INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

La Programación Orientada a Objetos (POO) es un paradigma de programación que busca que nuestra forma de programar sea más cercana a la forma como nos relacionamos en nuestro día a día.

Clases

Es la descripción de un conjunto de objetos similares; consta de métodos y de datos que resumen las características comunes de dicho conjunto

Objetos

Elemento o individuo de la clase

Atributos

Características del objeto

Métodos

Son las acciones que puede realizar el objeto

Clase : Gato
Atributo : 2 orejas , 2 años
Métodos : maullar , saltar
Objeto : michilion

En Python

```python
class Gato:
    def __init__(self):
        self.edad = 2
        self.orejas = 2
    def maullar(self):
        print("Miau ... Miau")
    def saltar(self):
        print("Me gusta saltar")

michilion = Gato()
```

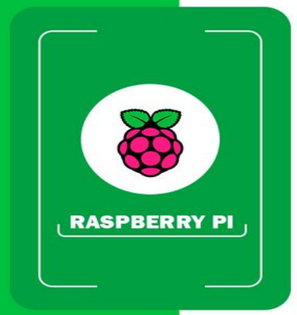
RASPBERRY PI

PYQT5

Es un conjunto de bibliotecas de Python que permite la creación de aplicaciones de escritorio con interfaces gráficas de usuario (GUI).

Sus principales características son:

Diseño de interfaz Gráfica de Usuario (GUI)
Widgets personalizables
Conexión con funciones de Python
Soporte para eventos y señales
Multiplataforma
Optimizado para un rendimiento gráfico eficiente

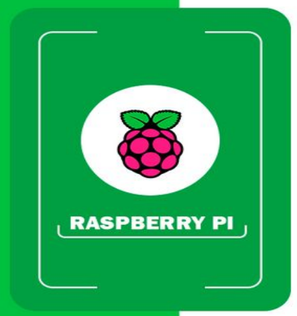


Conceptos previos

1. GUI: Es una forma visual de interactuar con una aplicación. En lugar de utilizar comandos de texto, los usuarios pueden interactuar con la aplicación a través de elementos gráficos como ventanas, botones, menús y cuadros de texto.

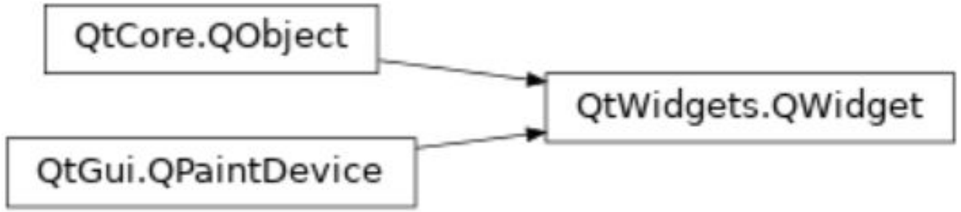
2. Widgets: Los widgets son componentes gráficos de una GUI. Como botones, etiquetas, cuadros de texto, barras de desplazamiento y ventanas.

6. Diseñador de Qt (Qt Designer): Qt Designer es una herramienta visual que te permite diseñar interfaces de usuario gráficas para tus aplicaciones de PyQt5.



CLASES PRINCIPALES

QWidget



Se utiliza para crear ventanas y componentes de la interfaz gráfica de usuario (GUI) en aplicaciones de escritorio. QWidget es una clase base que proporciona funcionalidades comunes a todos los elementos de la interfaz gráfica en PyQt5

```
QWidget(parent=None, flags=Qt.WindowFlags())
```

CLASES PRINCIPALES

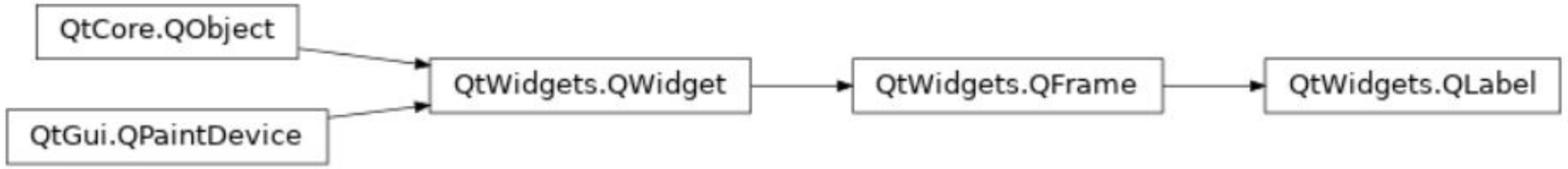
QApplication

se utiliza para crear la instancia principal de la aplicación de PyQt5,iniciar la aplicación, manejo de eventos de la aplicación, manejo de argumentos en la línea de comandos y soporte para aplicaciones multiventana.

```
QApplication(sys.argv)
```

CLASES PRINCIPALES

Texto : QLabel



La clase QLabel proporciona visualización de texto e imágenes en pantalla

QLabel(text: str, parent: QWidget = None, flags: Union[Qt.WindowFlags, Qt.WindowType] = Qt.WindowFlags())

text : Texto a imprimirse
parent: Especifica el padre del widget(ventana), la instancia se vuelve padre secundario
flags: Comportamiento de la etiqueta

CSS

Se utiliza para personalizar la apariencia los elementos de la GUI. CSS se utiliza en PyQt5 para aplicar estilos a los widgets, como colores, fuentes, tamaños de texto, negritas, cursivas, entre otras posibilidades.

QLabel.setStyleSheet(p:str)

font-family	Define la fuente, ejemplo: font-family:Arial,sans-serif;
font-size	Tamaño de fuente en pixeles, ejemplo: font-size:16px
font-weight	controla el grosor de la fuente, usa valores : normal,bold,bolder
font-style	Estilo de la fuente, usa valores como: normal, italic, oblique.
color	Color del texto
background-color	color de fondo del elemento
border	controla el estilo, ancho y alto del borde
padding	Espacio interno alrededor del elemento
margin	Espacio externo alrededor del elemento
text-align	Alinea el texto dentro del elemento, usa valores como : center, left,right o justify
text-decoration	controla la decoración del texto
text-transform	cambia el formato del texto, usa valores como: uppercase, lowercase o capitalize.

BOTONES: QPushButton

Proporciona el uso de un boton

QPushButton(text: str, parent: QWidget = None)

ELEMENTOS PARA GENERAR EVENTOS

Señales: Una señal es un objeto que representa un evento que puede ocurrir en un widget (como un botón, una ventana, un cuadro de texto, etc.) o en otro componente de la interfaz gráfica. Ejemplos de señales comunes incluyen "clicked" (cuando se hace clic en un botón) o "textChanged" (cuando cambia el texto en un cuadro de texto).

BOTONES: QPushButton

Proporciona el uso de un boton

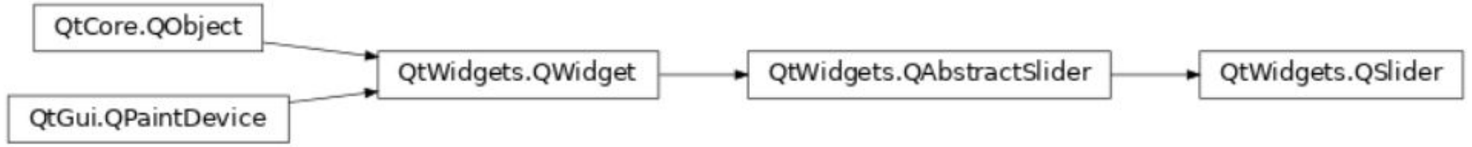
QPushButton(text: str, parent: QWidget = None)

ELEMENTOS PARA GENERAR EVENTOS

Señales: Una señal es un objeto que representa un evento que puede ocurrir en un widget (como un botón, una ventana, un cuadro de texto, etc.) o en otro componente de la interfaz gráfica. Ejemplos de señales comunes incluyen "clicked" (cuando se hace clic en un botón) o "textChanged" (cuando cambia el texto en un cuadro de texto).

Slots: Un slot es una función o método que se puede conectar a una señal. Los slots son funciones que se ejecutarán cuando la señal asociada a ellas sea emitida.

QSLIDER



Proporciona un control deslizante vertical u horizontal

QSlider(orientation: QtCore.Qt.Orientation, parent:QWidget = None)

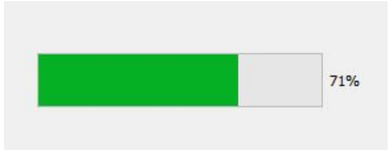
ELEMENTOS PARA GENERAR EVENTOS

Señales: valuechanged, está diseñada para pasar el valor actual del slider (argumento enviado al slot)

Slots: Puede tener cualquier nombre y deberá tener un parámetro para recibir el argumento de la señal.

RASPBERRY PI

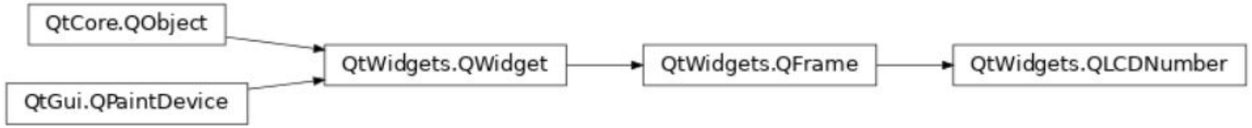
QProgressBar



Proporciona una barra de progreso horizontal o vertical

QProgressBar(parent:QWidget = None)

QLCDNumber



Muestra un número con dígitos similares a los de una pantalla LCD

QLCDNumber(parent:QWidget = None)

RASPBERRY PI