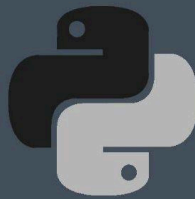


# **DESARROLLO DE INTERFACES**



**José Carlos Mora**

**11/11/2025**

**2ºDAM**

**Tarea 1.2**

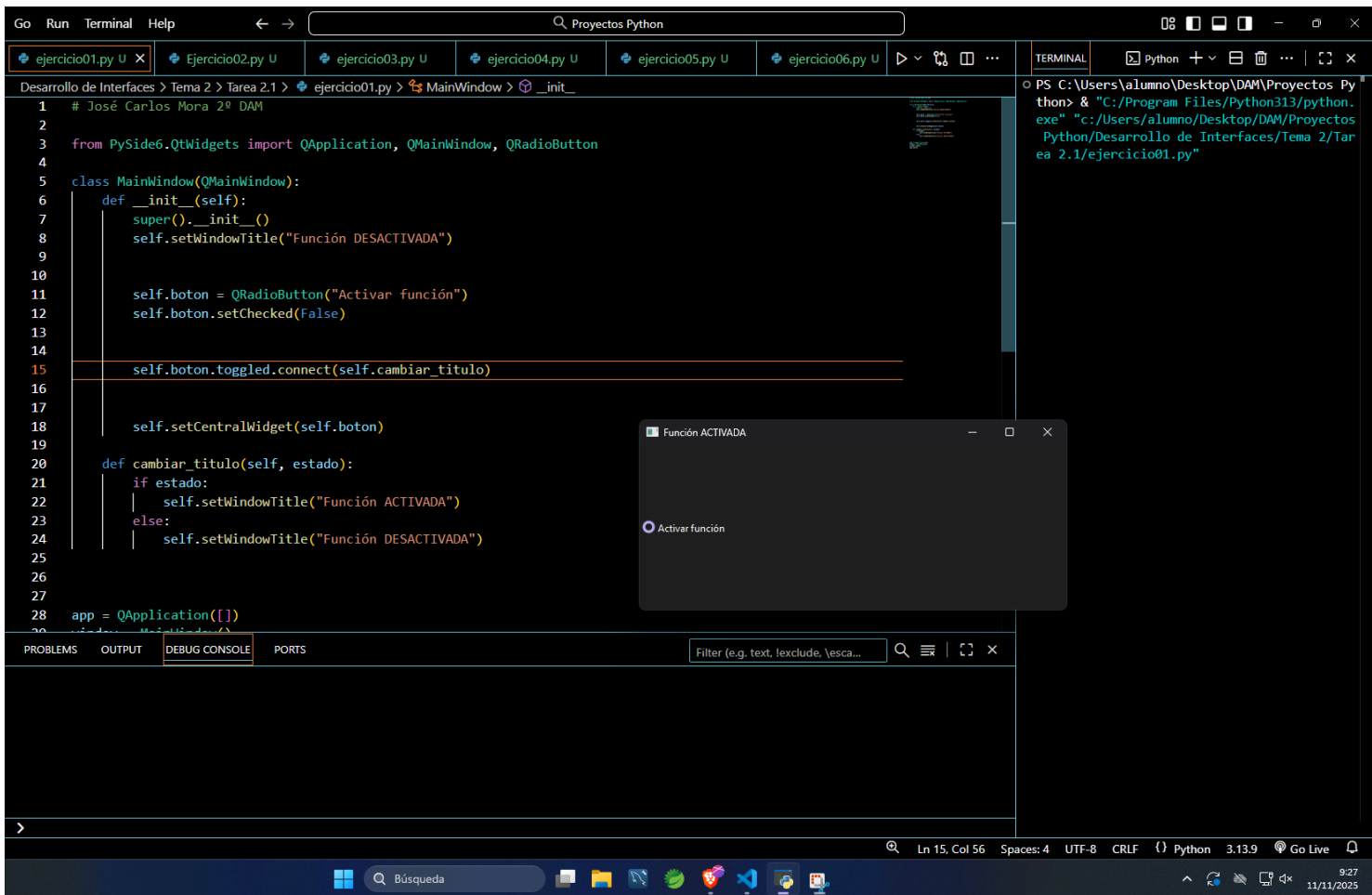
## Índice

QRadioButton.....	3
QTabWidget.....	4
QProgresBar.....	6
QDateTimeEdit.....	8
QSlider.....	9
QDial.....	11

## QRadioButton

- **Descripción general:** es un tipo de widget de interfaz gráfica de usuario que permite al usuario elegir una única opción entre varias opciones ya predefinidas
- **Principales métodos:**
  - **.setChecked(bool):** Es un boolean que establece en qué estado va a estar el botón, es decir, si es True, el botón estaría seleccionado
  - **.isChecked():** Devuelve en qué estado está el botón mediante True o False
  - **.text():** Muestra el texto visible de las opciones
  - **.setText():** Cambia el texto de las opciones
- **Señales (slots) más frecuentes y su uso**
  - **toggled(bool):** Devuelve cuando el estado del botón cambia
  - **pressed():** Devuelve cuando el usuario presiona el botón sin soltarlo
  - **released():** Devuelve cuando el usuario suelta el botón presionado

- **Captura de pantalla de la ejecución del ejercicio propuesto**



- **Webgrafía:**
  - <https://www.tutorialspoint.com/>
  - <https://doc.qt.io/qt-6/qtdesigner-manual.html>
  - <https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html>
  - Apuntes del tema

## QTabWidget

- **Descripción general:** Es un widget que permite organizar la interfaz de un usuario en pestañas, con la que cada pestaña contiene un widget independiente
- **Principales métodos:**
  - **addTab(widget, "Título"):** Añade una nueva pestaña indicando el título que tendrá y el widget

- **.setCurrentIndex(int):** Cambia la pestaña que está visible actualmente según el número que le indiques
- **.count():** Devuelve el número total de pestañas que hay
- **.removeTab():** Elimina la pestaña en la posición que indiques
- **.currentIndex():** Devuelve la posición de la pestaña en la que se encuentra actualmente
- **Señales (slots) más frecuentes y su uso**
  - **currentChanged(int):** Devuelve al cambiar de pestaña
  - **tabCloseRequested(int):** Devuelve cuando el usuario pide cerrar una pestaña
  - **tabBarClicked(int):** Devuelve cuando se hace click en una pestaña
  - **tabBarDoubleClicked(int):** Devuelve cuando se hace doble click en una pestaña
- **Captura de pantalla de la ejecución del ejercicio propuesto**

```

1 # José Carlos Mora 2º DAM
2
3 from PySide6.QtWidgets import QApplication, QMainWindow, QTabWidget, QLabel
4
5 class MainWindow(QMainWindow):
6     def __init__(self):
7         super().__init__()
8         self.setWindowTitle("Pestaña 1")
9
10
11         self.tabs = QTabWidget()
12
13
14         self.tabs.addTab(QLabel("Bienvenido"), "Pestaña 1")
15         self.tabs.addTab(QLabel("Segunda pestaña"), "Pestaña 2")
16         self.tabs.addTab(QLabel("Tercera pestaña"), "Pestaña 3")
17
18
19         self.titulos = ["Pestaña 1", "Pestaña 2", "Pestaña 3"]
20
21
22         self.tabs.currentChanged.connect(self.cambiar_titulo)
23
24
25         self.setCentralWidget(self.tabs)
26
27     def cambiar_titulo(self, indice):
28         print(indice)
29         self.setWindowTitle(self.titulos[indice])
30
31
32
33 app = QApplication([])
34 window = MainWindow()
35 window.show()
36 app.exec()
37

```

Terminal Output:

```

PS C:\Users\alumno\Desktop\DAM\Proyectos Python> python ejercicio02.py
Segunda pestaña

```

- **Webgrafía:**

- <https://www.tutorialspoint.com/>
- <https://doc.qt.io/qt-6/qtdesigner-manual.html>
- <https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html>
- Apuntes del tema

## **QProgresBar**

- **Descripción general:** Permite mostrar el progreso de una tarea en forma de barra

- **Principales métodos:**

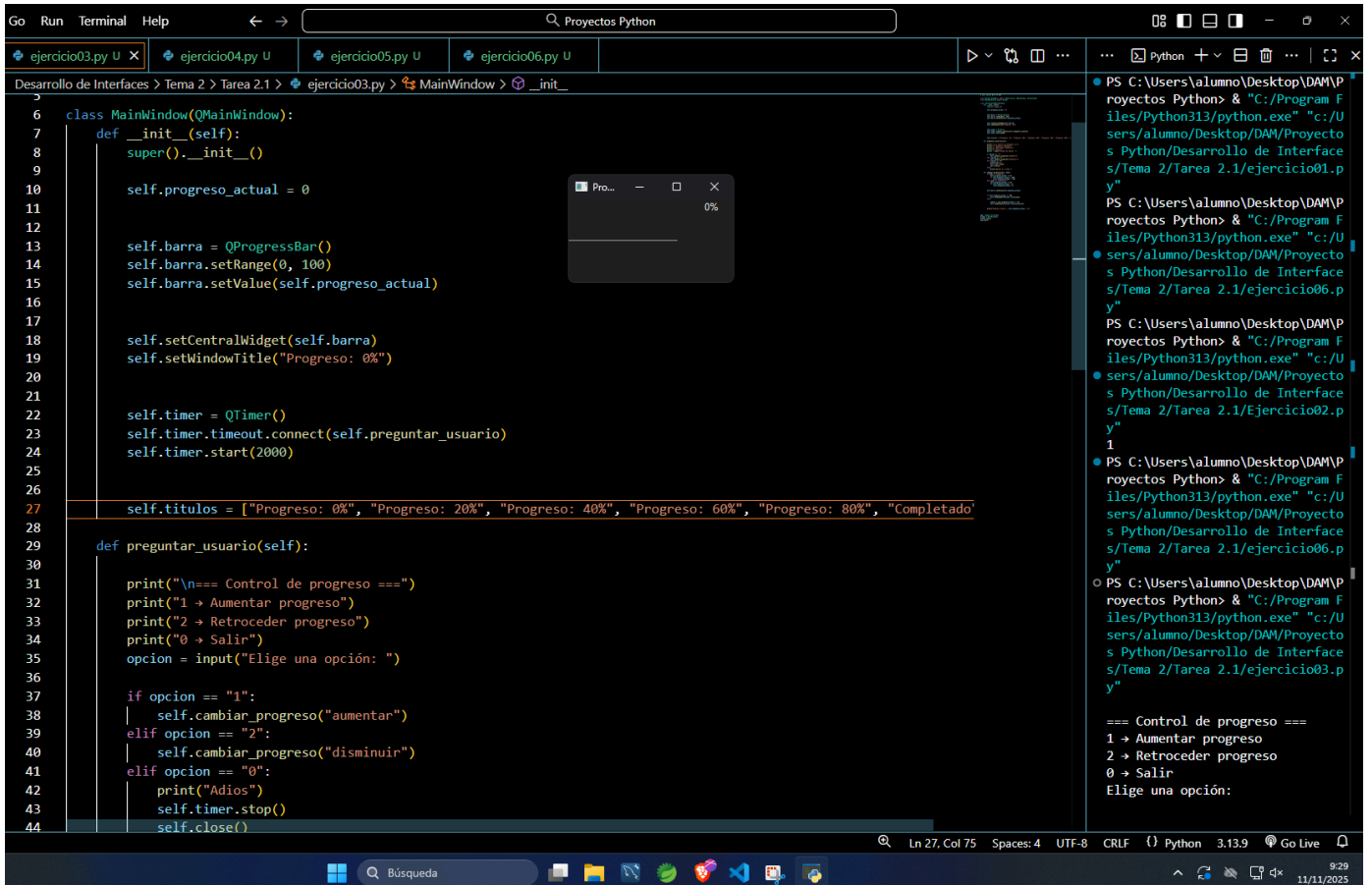
- **.setValue(int):** Define el valor actual de la barra
- **.setRange(min, max):** Define los límites de la barra
- **.reset():** Reinicia la barra
- **.setMinimum(int):** Fija el valor mínimo de la barra
- **.setTextVisible(bool):** Muestra u oculta el texto que tenga que la barra
- **.setStyleSheet(str):** Sirve para personalizar el diseño de la barra

- **Señales (slots) más frecuentes y su uso:**

No tiene señales propias, pero reacciona a la de otros widgets como por ejemplo:

- QSlider
- QDial

- Captura de pantalla de la ejecución del ejercicio propuesto



- Webgrafía:

- <https://www.tutorialspoint.com/>
- <https://doc.qt.io/qt-6/qt designer-manual.html>
- <https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html>
- Apuntes del tema

## QDateTimeEdit

- **Descripción general:** Sirve para seleccionar la fecha y hora, donde se puede editar
- **Principales métodos:**
  - **.setDateTime(QDateTime):** Establece la fecha y hora
  - **.dateTime():** Devuelve la fecha y hora actual
  - **.setDisplayFormat(str):** Cambia el formato de la fecha
  - **.setMinimumDate(QDate):** Establece la fecha mínima
  - **.clear():** Borra todo el contenido de la fecha y hora
  - **.setCalendarPopup(bool):** Sirve para mostrar un calendario emergente
- **Señales (slots) más frecuentes y su uso:**
  - **dateChanged(QDate):** Cambia la fecha
  - **timeChanged(QTime):** Cambia la hora
  - **dateTimeChanged(QDateTime):** Cambia tanto fecha como hora
  - **editingFinished():** Devuelve cuando termine la edición manual



- **Captura de pantalla de la ejecución del ejercicio propuesto**

```

1  # José Carlos Mora 2º DAM
2
3  from PySide6.QtWidgets import QApplication, QMainWindow, QDateTimeEdit
4  from PySide6.QtCore import QDateTime
5
6  class MainWindow(QMainWindow):
7      def __init__(self):
8          super().__init__()
9
10         self.dt_edit = QDateTimeEdit()
11         self.dt_edit.setDate(QDateTime.currentDateTime())
12
13         self.dt_edit.setDisplayFormat("dddd, d 'de' MMMM 'de' yyyy hh:mm")
14
15         self.dt_edit.dateTimeChanged.connect(self.mostrar_fecha)
16
17         self.setCentralWidget(self.dt_edit)
18
19         self.mostrar_fecha(self.dt_edit.dateTime())
20
21     def mostrar_fecha(self, fecha_hora):
22         self.setWindowTitle(fecha_hora.toString("dddd, d 'de' MMMM 'de' yyyy hh:mm"))
23         print("Fecha elegida:", fecha_hora.toString("dddd, d 'de' MMMM 'de' yyyy hh:mm"))
24
25 app = QApplication([])
26 ventana = MainWindow()
27 ventana.show()
28 app.exec()
29

```

Output: miércoles 12 de noviembre de 2025 09:30

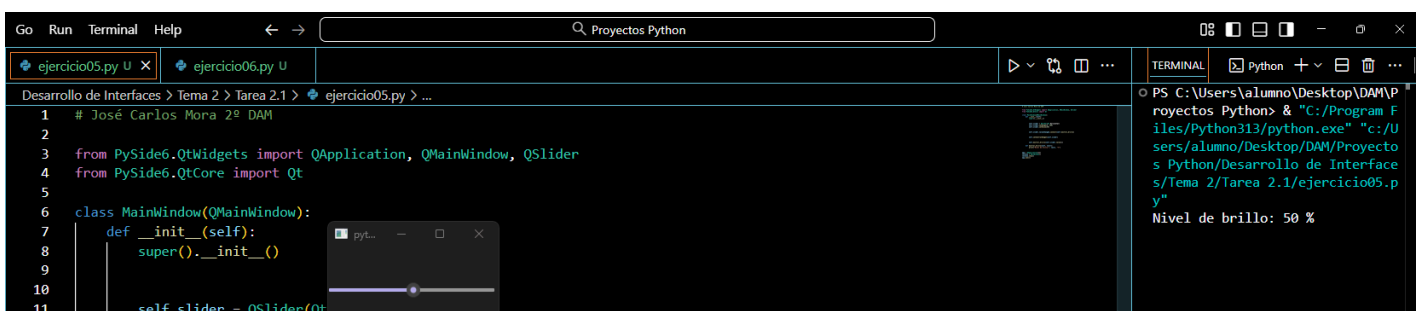
- **Webgrafía:**

- <https://www.tutorialspoint.com/>
- <https://doc.qt.io/qt-6/qtdesigner-manual.html>
- <https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html>
- Apuntes del tema

## QSlider

- **Descripción general:** Sirve para crear un control deslizante o vertical que permite seleccionar un valor numérico dentro de un rango
- **Principales métodos:**
  - **.setMinimun():** Define el valor mínimo

- **.setMaximun():** Define el valor máximo
- **.setValue():** Fija el valor actual
- **.Value:** Devuelve el valor actual
- **.setOrientation(Qt.Orientation):** Define la orientación
- **.setTracking(bool):** Devuelve señales constantemente mientras se mueve, siempre y cuando sea true
- **.setTickPosition(QSlider.TicksBelow):** Define la posición de las marcas
- **Señales (slots) más frecuentes y su uso:**
  - **valueChanged(int):** Se emite cuando cambia el valor
  - **sliderPressed():** Devuelve cuando se presiona el control
  - **sliderReleased():** Devuelve cuando se suelta el control
  - **sliderMoved(int):** Devuelve cuando se arrastra el control
- **Captura de pantalla de la ejecución del ejercicio propuesto**



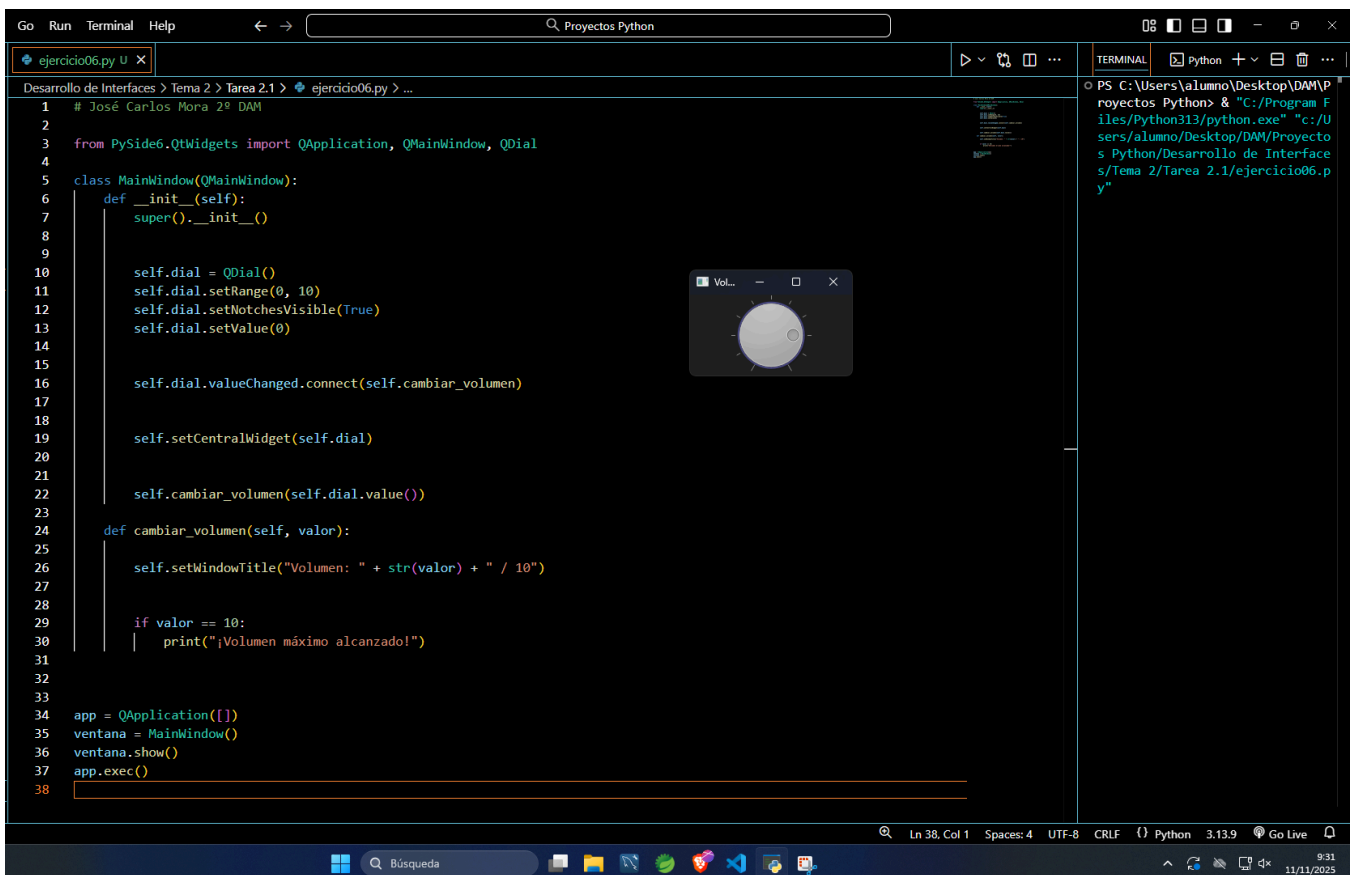
- **Webgrafía:**

- <https://www.tutorialspoint.com/>
- <https://doc.qt.io/qt-6/qtdesigner-manual.html>
- <https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html>
- Apuntes del tema

## **QDial**

- **Descripción general:** Sirve para hacer un control en forma de rueda donde el usuario puede elegir un número dentro de un rango
- **Principales métodos:**
  - **.setNotchesVisible(True):** Muestra las muescas o marcas
  - **.setRange(min, max):** Define los valores mínimo y máximo
  - **.setValue():** Devuelve el valor actual

- **.setEnabled(bool):** Habilita o deshabilita el control
- **.setInvertedAppearance(bool):** Invierte la dirección visual
- **Señales (slots) más frecuentes y su uso:**
  - **valueChanged(int):** Se emite cuando cambia el valor
  - **sliderPressed():** Devuelve cuando se presiona el control
  - **sliderReleased():** Devuelve cuando se suelta el control
  - **sliderMoved(int):** Devuelve cuando se arrastra el control
- **Captura de pantalla de la ejecución del ejercicio propuesto**



```

1 # José Carlos Mora 2º DAM
2
3 from PySide6.QtWidgets import QApplication, QMainWindow, QDial
4
5 class MainWindow(QMainWindow):
6     def __init__(self):
7         super().__init__()
8
9
10        self.dial = QDial()
11        self.dial.setRange(0, 10)
12        self.dial.setNotchesVisible(True)
13        self.dial.setValue(0)
14
15
16        self.dial.valueChanged.connect(self.cambiar_volumen)
17
18
19        self.setCentralWidget(self.dial)
20
21
22        self.cambiar_volumen(self.dial.value())
23
24    def cambiar_volumen(self, valor):
25
26        self.setWindowTitle("Volumen: " + str(valor) + " / 10")
27
28
29        if valor == 10:
30            print("¡Volumen máximo alcanzado!")
31
32
33
34 app = QApplication([])
35 ventana = MainWindow()
36 ventana.show()
37 app.exec()
38

```

PS C:\Users\alumno\Desktop\DAW\Proyectos Python> & "C:/Program Files/Python313/python.exe" "c:/Users/alumno/Desktop/DAW/Proyectos Python/Desarrollo de Interfaces/Tarea 2.1/ejercicio06.py"

- <https://www.tutorialspoint.com/>

- <https://doc.qt.io/qt-6/qtdesigner-manual.html>
- <https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html>
- Apuntes del tema