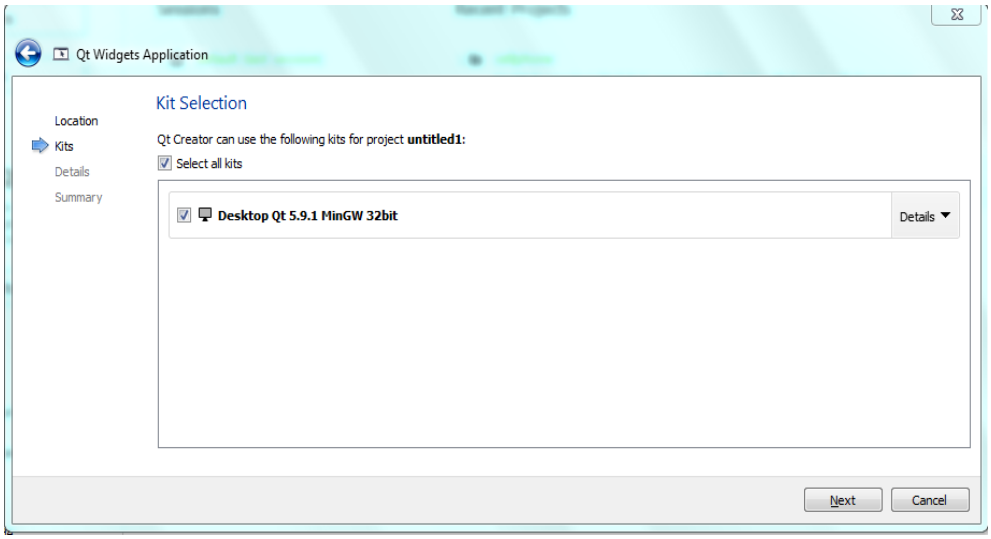
Figura 3.16: configuración de Aplicación



En la Figura 3.17se muestra la ventana con el que se permite seleccionar el kit en el cual se va

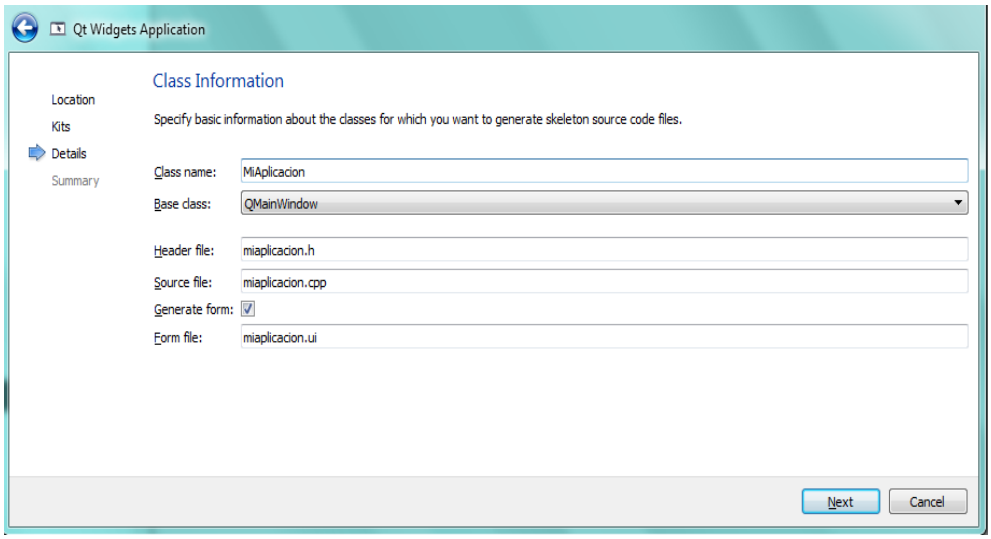
visualizar la nueva aplicación. También nos muestra los detalles dek debug, Release y Profile. Después clic en Next.

Figura 3.17: Kit de selección



En la Figura se especificará la información básica sobre las clases, para generar los archivos fuente, colocaremos el nombre de la clase y de manera automática se coloca en el mismo nombre al archivo de cabecera, al archivo fuente al nombre del formulario. Clic en Next y en la siguiente ventana dar clic en Finish, para finalizar.

Figura 3.18: Información de la clase



Después de esto lo que resultará es que tendremos creados todos los proyectos y junto con esto se crearán todos los archivos para desarrollar.

3.4. Proyectos desarrollados en Qt

Qt se ha convertido en una herramienta muy importante para los desarrolladores y para los programadores, quién son los principales autores de las miles de aplicaciones y programas que hoy en día se ofrece en el mercado de la tecnología. Como ya se mencionó anteriormente Qt es una plataforma, el cual que tuvo gran éxito con la empresa Nokia Corporation, una empresa dedicada a la fabricación de teléfonos móviles, que tiempo después realizó una alianza con Microsft Windows, Digia una empresa dedicada al software compró la licencia de QT que Nokia tenía, haciendo la promesa de renovarla y mejorarla para tener mejor rendimiento y mejoras de los proyectos que están hechos con Qt.

Qt de manera general se utiliza en:

- Autodesk, que es un software de ingeniería usada para el diseño.
- Google Earth, que es un programa informático que permite visualizar múltiples categorías basados en fotografías.
- KDE es un entorno de escritorio desarrollado para diversos sistemas operativos, como GNU/-Linux.
- Adobe Photoshop es un editor gráfico, usado para el retoque de fotografías y gráficos.
- Agencia Espacial Europea, es una agencia especializada para la exploración espacial.
- Opie es un fork, desarrollado por la empresa de Trolltech, diseñada para el soporte de estandares que contiene una serie de aplicaciones para PDAs y dispositivos en los que se encuentre un sistema Linux.
- Entre otros como Volvo, Walt Disney Animation Studios, VLC media player, Samsung, Philips, Panasonic, etc.

3.4.1. Ventajas de usar Qt

- Es comprometido con su desarrollo, además de contar con código abierto, que le permite avanzar más que los proyectos que están financiados.
- Utilizar una editor visual, esto permite que se puedan crear interfaces fluidas, permitiendo el ajuste de dimensiones.
- La herramienta de Qt Creator utiliza una herramienta de edición bastante sofisticada, que permite la edición, depuración, gestión, localización y la compilación de los proyectos para equipos de cómputo y equipos móviles.

- Las herramientas de Qt son de uso gratuito, es decir, se pueden bajar y usar incluso para crear proyectos de uso comercial.
- Por sus aplicaciones para múltiples plataformas también Qt está presente en algunas aplicaciones populares como Google Earth o Skype, Symbian, Maemo y muy pronto en dispositivos como MeeGo.
- Para el desarrollo de los programadores de Python, la Comunidad de Qt ha logrado que los desarrolladores puedan programar con python y utilizando interfaz de Qt.

Capítulo 4

Desarrollo de aplicaciones con Python y Qt

Con firme propósito de crear interfaces gráficas para usuarios, Python contiene varias bibliotecas de cuales podemos tomar para poder interfaces, estas bibliotecas son: PyGTK, wxPython, PyQt, Tkinter, entre otros, de todas sólo se explicará PyQt, ya que es una poderosa herramienta para la programación orientada a objetos (POO).

PyQt es la unión de la biblioteca gráfica de Qt y el lenguaje de programación Python, en otras palabras en un conjunto de herramientas que permiten que se trabaje el diseño gráfico en Qt y se ensamble con la programación del lenguaje Python, permitiendo que se ejecuten en la mayoría de las plataformas como por ejemplo, Windows, OS X, Linux, iOS y Android. PyQt cuenta con las licencias de GNU GPL v3 y la licencia comercial de Riverbank [19].

PyQt está desarrollada por la compañía británica Riverbank Computing, por Phil Thomson. PyQt se implementó con los módulos de Python de los cuales se mencionan los de uso más común como QtGui, QtCore, QtMultimedia, QtNetwork, QtOpenGL, QtScript, QtSQL, QtSvg, QtWebKit, QtXml, QtAssistant, QtDesigner, que en la Tabla 4.1 [16] se describen. Estos módulos cuenta con más de 300 clases y aproximadamente 6000 funciones y métodos, esto en la versión PyQt4, pero a partir de la versión PyQt5 se incrementaron hasta más de 1,000 clases y este último es recomendable, ya que la versión anterior no es compatible con biblioteca de Qt, además de constituirse en varios módulos.

Cuadro 4.1: Módulos de PyQt

Modulo	Descripción
QtGui	tiene los componentes básicos y las clases relacionadas de los botones, ventanas, barras de estado y de herramientas, deslizadores, mapa de bits, colores y fuentes.
QtCore	Se usa para trabajar con tiempo,archivos, directorios, tipos de datos, flujos y url.
QtMultimedia	Es un modulo que proporciona los tipos QML y las clases de C++ para utilizar el contenido multimedia. En conjunto permite también accede a la cámara y la función de radio.
QtNetwork	Tiene las clases para la programación de redes, facilitando que la programación del cliente y el servidor de TCP/IP y UDPT, sea más sencilla y portable.
QtOpenGL	Es utilizado para la visualización de gráficos en 2D y 3D
QtScript	Es un modulo que permite la evaluación, depuración de scripts y la utilización de objetos y funciones avanzados.
QtSql	Contiene las clases para trabajar con bases de datos.
QtSvg	Contiene las clases para desplegar archivos SVG (Scalable Vector Graphics).
QtWebKit	Es un motor de procesamiento que ofrece soporte para tecnologías web estándar.
QtXml	Contiene las clases para trabajar con los archivos XML
QtAssistant	Es una herramienta para ver documentación en línea.
QtDesigner	Herramienta diseñada par acrear interfaces gráficas de usuario.

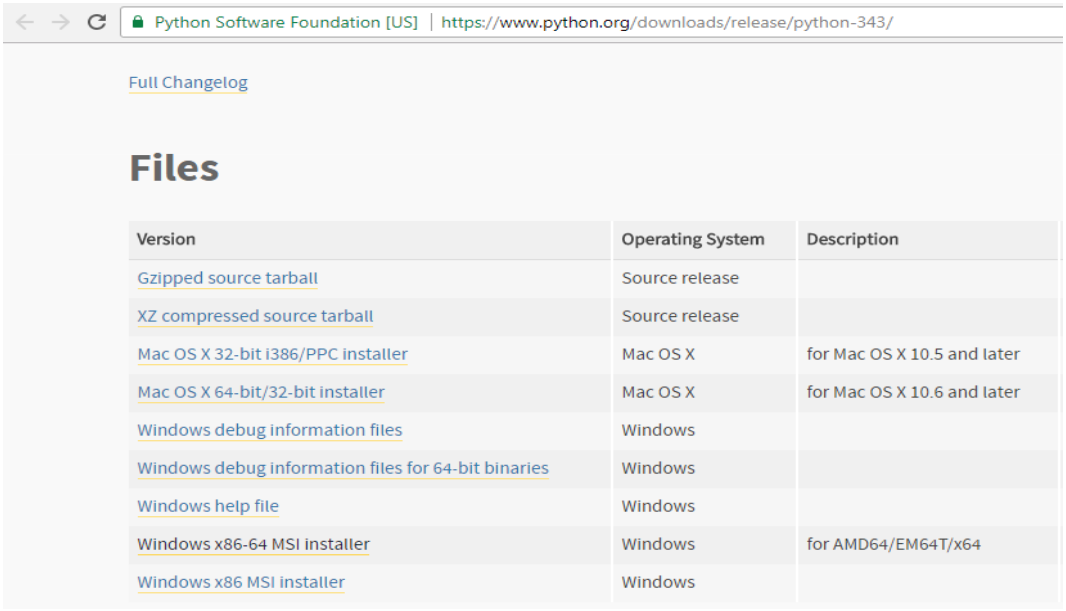
4.1. Instalación Python

En este apartado se abordará la instalación de Python y PyQt4 , para este caso la instalación se realiza para la plataforma de Windows, lo cual permitirá tener el compilador de Python y el manejador de interfaz gráfica. En capítulo anterior se habló de Python y recordemos que es un lenguaje de programación multiplataforma, potente y su curva de aprendizaje es sencilla, ya que no requiere tantos conocimientos de programación. Para los que inician en el mundo de la programación resulta ser una herramienta muy útil ya que su código y la sintaxis es muy amigable para comprenderla.

Comencemos con la instalación de Python para Windows,

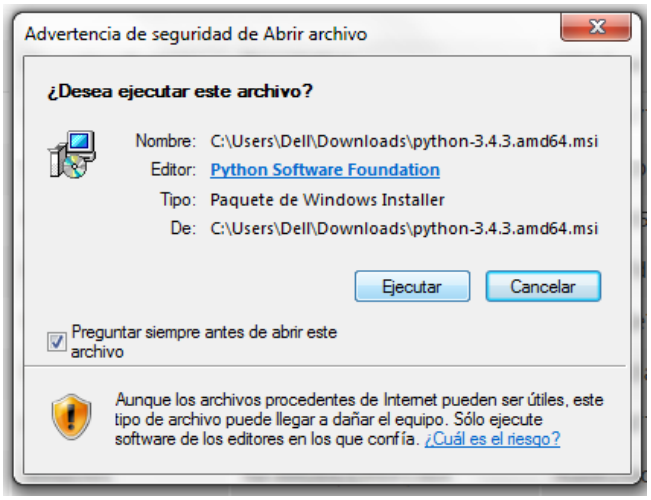
- Para esto nos vamos al link `https://www.python.org/downloads/release/python-343/`, estando en la pagina oficial de Python nos movemos a la sección «Files», que se encuentra hacia abajo de la pagina. Ver la Figura 4.1 .

Figura 4.1: Página principal de Python



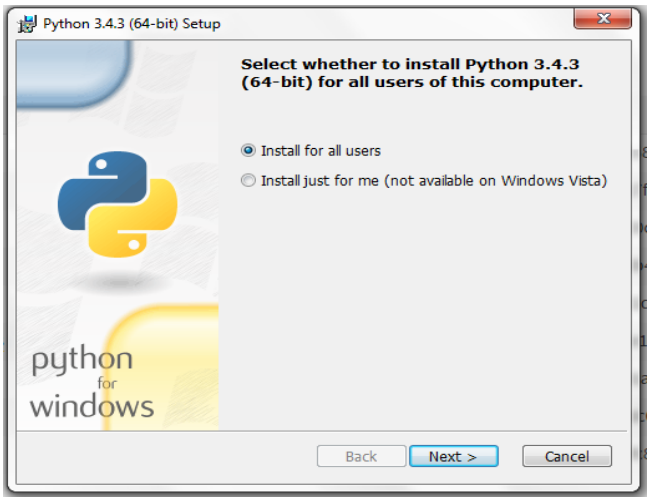
- Para el caso de Windows es necesario conocer la arquitectura que tiene la computadora donde se instalará Python, en este caso será para una arquitectura de 64 bits, entonces se descarga el archivo con la extensión *.msi (un archivo msi es un instalador de Windows, que proviene del instalador de Microsoft). Despues de descargase por completo, se da doble click en el archivo para ejecutarlo. Ver Figura 4.2.

Figura 4.2: Ejecutar



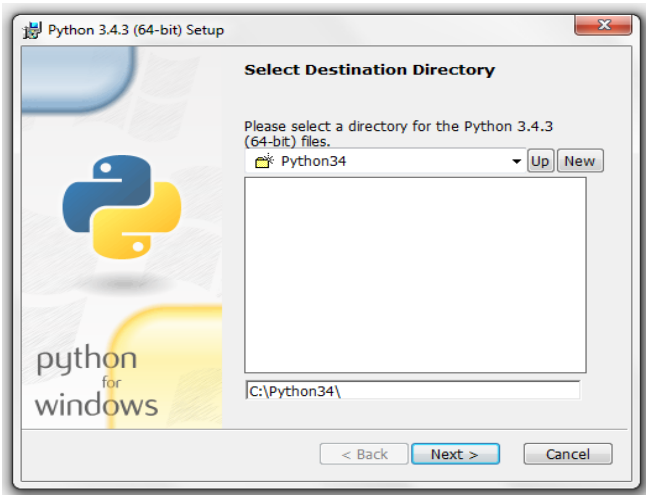
- Aparecerá el asistente de instalación, en mi caso deja las opciones por default, que es la que esta selecciona como se muestra en la Figura 4.3 y clic en Next.

Figura 4.3: Asistente de instalación



- Seleccionar el directorio destino (ver Figura 4.4) y clic en Next.

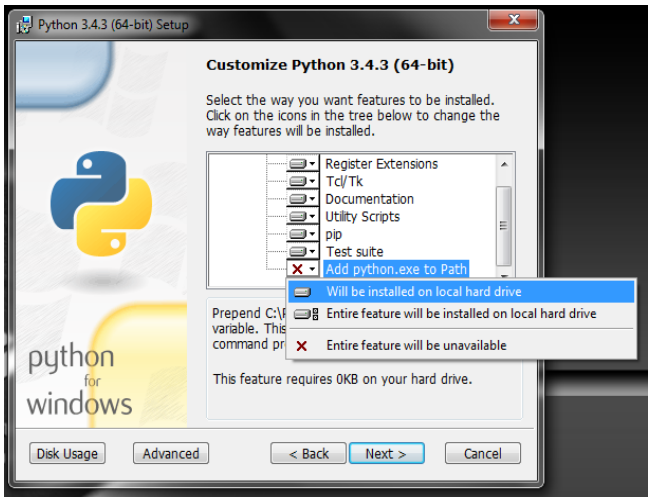
Figura 4.4: Directorio destino



- En la siguiente ventana de instalación que es la de personalizar a Python, hay que tener en cuenta que la ultima opción del arbol esta desabilitada, se habilita dando clic en la priemra

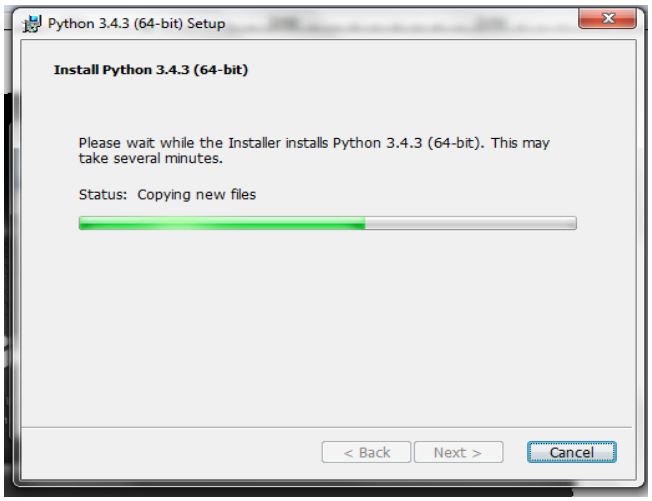
opción, para instar Python en el disco duro local. Ver Figura 4.5. Clic en Next

Figura 4.5: Personalizar Python



- En la siguiente Figura 4.6 se muestra el progreso de la instalación

Figura 4.6: Progreso de la instalación de Python



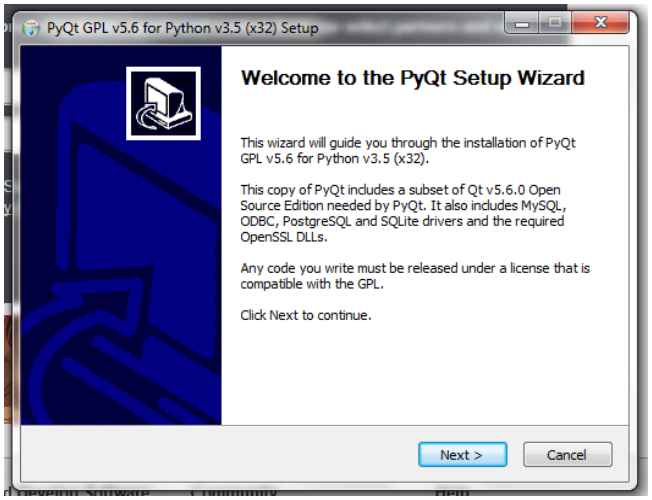
- Al terminar la instalación nos mostrara la siguiente ventana que se muestra en la Figura, clic en Finish y se puede comprobar que esta instalado Python en nuestra computadora revisando los programas.

Figura 4.7: Finaliza instalación de Python

4.2. Instalación PyQt4 y configuración

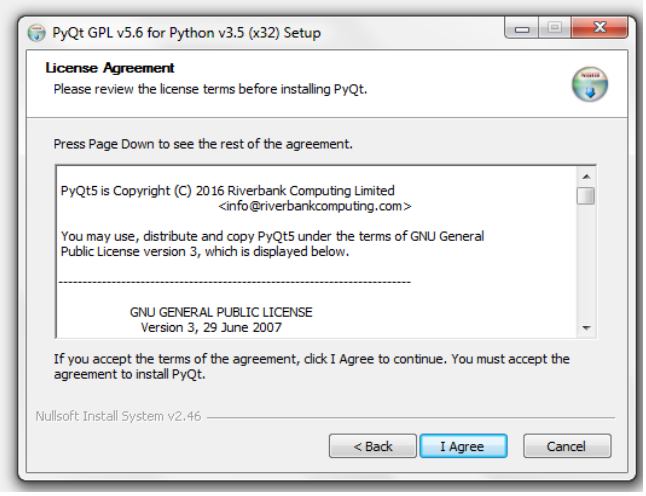
La herramienta de PyQt es multiplataforma para el desarrollo de sistemas, bajo el lenguaje de programación de Python y la librería pyqt4 en las siguientes páginas se muestra la instalación de Pyqt4. En el capítulo anterior se mostró la instalación de Python para Windows de 64 bits (para este caso, o bien puedes descargar la versión de acuerdo a tus necesidades y recursos). Como primer paso se descarga la versión de pyqt4 que corresponda con la versión de Python que se instaló, para esto descargamos de la siguiente página, la instalación es muy típica en entornos de Windows y no habrá ningún problema. En la siguiente imagen aparece el asistente de instalación y damos clic en Next.

Figura 4.8: Asistente de instalación



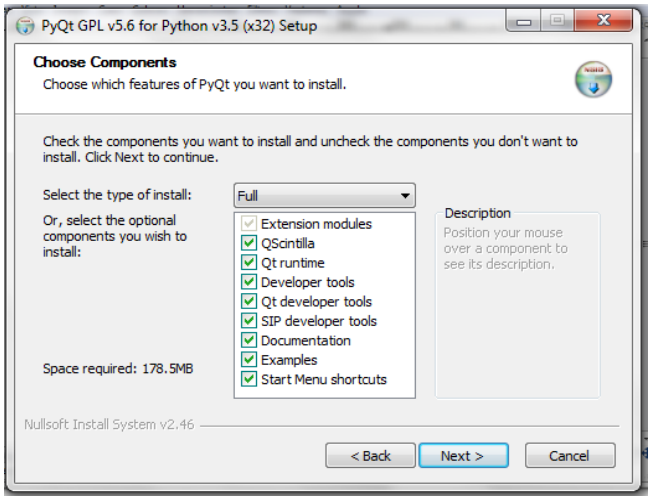
Después aparecen los términos de la licencia, clic en I agree.

Figura 4.9: Términos de la licencia



En las siguiente imagen se muestran los componentes que se van a instalar junto con pyqt, para esto se tiene la opción de decidir que componentes se quieren instalar y cuales no son requeridos, al terminar de elegir clic en Next

Figura 4.10: Componentes



A continuación se muestra la carpeta de ubicación de la instalación, clic en Next

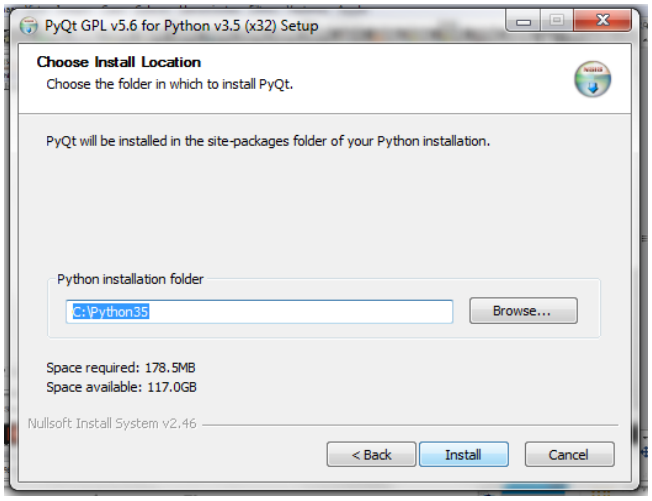
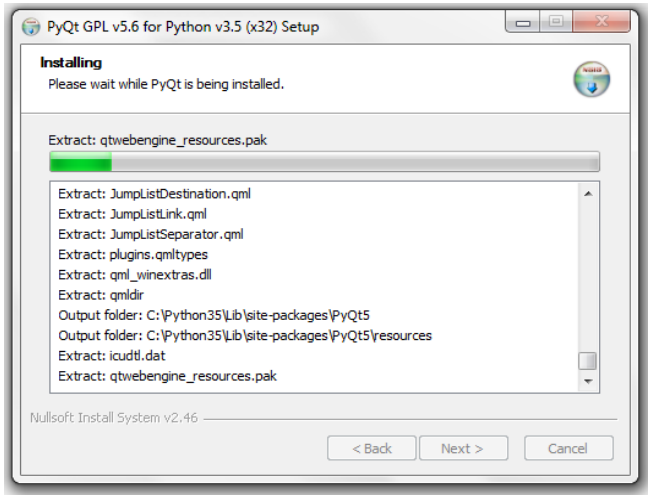


Figura 4.11: Localización de la instalación

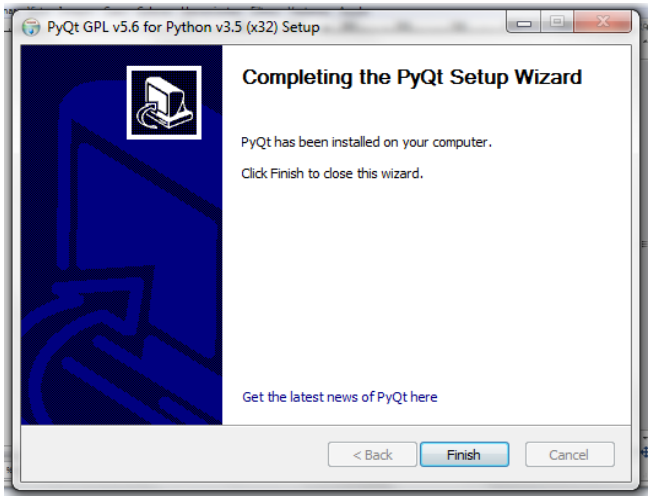
Después empezará el progreso de la instalación y se mostraran los detalles que lo que se extrae y se instala, al terminar clic en Next

Figura 4.12: Instalación PyQt4



Finaliza la instalación de PyQt y clic en Finish.

Figura 4.13: Finalizar la instalación PyQt4

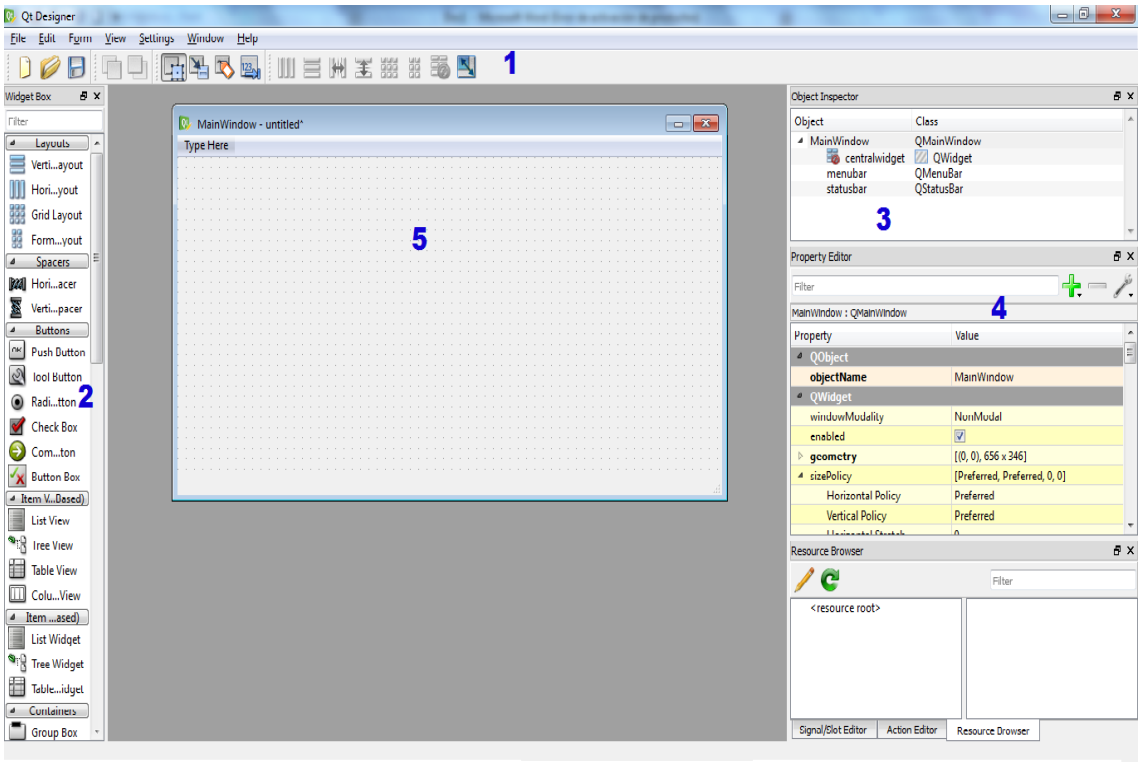


Una vez finalizado la instalación puedes observar en todos los programas que se creo una carpeta que almacena las herramientas que puedes utilizar para el desarrollo de sistemas. En los programas se encuentran son: Assistant, Designer y Linguist. Designer contiene los elementos necesarios para diseñar y manipular formularios, mismos que se pueden manejar con python.

4.2.1. Utilizando Qt Designer

Esta herramienta permite diseñar interfaces gráficas de usuarios (GUI). Después de que se haya instalado PyQt 4, para el caso de Windows en el menú de inicio podremos encontrar la herramienta de Qt Designer, ver Figura 4.14.

Figura 4.14: Partes de Qt Designer



Las partes con las que está conformado nuestro diseñador de interfaces es la siguiente:

1. **Barra de Menú y herramientas.** Contiene los elementos principales para crear, abrir y guardar un formulario. Además el menú de las herramientas del formulario.

2. **Barra de Widget Box**, esta contiene todos los elementos como son etiquetas, cajas de texto, botones etc. Todos los objetos necesarios para crear la interfaz.

3. **Object Inspector**, se puede observar como una lista de manera jerárquica de todos los objetos que contenga el formulario que se esté empleando. Cada objeto que se encuentre en el formulario se puede modificar sus propiedades haciendo clic con el botón derecho, o con tecla rápida Ctrl + F. Si se hace doble clic en el objeto, se puede cambiar el nombre.

4. **Propiedades de los objetos.** en este se concentran todas las propiedades de entrada de los objetos en el formulario. Cuando se trata de editar la propiedades de los objetos podemos tratar de los siguientes: propiedades textuales, enteras, booleanas y compuesta. Estas pueden ser vistas de diferente manera como las propiedades textuales, su modificación se puede ver en una línea. Para el caso de las propiedades enteras la forma de ver esta propiedad es por medio de cajas de cambio. Las propiedades booleanas se pueden en casillas de verificación. Y por último las propiedades compuestas tales como los colores y los tamaños, estas se representan

mediante listas. Dentro de las propiedades de los elementos existen las propiedades dinámicas, su función principal es agregar nuevas propiedades tanto a los formularios como a los widget. Estas propiedades se encuentran en la parte superior derecha de la ventana principal de las propiedades y podemos observar que tienen el icono  .

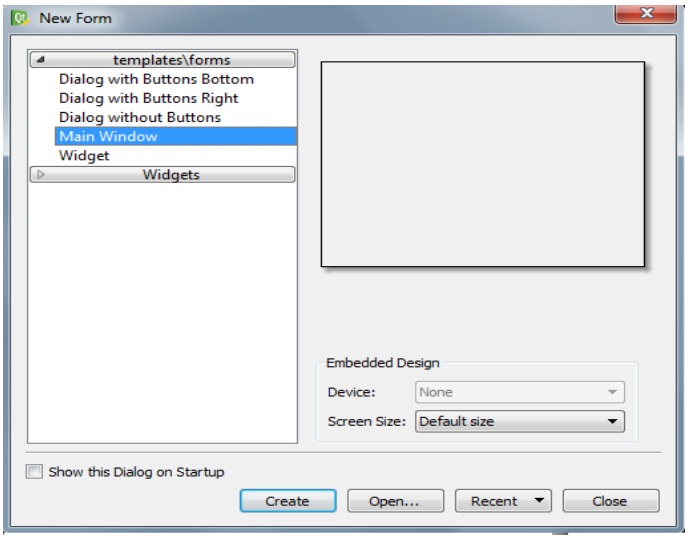
5. **Formulario o área de trabajo**. Es el espacio asignado para diseñar la interfaz y es el que aloja todos los objetos para el diseño de la interfaz.

Cuando se emplea Qt Designer se aplican cuatro pasos esenciales.

1. Elegir un diseño de formulario
2. Definir los objetos que se emplearán en el formulario
3. Realizar la conexión de señales.
4. vista previa del formulario.

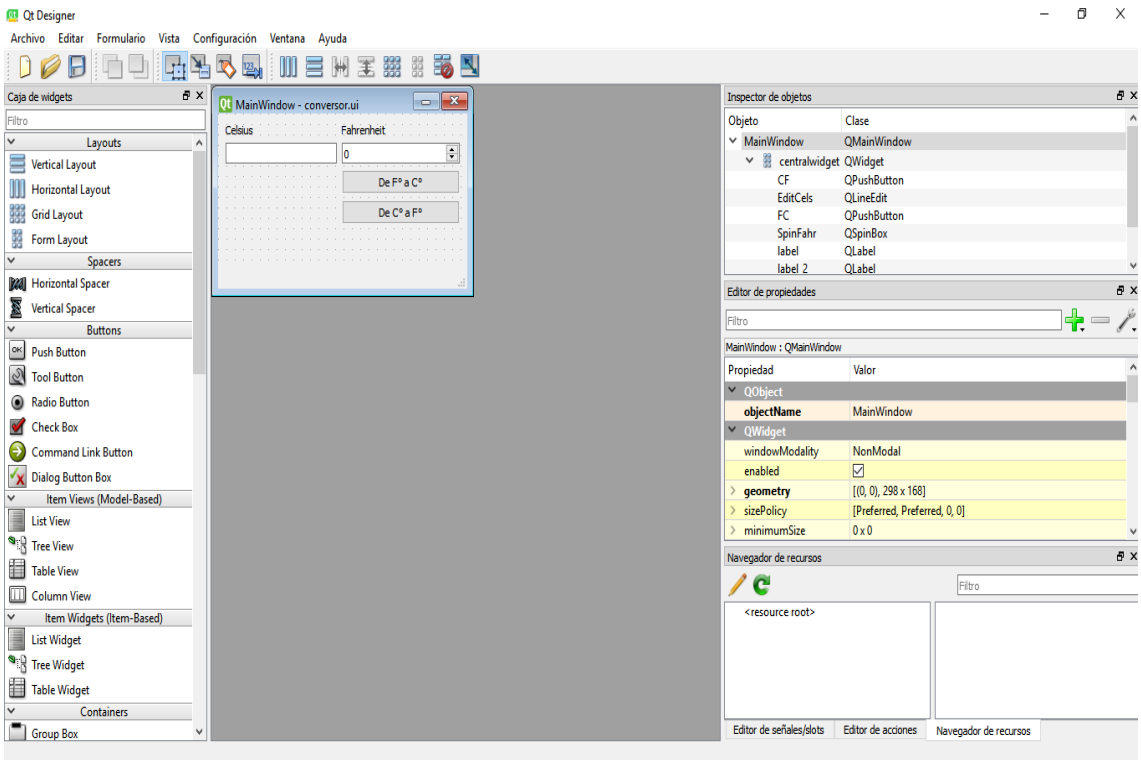
Ahora tenemos el software necesario realicemos un ejemplo de una pequeña interfaz en Qt Designer para conocer sus elementos, lo primero que nos aparecerá cuando abrimos QtDesigner es la ventana principal sin formularios. Después le vamos a dar en la opción de NEW para crear un Form Nuevo y aparece la siguiente ventana (ver Figura 4.15), en la cual se selecciona la opción Main Window, clic en el botón de Create

Figura 4.15: New from



Y aparecerá el formulario de la siguiente forma. Ahora ya tenemos listo nuestro espacio de trabajo para realizar nuestra interfaz. En la Figura 4.16se observa la colocación de algunos objetos, los cuales realizarán la función de convertir grados Celsius a Fahrenheit y viceversa.

Figura 4.16: GUI convertidor de grados



El método para que los objetos sean insertados en el formulario es de manera sencilla, se pueden arrastrar. Cuando se requiera se puede guardar el proyecto, seleccionando en el menú File -> Save o Save As..., también es posible hacer uso de las teclas rápidas, para este caso es Crtl + s. Los objetos pueden ser seleccionados dando clic sobre ellos o bien con el botón izquierdo del mouse. Para seleccionar objetos adicionales se mantiene oprimida la tecla Mayús y haciendo clic sobre ellos.

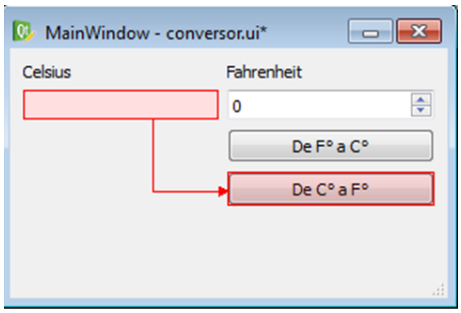
4.2.2. Señales y slot

El modo edición de señales tiene la función de conectar los objetos de forma conjunta permitiendo que exista la gestión de eventos para las conexiones señales-slot. La traducción de slot es una ranura, o dicho en otras palabras se relaciona a una acción de "tomar". En los diseños se puede conectar una interfaz con las señales y estas se guardan cuando el formulario es guardado.

Cuando se conectan los objetos, nos dirigimos al modo de edición de señales y ranuras en el menú Edit y elegir la opción Edit Signals/Slots o haciendo uso de las teclas rápidas, presionando la tecla F4.

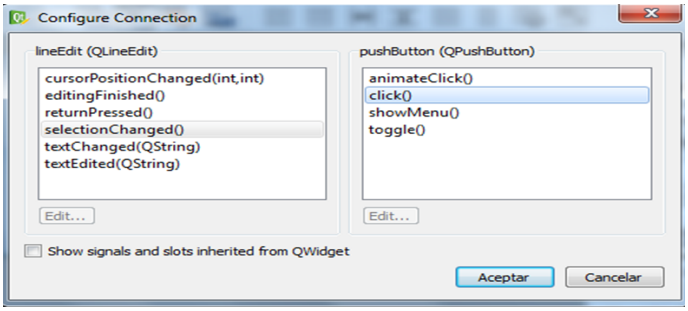
En la Figura 4.17 podemos observar cómo se realiza una conexión, cuando el cursor se coloca sobre algún objeto se resalta en color rojo y se arrastra hasta el objeto que se desea conectar y la línea quedará en línea con punta de flecha, indicando que se dirige al objeto destino.

Figura 4.17: Conexión en Qt Designer



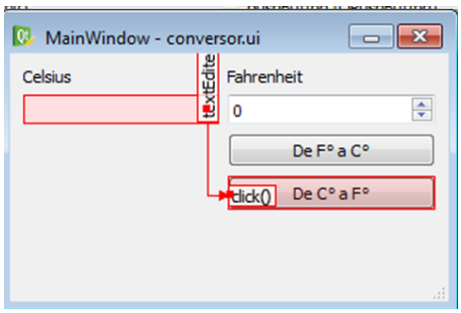
El cuadro de diálogo de la Figura 4.18 muestra la conexión que se va a realizar entre los objetos, es decir se muestra la señal del objeto origen y la ranura del objeto destino, y para terminar con la configuración de la conexión clic en Aceptar.

Figura 4.18: Configuración de conexiones



En la Figura 4.19se observa cómo se visualiza la conexión de los objetos. Los Widgets como los diseños se pueden conectar de manera intuitiva. Las conexiones que decidas hacer serán tantas como desees hacer, en el formulario se puede conectar a un objeto, o dicho de otra manera un objeto se puede conectar al formulario, esto se representa mediante un signo de tierra en el formulario.

Figura 4.19: Visualizar la conexión



Cuando una conexión se realiza de manera automática se crean dos etiquetas como las que se observan en la Figura 4.17. Estas conexiones se pueden cambiar después de que haberse configurado, esto mediante haciendo doble clic sobre las etiquetas y se muestra el cuadro de diálogo de la conexión. Para eliminar una conexión se selecciona la ruta de conexión y se oprime la teclas Suprimir.

4.3. Usando el Diseño en Qt Designer

Los objetos de un formulario deben de estar colocados correctamente cuando se muestre el formulario, esto también implica que el diseño puede redimensionarse correctamente. El objetivo principal para aplicar el diseño, es para administrar los objetos existentes dentro del formulario. El diseño sirve para que los objetos no se muevan y no se pueda cambiar su tamaño, permitiendo ajustar la geometría de cada uno de los objetos.

El menú de diseño  se encuentra en la barra de herramientas o en el menú contextual.

Este menú de diseño se pueden colocar un diseño horizontal o vertical  , ambos garantizan la alineación de sus elementos dependiendo de su diseño.

También existe el diseño en cuadrícula  , que permite tener más control sobre la ubicación de los objetos y permite el uso más libre sobre los widgets y permite que su diseño sea menos flexible.

El diseño Splitter  administra los objetos colocando un divisor, este se encarga de organizar los objetos de manera horizontal o vertical pero permitiendo al diseñador ajustar el espacio asignado para cada objeto.

Para terminar o interrumpir el diseño, hacer clic en el icono  , o mediante el menú principal Form elegimos Break Layout, o haciendo uso nuevamente de las teclas rápidas con Ctrl + 0.

Teclas rápidas para el diseño,ver Cuadro 4.2

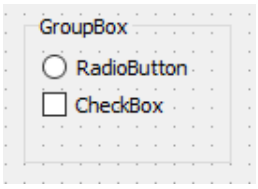
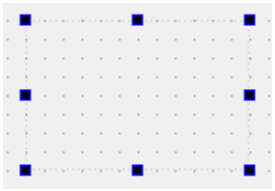
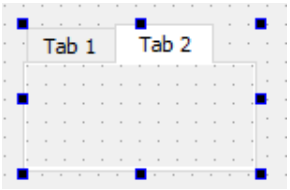
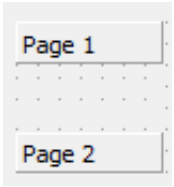
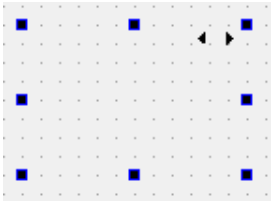
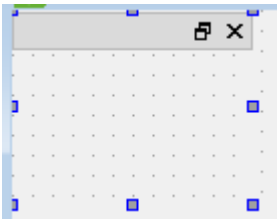
Cuadro 4.2: Teclas rápidas

Tipo de diseño	Icono	Teclas rápidas	Descripción
Eliminar diseño		Ctrl + 0	Se usa para romper un diseño
Horizontal		Ctrl + 1	Se usa para el diseño de los objetos de manera horizontal
Vertical		Ctrl + 2	Se usa para el diseño de los objetos de manera vertical
Cuadrícula		Ctrl + 5	Se usa para el diseño de los objeto en forma de cuadrícula
Form		Ctrl + 6	Se usa para diseñar en un diseño de formulario
Divisor horizontal		Ctrl + 3	Se utiliza para crear un divisor horizontal.
Divisor vertical		Ctrl + 4	Se utiliza para crear un divisor vertical.
Ajuste de tamaño		Ctrl + J	Se utiliza para ajustar el tamaño del diseño.

4.3.1. Contenedores

Los contenedores juegan un papel importante dentro de los widgets, ya que proporcionan un control sobre los objetos que se encuentran dentro de ellos. Qt Designer tiene la facilidad de proporcionar a los diseñadores que puedan crear automáticamente su código. De esta manera el contenedor normalmente permite crear objetos secundarios, que se organicen en diferentes diseños. Este apartado es importante porque es de gran utilidad conocer para qué sirve o para que se emplea cada contener y esto ayuda despejar la duda de qué tipo de contenedor emplear para la necesidad de cada diseño. En cada contenedor la administración de pende del tipo de contenedor, en el Cuadro 4.3se mostraran algunos de los contenedores disponibles.

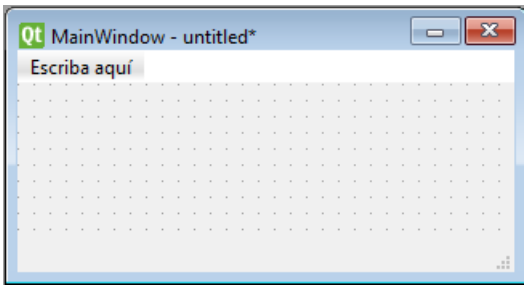
Cuadro 4.3: Tipos de contenedores

Tipo de contenedor	Descripción	Representación gráfica
Caja de grupo	Son empleadas para almacenar colecciones de casillas para la verificación y botones de radio. Estas cajas por lo general pueden contener su propio diseño, pero si se desea aplicar un nuevo diseño se puede hacer.	
Marcos	Es utilizado para agrupar widgets, para la decoración para contenedores complejos o de posición para formularios.	
Widgets de pestañas	Estos widgets permiten dividir en diferentes secciones, mediante las pestañas que se pueden eliminar, agregar y cambiar de nombre.	
Caja de herramientas	Estos permiten colocar una serie de páginas o compartimientos muy similar a una caja de herramientas. Se puede cambiar el nombre de casa páginas y también eliminar páginas.	
Widgets apilados	Estos se caracterizan por tener capas y dentro de ellas puede contener objetos como el combo box. las flechas de la esquina superior derecha indican que se puede deslizar al final de todos los componentes.	
Widgets de muelle	Es se trata de paneles flotantes, que contienen entradas y controles de widgets mucho más complejos. Este widgets se fija al tamaño de los bordes de la página principal donde se está diseñando, simulando ser una caja de herramientas.	

4.3.2. Ventanas y Menús

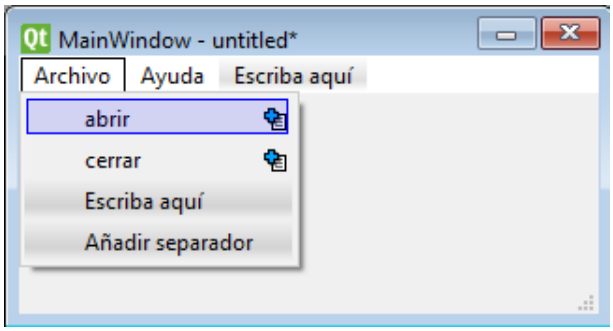
Qt Designer es una herramienta muy poderosa en cuestiones de creación de interfaces y sobre todo de uso para múltiples propósitos y diversas plantillas para crear formularios. En las plantillas podemos emplear ventanas, crear menús con cajas de herramientas y widgets. Las plantillas principales proporcionan una opción para crear barras de menú. Un menú puede contener varias cajas de herramientas. Los menús se agregan a las barras modificadas por los marcadores. Cuando se crea una venta con menú, como es el caso de la Figura 4.20 , este menú se puede acceder a sus propiedades mediante el inspector de objetos, así como también se puede eliminar el contenido contextual del menú.

Figura 4.20: Ventana de menú



Para crear nuevos elementos en el menú, se da doble clic en el menú para agregar nuevos elementos y a continuación en el nuevo elemento aparece una lista de opciones como la que se observa en la Figura 4.21.

Figura 4.21: Ejemplo de menú



Para aceptar el nuevo elemento se da doble clic en el elemento. Es característico de un menú que exista mnemónicos, esta técnica es recomendable para favorecer el aprendizaje y aumentar la capacidad de recordar cosas. Contemplar tener un orden en el menú es prioridad, ya que esto permite a los usuarios finales familiarizarse mucho más rápido y de manera sencilla con la interfaz. Es posible reorganizar los elementos del menú con solo arrastrar y soltarlos en donde se prefiera.

En cada menú se pueden colocar la cantidad de entradas que sean necesarias al igual que los separadores. Las entradas se pueden añadir a profundidad, tanto como sea posible o como dependiendo de las necesidades del diseño. 4.5 Ejemplos simples con GUI En este apartado se muestran ejemplos de programas con interfaz gráfica diseñada en Qt Designer y programada en Python. Esto en otras palabras se llama desarrollar programas en PyQt.

Estos son los pasos para crear una interfaz gráfica en Qt Designer:

- Teniendo abierta la herramienta de Qt Designer, dar clic en New.
- Aparecerá un ventana en donde aparecen varias opciones de Form, y seleccionamos la que dice "Main Window" y clic en Create
- El formulario que se crea tiene incrustado un menú bar, la cual se puede remover para solo tener el formulario limpio o si lo que deseas hacer es con un menú puedes conservarlo.
- Después insertar los elementos necesarios para el diseño de GUI.
- Terminando de diseñar tu GUI, se guarda la interfaz con la extensión .ui, esto haciendo clic en "File" y después de "Save" o directo desde las teclas rápidas con Ctrl + s, eleiges la ruta donde se guardara la GUI.

Cuando ya tenemos lista la interfaz, de manera automatica QtDesigner le otorga su extensión .ui, teniendo eso ahora toca el turno de Python pero antes el archivo con extensión .ui debe de ser exportado a codigo fuente Python, en otras palabras con extensión .py. Para la conversión de la interfaz.

4.3.3. Herramienta pyuic

Python cuenta con una herramienta llamada pyuic, que se encarga de la exportación de los archivos con extensión.ui a los archivos fuentes de Python. ¿Pero qué significa pyuic? "py" por Python, "ui" que significa user interface

Esta herramienta viene integrada con PyQt cuando se instala. La ruta donde se encuentra el pyuic depende del S.O. que se esté usando, para este caso el S.O. es Windows y la ruta es la siguiente ***C:\Python27\Lib\site-packages\PyQt4\uic\pyuic.py -o C:\reproductor.py C:\reproductor.ui,*** observemos que en la ruta aparece "PyQt4", el número 4 es por la versión de PyQt.

Para poder hacer uso del script pyuic es necesario usar el símbolo del sistema de Windows (cmd), después, ya estando ahí nos ubicamos en la ruta donde se guardó el arhivo.ui que será el archivo a convertir, para este caso se creó una carpeta en documentos.

La sintaxis para usar el pyuic es la siguiente:

pyuic4 [options] ui-file

pyuic4 –x archivo.ui –o archivo.py

Observa que se coloca la opción –x que hace la acción de ejecutar el archivo.ui generando código adicional que crea y muestra la GUI cuando se ejecuta como una aplicación. También se coloca la opción de –o que la acción que realiza es escribir un archivo de salida.

Después de generar el archivo con pyuic es importante resaltar que ese archivo generado no se debe de modificar, ya que ese archivo contiene todos los elementos de la interfaz a manera de código y si se modifica ese código se generaran errores en la compilación del programa final. En el caso que hubiera una modificación de la interfaz desde el gestor de Qt Designer, se tendría que guardar y exportan nuevamente.

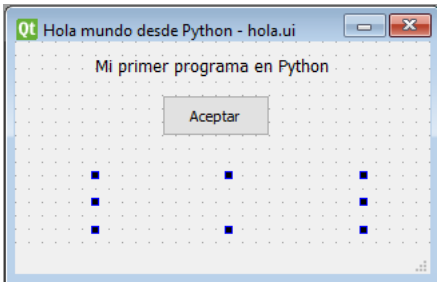
Después de tener ya nuestra exportación, se crea un nuevo archivo para este caso se hace uso de la herramienta de IDLE de Python, que es el entorno de desarrollo que se encuentra integrado en Python. IDLE cuanta con dos ventanas, una es la de Shell y la otra la ventana de Editor, que para nuestros ejemplos de este trabajo siempre se ocupara el editor.

4.4. Mi primera aplicación "Hola mundo"

En este apartado se creara el primer programa con GUI diseñada en QtDesginer y programada en Python. Como en todos los lenguajes de programación por lo general el primero programa que se realiza es el tradicional "Hola mundo", y para nuestro proyecto empezaremos con este.

Anteriormente se mencionaron los pasos que debemos seguir para empezar a diseñar nuestra interfaz. Para este sencillo programa solo se ocupa un botón, el cual realizara la función y una etiqueta que mostrará el mensaje. En la Figura 4.22 se muestra los elementos que se utilizaron para esta interfaz.

Figura 4.22: Interfaz Hola



Teniendo ya nuestra interfaz, exportamos nuestro archivo llamado hola.ui, como se observa en la Figura 4.23 en la cual se puede observar el convertidor pyuic4, clic en enter.

Figura 4.23: Convertidor pyuic

Después de tener listo el archivo hola.py, en la Figura 4.24 se observa el código que genero el pyuic.

Figura 4.24: Archivo generado con pyuic

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'hola.ui'
#
# Created: Fri Jun 22 21:31:06 2018
#      by: PyQt4 UI code generator 4.10.3
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName(_fromUtf8("MainWindow"))
        MainWindow.resize(311, 174)
        self.centralwidget = QtGui.QWidget(MainWindow)
        self.centralwidget.setObjectName(_fromUtf8("centralwidget"))
        self.pushButAceptar = QtGui.QPushButton(self.centralwidget)
        self.pushButAceptar.setGeometry(QtCore.QRect(110, 40, 81, 31))
        self.pushButAceptar.setObjectName(_fromUtf8("pushButAceptar"))
        self.label = QtGui.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(60, 100, 201, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.label.setFont(font)
        self.label.setText(_fromUtf8(""))
        self.label.setObjectName(_fromUtf8("label"))
        self.label_2 = QtGui.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(60, 10, 201, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_2.setFont(font)
        self.label_2.setObjectName(_fromUtf8("label_2"))
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtGui.QStatusBar(MainWindow)
        self.statusbar.setObjectName(_fromUtf8("statusbar"))
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(_translate("MainWindow", "Hola mundo desde Python", None))
        self.pushButAceptar.setText(_translate("MainWindow", "Aceptar", None))
        self.label_2.setText(_translate("MainWindow", "Mi primer programa en Python", None))


if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    MainWindow = QtGui.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

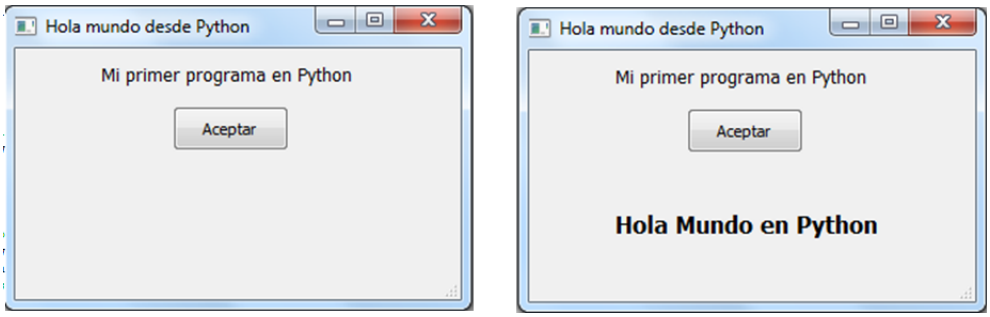
Se guarda el archivo con el nombre de funcióngHola.py, para este ejemplo se guarda con este nombre para identificarlo del archivo que se generó del pyuic y del que contiene las instrucciones programadas en Python. El siguiente paso es ejecutar el código anterior, y para esto volvemos a la consola y escribimos lo siguiente, ver Figura 4.25 :

Figura 4.25: Ejecutar pyuic

```
C:\Users\Luis\Documents\Hola1>python funcionHola.py
```

Se obtiene como resultado la siguiente ventana que se muestra en la Figura 4.26. La ventana de la izquierda muestra el resultado de la compilación, presionas el botón de "Aceptar", muestra el mensaje que se ve en la ventana de la derecha.

Figura 4.26: Hola mundo en Python con GUI



Con esto se concluye el famoso "Hola mundo", ahora se explicara de manera breve el codigo de este programa:

- La primera parte que está conformada por las primeras 5 líneas se importan las librerías de PyQt y el archivo denominado "hola"
- Las segunda partes está formada por la definición de la clase (class), se define y se llama al constructor `QtGui.QMainWindow.__init__(self)` . Dentro de la misma clase se accede a los elementos de la interfaz por medio de conexiones (connect) y al mismo tiempo se le coloca la señal (SIGNAL) que son los que ejecutan la acción.
- La tercera parte está definida por `def Click(self):` que define el método para mostrar el mensaje en la etique.
- Y por último la parte final del programa donde se coloca import sys y las últimas instrucciones para verificar que se importen los módulos, que se otorguen las funciones de la interfaz, mostrar la clase, mostrar la ventana y cerrar el intérprete.

En la creación de este sencillo programa se muestra de manera general como se crea y ejecutar una interfaz, así como la programación en Python.

4.5. Pares e impares con Python

En este apartado se aborda el tema de los números pares e impares, en el cual determinaremos si un número insertado es par o impar

Se comienza diseñando una interfaz, recordemos cual es el problema a resolver:

- Determinar si un número es par o impar

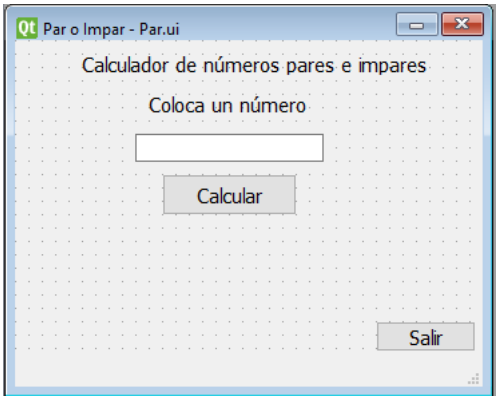
Esto significa que el usuario insertara un número y ese número deberá ser evaluado para determinar si es par o impar, recordemos que los números pares son aquellos que son divisibles entre dos y cuyo resultado es cero. Los números impares son aquellos que no son divisibles entre dos y cuyo resultado siempre es 1.

La GUI está conformada por:

- 3 label, 2 contienen un mensaje y la otra contendrá el mensaje "El número es..."
- 1 LineEdit, es el elemento donde se insertara los números
- 2 Push Button, 1 tiene la etiqueta de "Calcular", este realizará la función de evaluar el numero insertado en la LineEdit y mostrara el resultado en una Label. El otro Push Button tiene la etiqueta de "Salir", este botón su nombre lo dice, tiene la función de salir o cerrar el programa.

En la Figura 4.27se observa de manera gráfica como esta diseña la GUI.

Figura 4.27: GUI de Par e impar



Como en el programa anterior de "Hola mundo", se guarda la GUI y se exporta para convertir la GUI a código Python, teniendo esto se crea un archivo para programar en lenguaje Python.

En la Figura 4.28 se muestra el código del programa de par o impar.

Figura 4.28: Código de Par e impar

```
#-*- coding: utf-8 -*-
#importar libreria PyQt
from PyQt4 import QtCore, QtGui
from PyQt4.QtGui import *
from PyQt4.QtCore import*
# se importa la interfaz
import Par
#crear clase principal
class Pary(QtGui.QMainWindow):
    def __init__(self):
        QtGui.QMainWindow.__init__(self)
        #para aaceder a los elementos de la interfaz
        self.aut = Par.Ui_MainWindow()
        #line_edit = QtGui.QLineEditInsert()
        self.aut.setupUi(self)
        self.connect(self.aut.pushButCal,SIGNAL("clicked()"),self.Click)
        #self.connect(self.aut.pushButBorrar,SIGNAL("clicked()"),self.borrar)
        self.connect(self.aut.pushButSalir,SIGNAL("clicked()"),self.salir)
    def Click(self):
        numero = self.aut.lineEditInsertar.text()
        #funcion type retorna el tipo de dato
        if int(numero) %2 == 0:
            self.aut.labelResultado.setText("El numero " + str(numero)+ " es par")
            self.aut.lineEditInsertar.setText("")
        else:
            self.aut.labelResultado.setText("El numero " + str(numero)+ " es impar")
            self.aut.lineEditInsertar.setText("")
    def salir(self):
        exit()
    import sys
if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    MainWindow = QtGui.QMainWindow()
    ui = Par.Ui_MainWindow()
    mostrar = Pary()
    mostrar.show()
    sys.exit(app.exec_())
```

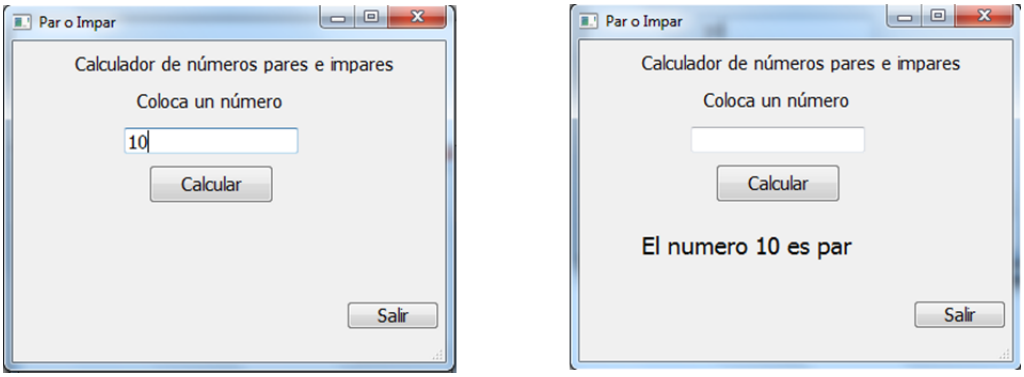
Teniendo el código listo, se ejecuta desde la terminal con la siguiente línea, recuerda que hay que moverse a la ruta donde se guardo la GUI y el archivo exportado con pyuic y ejecutamos con esta instrucción, ver Figura 4.29.

Figura 4.29: Ejecutar el GUI de Par e impar

```
C:\Users\Luis\Documents\Par>python FuncionPar.py
```

En la Figura 4.30 se muestra el resultado de la compilación, y se observa que el programa pide insertar número, en la imagen de la izquierda se muestra que se inserta el número 10 y al hace clic en el botón "calcular", el resultado se en la imagen de la derecha.

Figura 4.30: Compilación de par e impar



Al igual que el anterior programa la estructura es muy parecida, ahora se describe el código de esté programas.

- En la primera parte se importan las librerías de PyQt incluyendo el archivo donde se encuentran los elementos de interfaz, convertido a código Python. Esto se identifica porque tienen la palabra reserva `from e import` .
- La segunda parte del código está formado por la definición de la clase, la creación e inicialización del constructor (def __init__(self), QtGui.QMainWindow.__init__(self)). El constructor sirve para inicializar los atributos de los objetos que se crean y también para ejecutarlos de manera automática. En esta misma sección se realizan las conexiones y las señales de los objetos de la interfaz.
- Después se define def Click(self):, que es el método el cual tiene dentro la acción que realizara el botón y la etiqueta, así como también se define la instrucción if y else para que realice la evaluación del número insertado en la etiqueta. Existe otro método denominado def salir(self), cual la acción que realizara es salir del programa, mediante el botón "Salir".
- Por último se encuentra la parte donde se verifica que los módulos sean importados, muestra la clase, muestra la ventana y la salida de la ventana.

4.6. Calcular factorial de un número

La función factorial quiere decir que hay que multiplicar todos los números positivos que hay entre el número dado y el número 1. El factorial está representado por signo de exclamación "!". Por ejemplo: el factorial de 5! es

$5! = 1x2x3x4x5 = 120$

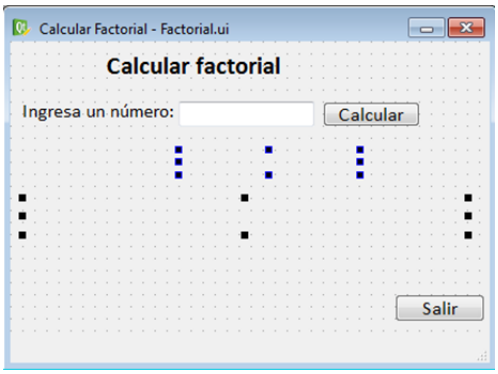
Para este caso el problema a resolver es obtener el factorial de un número, teniendo noción de que es un factorial y como se obtiene, ahora solo queda diseñar un interfaz para la inserción del número y programar en lenguaje Python.

El Qt Designer se diseña la interfaz para el factorial, y estos son los elementos que ocupan:

- 4 Label, dos contienen mensajes, la otra muestra el mensaje del factorial a convertir, y otra label muestra el resultado al calcular el factorial.
- 1 LineEdit, en el cual se insertara el número.
- 2 PushButton, uno para calcular la operación del factorial y el otro para realizar la acción de salir de la ventana.

De los elementos anteriores se pueden ver de manera gráfica en la Figura 4.31, los cuales se algunos se pueden ver como son las Label y están seleccionados para su visualización.

Figura 4.31: GUI Factorial



Teniendo lista nuestra interfaz, se guara y se exporta a lenguaje Python. A continuación sigue realizar el código en Python para calcular el factorial. Recuerda que no se modifica el archivo exportado por pyuic, para realizar el factorial se crea otro archivo dentro de la misma carpeta donde esta almacenada la GUI.

En la Figura 4.32 se observa el código de la función factorial.

Figura 4.32: Código Función factorial

```
from PyQt4 import QtCore, QtGui
from PyQt4.QtGui import*
from PyQt4.QtCore import*
#from math import*
import Factorial
import sys
class Facto(QtGui.QMainWindow):
    def __init__(self):
        QtGui.QMainWindow.__init__(self)
        self.aut = Factorial.Ui_MainWindow()
        self.aut.setupUi(self)
        self.connect(self.aut.ButtonCal,SIGNAL("clicked()"),self.factorial)
        self.connect(self.aut.ButtonSalir,SIGNAL("clicked()"),self.salir)
    def factorial(self):
        numero = int (self.aut.lineEditInsertar.text())
        resul = 1
        for i in range(numero):
            resul *= (i + 1)
            numero -= 1
        self.aut.label_3.setText("El factorial de: "+ self.aut.lineEditInsertar.text()+ "!")
        self.aut.labelResul.setText("es: "+ str(resul))
        self.aut.lineEditInsertar.setText("")
    def salir(self):
        exit()
import sys
if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    MainWindow = QtGui.QMainWindow()
    ui = Factorial.Ui_MainWindow()
    mostrar = Facto()
    mostrar.show()
    sys.exit(app.exec_())
```

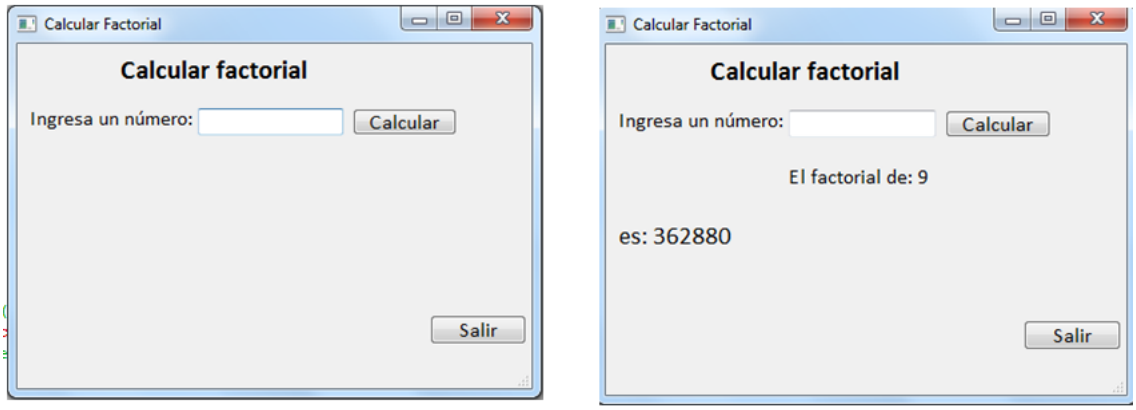
Teniendo ya el código para compilar el factorial, ahora se guarda y complia con la siguiente linea desde el cmd de windows, observa la Figura 4.33.

Figura 4.33: Compilar el factorial

```
C:\Users\Luis\Documents\Facto>python FuncionFactorial.py
```

El resultado de la compilación se muestra en la Figura 4.34, la imagen de la izquierda muesta la interfaz compilada en espera de insertar un número para calcular el factorial. La imagen de la derecha muestra el número insertado, que en este caso es 9 y el resultado del factorial este número es: 362880.

Figura 4.34: Resultado de la compilación



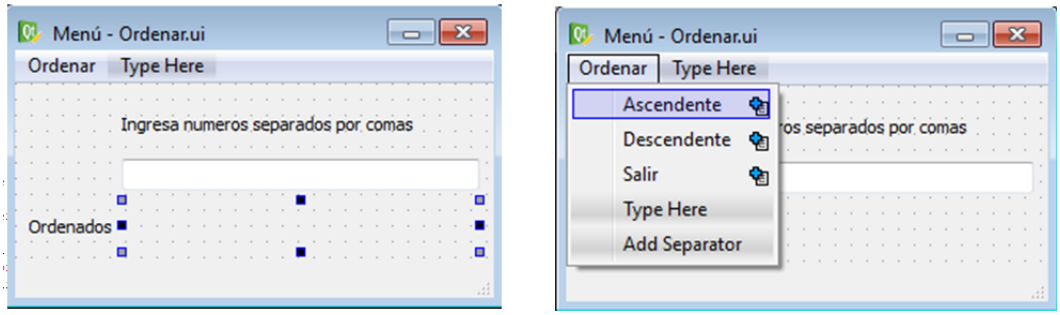
Se describe de manera general el código de esta programa:

- La primera parte esta argumentado por la importación de la librerías de PyQt y la importación de la interfaz convertida en código Python que en este caso es Factorial.
- Las siguiente parte está dada por la declaración de la clase `class Facto(QtGui.QMainWindow):`, la creación e inicialización del constructor, así como la conexión y las señales de los PushButton.
- En la tercera parte se define un método denominado `def factorial(self):`, el cual se encuentra toda la estructura de la función factorial, en esta función de hace uso de bucles como el `for`. El siguiente método declarado es de salir `def salir(self):`, el cual realiza la acción de salir de la ventana.
- La última parte se encarga de verificar que los módulos se carguen correctamente para la ventana diseña en Qt Designer sea visualizada.

4.7. Ordenamiento de números empleando un menú

Ordenar una serie de número puede ver de manera muy tediosa y que llevara un poco más de tiempo. La formas de ordenar números es de mayor a menor y viceversa, para temas de programación esto se interpreta como listas. El siguiente problema que se resolverá es el ordenamiento de números, para esto se le pide al usuario que inserte número al azar. Pero para ponerle un poco más de detalle a la interfaz se colocara un menú, en cual se despliegan las opciones de Ascendente y Descendente. En la Figura 4.35 se muestra la GUI para el ordenamiento de números.

Figura 4.35: GUI ordenamiento de números con menú



Esta GUI esta formada por los siguientes elementos:

- 1 Menú Bar el cual contiene las opciones de Ascendente, Descendente y Salir (ver imagen de la derecha).
- 3 Label, una tiene la leyenda de "ingresar números...", otra dice "Ordenados" y la ultima mostrara el resultado de los números ordenados.
- 1 LineEdit en la cual se ingresan los números.

Se guarda la interfaz en una carpeta y se coloca el nombre de Menu.ui, lo siguiente que se hace es exportar esta interfaz a código Python, esto por medio de pyuic.

En un archivo nuevo se escribe el código para hacer el ordenamiento de números, en la Figura 4.36 se muestra este código y mas adelante se describe.

Figura 4.36: Código de ordenamiento de números

```
from PyQt4 import QtCore, QtGui
from PyQt4.QtGui import*
from PyQt4.QtCore import*
import Ordenar
import sys
class Orden(QtGui.QMainWindow):
    def __init__(self):
        QtGui.QMainWindow.__init__(self)
        self.aut = Ordenar.Ui_Menu()
        self.aut.setupUi(self)
        self.connect(self.aut.actionAscendente,SIGNAL("triggered()"),self.ordenarAsc)
        self.connect(self.aut.actionDescendente,SIGNAL("triggered()"),self.ordenarDes)
    def ordenarAsc(self):
        texto = list(str(self.aut.numeros.text()))
        for num in range(len(texto)-1,0,-1):
            for i in range(num):
                if texto[i]>texto[i+1]:
                    temp = texto[i]
                    texto[i] = texto[i+1]
                    texto[i+1] = temp
        print texto
        resultado = " ".join(str(x) for x in texto)
        print " ".join(str(x) for x in texto)
        self.aut.ordenados.setText(resultado)
    def ordenarDes(self):
        textol = list(str(self.aut.numeros.text()))
        for numl in range(len(textol)-1,0,-1):
            for j in range(numl):
                if textol[j]<textol[j+1]:
                    templ = textol[j]
                    textol[j] = textol[j+1]
                    textol[j+1] = templ
        print textol
        resultado = " ".join(str(y) for y in textol)
        print " ".join(str(y) for y in textol)
        self.aut.ordenados.setText(resultado)
import sys
if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    MainWindow = QtGui.QMainWindow()
    ui = Ordenar.Ui_Menu()
    mostrar = Orden()
    mostrar.show()
    sys.exit(app.exec_())
```

- En este código se al igual que los otros programas siempre la primera parte está formada por la importación de librerías de PyQt y la importación de la interfaz.
- En la siguiente parte de define la clase `class Orden(QtGui.QMainWindow):` y se construye e inicia al constructor `def __init__(self):`. También se encuentran instanciados los elementos de tiene alguna acción en la interfaz, en este caso los elementos de menú "Ascendente, Descendente y Salir". o Ascendente y Descendente tiene la señal de `"triggered()"`, que es un dispador.
- Después se definen los métodos para realizar el ordenamiento, el primer método que se define es el ordenamiento Ascendente `def ordenarAsc(self):`, en el cual tomará la serie de números

que se ingresan en `texto = list(str(self.aut.numeros.text()))`, así se asigna a la LineEdit; y esto sera lo que entra en un bucle `for` para determinar el orden ascendente de los números ingresados. `.join` devuelve los elementos insertados en forma de cadena, esto porque los une por medio de un str.

- Se define el método Descendente `def ordenarDes(self):` en el cual se evalúan los números ingresados en el LineEdit(numeros) y son comparados uno por uno dentro del bucle `for`. También cuenta con un método `.join` que la acción que realiza es unir los elementos por medio de un str.
- Por ultimo resultados del ordenamiento son almacenados en una variable llamada resultado y esta misma es lo que se muestra cuando se elige la acción de Ascendente o Descendente.
- Al final del código se encuentran la verificación de que los módulos sean cargados correctamente, también de que se muestre la ventana y se cierra.

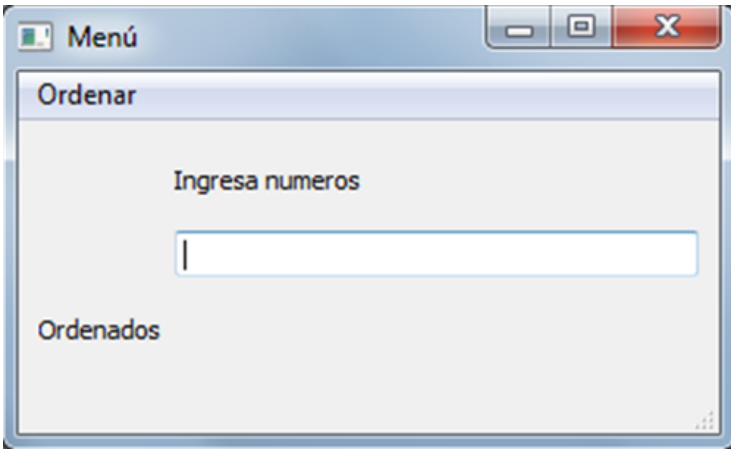
Se guarda en la misma ruta donde se encuentra la GUI, este archivo se guarda con la extensión .py y se compila de la siguiente manera, ver Figura 4.37:

Figura 4.37: Compilación menú

```
C:\Users\Luis\Documents\Ordenar>python FuncionOrdenar.py
```

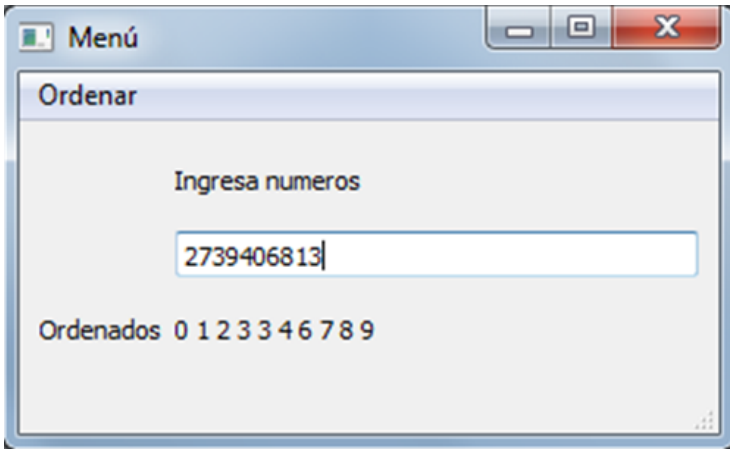
En la Figura 4.38 se muestra el resultado de de la compilación

Figura 4.38: Resultado de compilación



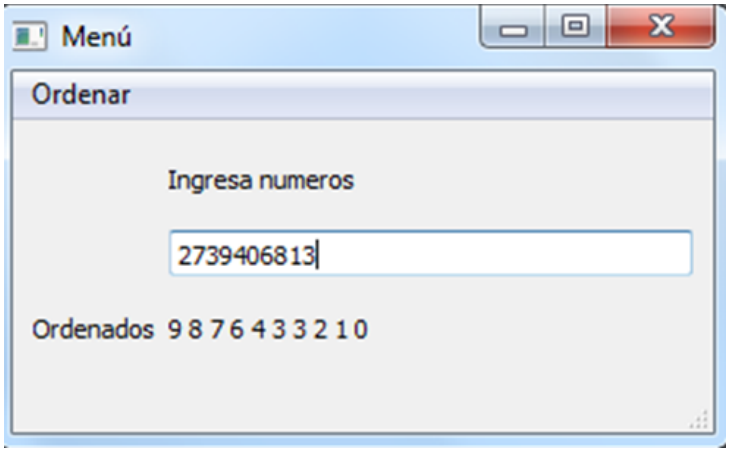
Se ingresa números y se elige el orden Ascendente, observa en la Figura 4.39 para ver los resultados.

Figura 4.39: Ejemplo de Orden Ascendente



Realizamos el ordenamiento Descendente con estos mismo números y observa en la Figura 4.40 que muestra los resultados de este ordenamiento.

Figura 4.40: Ejemplo Orden Descendente



Con esto ejemplos de programas desde el "Hola mundo", hasta el "Ordenamiento de números", se muestra el funcionamiento y la creación de GUI con QtDesigner en conjunto con la programación en lenguaje Python.

Capítulo 5

Conclusiones

El lenguaje de programación Python es uno de los cinco lenguajes más usados en el mundo en 2018, de acuerdo a la página https://www.tiobe.com/tiobe-index/. Sin embargo, en el Centro Universitario UAEM Zumpango así en otros espacios académicos de diferentes universidades, se presta poca atención a la enseñanza de este lenguaje. La sintaxis de Python es simple y más fácil de aprender con respecto a otros lenguajes, lo que permite un desarrollo de aplicaciones de forma rápida, y tiene la capacidad para crear interfaces gráficas de usuario con entornos de trabajo como Qt. Python tiene una licencia tipo libre, es de código abierto, además es multiplataforma, por lo que ha sido utilizado en diversos proyectos de importantes empresas. El propósito principal de este trabajo es popularizar el uso de Python en el CU UAEM Zumpango, y de mostrar la forma de desarrollar aplicaciones con este lenguaje a través de ejemplos simples pero completos. Es por ello que a lo largo del documento se ofreció un panorama amplio sobre el desarrollo de aplicaciones con Python. En los fundamentos se abordaron los temas de operadores, cadenas, estructuras de datos estándar, funciones y estructuras de control de flujo. Para cada tema, no únicamente se explicó el concepto, sino también se expusieron ejemplos simples que fueron creados para facilitar la explicación, estos ejemplos fueron retomados de algunas clases de UA como Programación Estructurada y Estructuras de Datos del programa educativo de Ingeniero en Computación que se imparte en el CU UAEM Zumpango. De esta forma, se presentan de una forma sencilla las principales características de Python, con lo que se espera pueda ser atractivo a alumnos y profesores de este campus de la UAEM, así como de otras instituciones.

Bibliografía

[1] & C. Meyers A. Downey, & J. Elkner. *Aprende a pensar como un programador con Python*. Green Tea Press, 2002.

[2] Á. Arias and I.T.C. Academy. *Aprende a Programar en Python:*. CreateSpace Independent Publishing Platform, 2015.

[3] M.I. Bobadilla. *Nociones Básicas de Python:*. Miguel Iván Bobadilla, 2016.

[4] S. Chazallet. *Python 3: los fundamentos del lenguaje*. Recursos informáticos. Ediciones ENI, 2016.

[5] M.K. Dalheimer. *Programming with Qt: Writing Portable GUI applications on Unix and Win32*. O'Reilly Media, 2002.

[6] Raúl González Duque. *Python para todos*. Creative Commons, 2012.

[7] A. FERNANDEZ. *Python 3 al descubierto - 2a ed.:*. Editorial Ink, 2013.

[8] Carlos Huamaní. Tutorial básico de python -parte iii: Funciones. In *Tutorial básico de Python -parte III: Funciones*, 2018.

[9] A. Martelli, A. Ravenscroft, and S. Holden. *Python in a Nutshell: A Desktop Quick Reference*. O'Reilly Media, 2017.

[10] A. Marzal, I. Gracia, and P. García. *Introducción a la programación con Python 3*. Departamento de Lenguajes y Sistemas InformÂŽaticos Universitat Jaume I, 2014.

[11] Natsys. *Introducción a Phyton:*. Monty Python, 2017.

[12] Mark Pilgrim. *Inmerción en Python 3*, 2009.

[13] Angélica Pineda. Perfiles y sueldos en el pmercado de las tics. *Expansi[ón*, 2018.

[14] Jakub Przywski. Python reference (the right way). In *Python Reference (The Right Way)*, 2015.

[15] published by the Free Software Foundation. *Qt Documentation*. The Qt Company Ltd, version 1.3 edition, 2017.

[16] Learn PyQt. *PyQt Tutorial*. Tutorials Points, 2018.

[17] rescursos python. recursos python. In *Recursos Python*, 2017.

[18] R. Rischpater. *Application Development with Qt Creator - Second Edition*. Community Experience Distilled. Packt Publishing, 2014.

[19] Riverbank. Pyqt. In Riverbank computing, editor, *What is PyQt ?*, 2016.

[20] Martin Sande. Programación en c++ con qt bajo entorno gnu/linux, 2004.

[21] B. Sintes. *Introducción a la Programación con Python*, April 2017.

[22] Choc Cac Software. Introducción programación en qt y qml. In *Introducción programación en Qt y Qml*, 2015.

[23] Sphinx. Documentación python. Technical report, Python Software Foundation, 2017.

[24] J. Thelin. *Foundations of Qt Development*. Expert's Voice in Open Source. Apress, 2007.

[25] Geffrey Velásquez Torres. prograación multiplataforma con qt, 2007.

[26] Guido van Rossum. *Tutorial de Python*. Software Fundation, fred l. drake, jr. edition, September 2009.

[27] R. Wachenchauzer, M. Manterola, M. Curia, M. Medrano, and N. Paez. *Algoritmos y Programación I con lenguaje Python*. Creative Commons, 171 Second Street, Suite 300, San Francisco, California,, 2011.