Introducción a Python

José Juan Sánchez Hernández - 19 de Enero de 2019

Tabla de contenidos

1. Tipos de datos

- Tipos booleanos
  - `bool`
- Tipos de datos numéricos
  - `int`
  - `float`
  - `complex`
- Cadenas de caracteres
  - `str`
- Tipos de secuencias
  - `list`
  - `tuple`
  - `range`
- Tipos de secuencias binarias
  - `bytes`
  - `bytearray`
  - `memoryview`
- Tipos mapa (Diccionario)
  - `dict`
- Conjuntos
  - `set`
  - `frozenset`

Referencia:

- Tipos de datos

Funciones de interés:

- `type` devuelve el tipo de dato de una variable.
- `dir` devuelve un listado de los métodos y atributos del objeto pasado como parámetro.
- `help` muestra la documentación del objeto o método pasado como parámetro.

2. Operaciones de Entrada/Salida

2.1. Mostrar por pantalla. `print()`

Esta función nos permite mostar un mensaje por pantalla. Podemos utilizar **comillas dobles o comillas simples**.

```
print("Texto")
print('Texto')
```

Podemos mostrar el contenido de una variable.

```
print(variable)
```

Podemos combinar una cadena de texto con variables.

```
print("Texto: ", variable)
```

A partir de Pyton 3.6 podemos utilizar las **cadenas f**.

```
print(f",Texto {variable}")
```

Por defecto, imprime un salto de línea, pero podemos indicar qué no haga un salto de línea indicando cuál debe ser el último carácter de la cadena que se debe imprimir. Si utilizamos `end=""`, entonces no escribirá el salto de línea.

```
print("Texto", end="")
```

2.2. Leer por teclado. `input()`

```
numero = input("Introduce un número: ")
```

Los valores que leemos desde el teclado se almacenarán como una cadena de caracteres, pero es posible convertirlos al tipo de datos que sea necesario con las siguientes funciones:

- `int()`
- `float()`
- `complex()`
- `bool()`

No se recomienda el uso de la función `eval` para convertir el contenido que leemos con `input` porque puede ocasionar problemas de seguridad.

Se recomienda la lectura del artículo: Eval really is dangerous.

2.3. Actividades

Actividad 1

Escriba un programa en Python que convierta de grados Celsius a grados Farenheit. El programa tiene que pedir por teclado el valor de una temperatura en grados Celsius y convertirla a grados Farenheit. Tenga en cuenta que la fórmula para convertir de grados Celsius a grados Farenheit es la siguiente:

$$farenheit = 1.8 \cdot celsius + 32$$

Solución

```
# Paso 1. Leemos una temperatura por teclado
celsius = float(input('Introduce una temperatura en grados Celsius: '))

# Paso 2. Convertimos de grados Celsius a Farenheit
farenheit = (1.8 * celsius) + 32

# Paso 3. Mostramos el resultado por pantalla
print(f'La temperatura en grados Farenheit es: {farenheit}')
```

Ver solución en GitHub

Actividad 2

Escriba un programa en Python que calcule el área de un círculo. El programa tiene que pedir por teclado el valor del radio del círculo. Tenga en en cuenta que la fórmula para calcular el área de un círculo es la siguiente:

$$area = \pi \cdot r^2$$

Puede importar el módulo `math` para hacer uso de la constante `pi`. Consulte la sección Importar módulos.

Solución

```python
# Paso 1. Importamos el módulo de funciones matemáticas
import math

# Paso 2. Leemos el valor del radio por teclado
radio = float(input('Introduce el valor del radio: '))

# Paso 3. Calculamos el área
area = math.pi * radio * radio
#area = math.pi * (radio**2)
#area = math.pi * math.pow(radio, 2)

# Paso 4. Mostramos el resultado
print(f'El área es {area}')
```

Ver solución en GitHub

Actividad 3

Escriba un programa en Python que calcula la hipotenusa de un triángulo rectángulo. El programa tiene que pedir por teclado el valor de los dos catetos. Tenga en cuenta que la fórmula para calcular la hipotenusa de un triángulo rectángulo es la siguiente:

$$hipotenusa = \sqrt{a^2 + b^2}$$

Para poder realizar la raíz cuadrada tendrá que utilizar la función `sqrt` del módulo `math`.

Solución

```python
import math

# Paso 1. Leemos por teclado el valor de los dos catetos
cateto1 = float(input("Introduce el valor del cateto 1: "))
cateto2 = float(input("Introduce el valor del cateto 1: "))

# Paso 2. Calculamos la hipotenusa
hipotenusa = math.sqrt((cateto1*cateto1) + (cateto2*cateto2))
#hipotenusa = math.sqrt((cateto1**2) + (cateto2**2))
#hipotenusa = math.sqrt(math.pow(cateto1,2) + math.pow(cateto2,2))

# Paso 3. Mostramos el resultado
print(f"El valor de la hipotenusa es {hipotenusa}")
```

Ver solución en GitHub

Actividad 4

Escriba un programa en Python que calcule el índice de masa corporal de una persona (IMC). El

programa tiene que pedir por teclado el valor del peso en kilogramos y la altura en metros. Tenga en cuenta que la fórmula para calcular el índice de masa corporal de una persona es la siguiente:

$$imc = \frac{peso}{altura^2}$$

Solución

```
peso = float(input("Introduce tu peso en kilogramos: "))
altura = float(input("Introduce tu altura en metros: "))

imc = peso / (altura*altura)

print(f"El valor del IMC es {imc}")
```

Ver solución en GitHub

Actividad 5

Escriba un programa en Python que pida por teclado una cantidad en euros y la convierta a dólares estadounidenses (USD) y bitcoins (BTC). Tenga en cuenta que los valores de las monedas a día de hoy son los siguientes:

- 1 euro equivale a 1,06 dólar estadounidense (USD).
- 1 euro equivale a 0,000034 bitcoins (BTC).

Solución

```
euros = float(input("Introduce una cantidad en euros: "))

usd = euros * 1.06
btc = euros * 0.000034

print(f"{euros} Euros  equivalen a: {usd} USD")
print(f"{euros} Euros equivalen a: {btc} Bitcoin")
```

Ver solución en GitHub

3. Importar módulos

3.1. Importar un módulo completo

Ejemplo 1

```
import math ①
print(math.pi) ②
```

① Importamos el módulo `math`. Al importar el módulo `math` podemos acceder a todas las variables, constantes y funciones que están definidas en él.

② Hacemos uso de la constante `pi` que está definida en el módulo `math`. Como el módulo ha sido importado con la instrucción `import math` debemos indicar el nombre del módulo seguido de un punto para acceder a sus variables, constantes y funciones.

3.2. Importar un elemento del módulo

Ejemplo 2

```
from math import pi ①
print(pi) ②
```

① Importamos **únicamente** la constante `pi` del módulo `math`. El resto de contenido del módulo `math` no estará disponible porque no ha sido importado.

② Hacemos uso de la constante `pi` que hemos importado en el paso anterior. A diferencia del ejemplo anterior, en este caso no es necesario indicar el nombre del módulo delante de la constante.

Creación de alias al importar un módulo

Es posible crear un alias al importar un módulo en Python.

Ejemplo 1

```
import math as m
print(m.pi)
```

Ejemplo 2

```
from math import pi as numero_pi
print(numero_pi)
```

4. Sentencias condicionales. `if - elif - else`

La sentencia condicional `if` se puede utilizar de tres formas distintas.

Sintaxis #1

```
if condición:
    bloque1
```

Ejemplo:

```
numero = int(input("Intoduce un número: "))
if numero > 0:
    print("El número es mayor que 0")
```

Sintaxis #2

```
if condición:
    bloque1
else:
    bloque2
```

La palabra reservada `else` es opcional y solo puede aparecer cero o una vez.

Ejemplo:

```
numero = int(input("Intoduce un número: "))
if numero > 0:
    print("El número es mayor que 0")
else:
    print("El número es menor o igual que 0")
```

Sintaxis #3

```
if condición:
    bloque1
elif condición:
    bloque2
else:
    bloque3
```

La palabra reservada `elif` es opcional y puede aparecer cero o muchas veces.

Ejemplo:

```
numero = int(input("Intoduce un número: "))
if numero > 0:
    print("El número es mayor que 0")
elif numero < 0:
    print("El número es menor que 0")
else:
    print("El número es 0")
```

4.1. Actividades

Actividad 1

Escriba un programa en Python que lea un número entero por teclado y muestre un mensaje indicando si el número es par o impar. Tenga en cuenta que un número es par si el resto de la división del número entre 2 es igual a 0. Para calcular el resto de una división puede utilizar el operador módulo (`%`).

Solución

```
# Paso 1. Leemos un número entero por teclado
numero = int(input("Introduce un número: "))

# Paso 2. Comprobamos si el número es par o impar
if numero % 2 == 0:
    print(f"El número {numero} es PAR")
else:
    print(f"El número {numero} es IMPAR")
```

Ver solución en GitHub

Actividad 2

Escribe un programa en Python que lea por teclado un número real entre 1 y 10, simulando una nota numérica, y muestre un mensaje indicando la calificación obtenida teniendo en cuenta los siguientes rangos:

- Insuficiente: [0, 5)
- Suficiente: [5, 6)
- Bien: [6, 7)
- Notable: [7, 9)
- Sobresaliente: [9, 10]

Si el número introducido no está en ninguno de los rangos anteriores debe mostrar un mensaje de error indicando que la nota no es válida.

Solución

```
# Paso 1. Leemos la nota por teclado
nota = float(input('Introduce una nota entre 0 y 10: '))

# Paso 2. Comprobamos la nota
if nota >= 0 and nota < 5:
    print("Insuficiente")
elif nota >= 5 and nota < 6:
    print("Suficiente")
elif nota >= 6 and nota < 7:
    print("Bien")
elif nota >= 7 and nota < 9:
    print("Notable")
elif nota >= 9 and nota < 10:
    print("Sobresaliente")
else:
    print("La nota no es válida")
```

Ver solución en GitHub

Actividad 3

Escribe un programa en Python que simule el comportamiento de lanzar una moneda al aire y muestre un mensaje indicando si ha salido cara o cruz.

Para generar un número entero aleatorio puede utilizar la función `randint` del módulo `random`. Esta función genera un número entero aleatorio comprendido entre los dos números que recibe como parámetros.

Por ejemplo, para generar un número entero entre 1 y 10 podríamos utilizar las siguientes opciones.

Opción 1

```
import random
n = random.randint(1, 10)
print(n)
```

Opción 2

```
from random import randint
n = randint(1, 10)
print(n)
```

Solución

```python
import random

# Paso 1. Generamos un número aleatorio entre 0 y 1
numero = random.randint(0, 1)

# Paso 2. Comprobamos si ha salido cara o cruz
if numero == 0:
    print("CARA")
else:
    print("CRUZ")
```

Ver solución en GitHub

Actividad 4

Escriba un programa en Python que muestre el siguiente mensaje por pantalla para preguntarle al usuario si desea seleccionar cara o cruz.

```
0 - Cara
1 - Cruz
Introduce una opción (0 o 1):
```

Después de leer la opción seleccionada por el usuario el programa generará un número aleatorio para simular el lanzamiento de una moneda y mostrará un mensaje indicando si el usuario ha ganado o ha perdido dependiendo del resultado.

Solución

```python
import random

# Paso 1. Leemos por teclado la opción del usuario
print("0 - Cara")
print("1 - Cruz")
opcion = int(input("Introduce una opción (0 o 1):"))

# Paso 2. Generamos un número aleatorio
moneda = random.randint(0, 1)

# Paso 3. Mostramos el valor de la moneda que ha sacado la máquina
if moneda == 0:
    print("Ha salido CARA")
else:
    print("Ha salido CRUZ")

# Paso 4. Comparamos la opción del usuario con el valor generado
if opcion == moneda:
    print("Enhorabuena, has ganado!!")
else:
    print("Lo siento, has pedido...")
```

Ver solución en GitHub

Actividad 5

Escriba un programa en Python que simule el juego de piedra, papel o tijera. En primer lugar el programa tendrá que mostrar un mensaje por pantalla al usuario para preguntarle qué opción desea elegir. Por ejemplo:

```
1. Piedra
2. Papel
3. Tijera
Seleccione una opción (1, 2 o 3):
```

Después de leer la opción seleccionada por el usuario el programa generará un número aleatorio para simular una jugada y mostrará un mensaje indicando si el usuario ha ganado o ha perdido dependiendo del resultado.

Tenga en cuenta que:

- La piedra gana a la tijera pero pierde contra el papel.
- El papel gana a la piedra pero pierde contra la tijera.
- La tijera gana al papel pero pierde contra la piedra.

Solución

```python
import random

# Paso 1. Leemos por teclado la opción seleccionada por el usuario
print("1. Piedra")
print("2. Papel")
print("3. Tijera")
jugada_usuario = int(input("Seleccione una opción (1, 2 o 3): "))

# Paso 2. Generamos un número entero aleatorio entre 1 y 3
jugada_ordenador = random.randint(1, 3)

# Paso 3. Mostramos la jugada del ordenador
if jugada_ordenador == 1:
    print("El ordenador ha sacado: PIEDRA")
elif jugada_ordenador == 2:
    print("El ordenador ha sacado: PAPEL")
else:
    print("El ordenador ha sacado: TIJERA")

# Paso 4. Comparamos la jugada del usuario con la del ordenador
if jugada_usuario == jugada_ordenador:
    print("Hay un EMPATE")
elif jugada_usuario == 1 and jugada_ordenador == 3:
    print("Gana el USUARIO")
elif jugada_usuario == 2 and jugada_ordenador == 1:
    print("Gana el USUARIO")
elif jugada_usuario == 3 and jugada_ordenador == 2:
    print("Gana el USUARIO")
else:
    print("Gana el ORDENADOR")
```

Ver solución en GitHub

5. Bucles

5.1. Actividades

Actividad 1

Escriba un programa en Python que muestre una lista de números del 1 al 10. Resuelva el ejercicio de dos formas distintas, utilizando los bucles `for` y `while`. Cuando utilice el bucle for puede hacer uso de la función `range`.

Solución

```
# Solución con bucle for
print("Bucle for")
for numero in range(1, 11):
    print(numero, end = " ")

# Solución con bucle while
print("\nBucle while")
numero = 1
while numero <= 10:
    print(numero, end = " ")
    numero = numero + 1
```

Ver solución en GitHub

Actividad 2

Escriba un programa en Python que muestre una lista de números del 10 al 1. Resuelva el ejercicio de dos formas distintas, utilizando los bucles `for` y `while`. Cuando utilice el bucle for puede hacer uso de la función `range`.

Solución

```
# Solución con bucle for
print("Bucle for")
for numero in range(10, 0, -1):
    print(numero, end = " ")

# Solución con bucle while
print("\nBucle while")
numero = 10
while numero >= 1:
    print(numero, end = " ")
    numero = numero - 1
```

Ver solución en GitHub

Actividad 3

Escriba un programa en Python que muestre los números pares que hay entre 0 y 10. Resuelva el ejercicio utilizando un bucle `for`.

Solución

```
for numero in range(0, 11, 2):
    print(numero, end = " ")
```

Ver solución en GitHub

Actividad 4

Escriba un programa en Python que muestre los números impares que hay entre 1 y 11. Resuelva el ejercicio utilizando un bucle `for`.

Solución

```
for numero in range(1, 12, 2):
    print(numero, end = " ")
```

Ver solución en GitHub

Actividad 5

Escriba un programa en Python que lea por teclado un número comprendido entre 1 y 10. Resuelva el ejercicio utilizando un bucle `while`. ¿Es posible resolver este ejercicio con un bucle `for`?

Solución

```
numero = float(input("Introduce un número entre 1 y 10: "))
while numero < 1 or numero > 10:
    print("Error. El número no es correcto")
    numero = float(input("Introduce un número entre 1 y 10: "))
```

Ver solución en GitHub

Actividad 6

Escriba un programa en Python que realice las siguientes operaciones:

- Leer por teclado un número comprendido entre 1 y 10.
- Una vez que ha leído el número tiene que mostrar su tabla de multiplicar.

Solución

```
# Paso 1. Leer por teclado un número comprendido entre 1 y 10
numero = int(input("Introduce un ńumero entre 1 y 10: "))
while numero < 1 or numero > 10:
    print("Error. El número no es válido.")
    numero = int(input("Introduce un ńumero entre 1 y 10: "))

# Paso 2. Mostramos la tabla de multiplicar
for valor in range(1,11,1):
    print(f"{numero}x{valor}={numero*valor}")
```

Ver solución en GitHub

Actividad 7

Escriba un programa en Python que realice las siguientes operaciones:

- Leer por teclado un número comprendido entre 1 y 10.
- Una vez que ha leído el número tiene que mostrar su tabla de multiplicar.
- Después de mostrar la tabla de multiplicar tiene que preguntar al usuario si desea introducir otro número o no. Si el usuario selecciona que quiere continuar el programa tendrá que volver a ejecutarse y repetir los mismos pasos. Si el usuario indica que no quiere continuar el programa finaliza.

Solución

```
# Inicializamos la variable respuesta
respuesta = "s"

while respuesta == "s" or respuesta == "S":

    # Paso 1. Leer por teclado un número comprendido entre 1 y 10
    numero = int(input("Introduce un ńumero entre 1 y 10: "))
    while numero < 1 or numero > 10:
        print("Error. El número no es válido.")
        numero = int(input("Introduce un ńumero entre 1 y 10: "))

    # Paso 2. Mostramos la tabla de multiplicar
    for valor in range(1,11,1):
        print(f"{numero}x{valor}={numero*valor}")

    # Paso 3. Preguntar al usuario si desea volver a ejecutar el código
    respuesta = input("¿Desea volver a ejecutar el programa (s/n)? ")
```

Ver solución en GitHub

Actividad 8

Escriba un programa en Python que muestre todas las tablas de multiplicar de los números comprendidos del 1 al 10.

Solución

```
for numero in range(1,11):
    print(f"Tabla del {numero}")

    for valor in range(1, 11):
        print(f"{numero} x {valor} = {numero*valor}")

    print("\n")
```

Ver solución en GitHub

6. Funciones

6.1. Actividades

Actividad 1

Escriba un programa en Python que haga uso de una función llamada `saludar` que cumpla los siguientes requisitos:

- *Entrada*: No tiene parámetros de entrada.
- *Salida*: No tiene parámetros de salida.
- *Funcionalidad*: Imprime por pantalla el texto "Hola mundo".

Solución

```
# Definimos la función saludar
def saludar():
    print("Hola mundo")

#--------------------------------------------------
# Programa Principal
# Utilizamos la función saludar
saludar()
```

Ver solución en GitHub

Actividad 2

Escriba un programa en Python que haga uso de una función llamada `saludar` que cumpla los siguientes requisitos:

- *Entrada*: Tiene 1 parámetro de entrada, que será una cadena de texto con el nombre de una persona.
- *Salida*: No tiene parámetros de salida.
- *Funcionalidad*: Imprime por pantalla un mensaje saludando al nombre de la persona que se haya pasado como parámetro de entrada. Por ejemplo, si el parámetro de entrada es "Pepe", la función mostrará por pantalla: "¡Hola Pepe!".

Solución

```python
# Definimos la función saludar
def saludar(nombre):
    print(f"¡Hola {nombre}!")

#-----------------------------------------------
# Programa Principal
# Utilizamos la función saludar
nombre = input("Dime tu nombre: ")
saludar(nombre)
```

Ver solución en GitHub

Actividad 3

Escriba un programa en Python que haga uso de una función llamada `saludar` que cumpla los siguientes requisitos:

- *Entrada*: Tiene 3 parámetros de entrada, que serán 3 cadenas de texto.
  - Nombre,
  - Primer apellido,
  - Segundo apellido
- *Salida*: No tiene parámetros de salida.
- *Funcionalidad*: Imprime por pantalla un mensaje saludando al nombre de la persona que se haya pasado como parámetro de entrada. Por ejemplo, si los parámetros de entrada son "Pepe López Martínez", la función mostrará por pantalla: "¡Hola Pepe López Martínez!".

Solución

```python
# Definimos la función saludar
def saludar(nombre, apellido1, apellido2):
    print(f"¡Hola {nombre} {apellido1} {apellido2}!")

#-----------------------------------------------
# Programa Principal
# Utilizamos la función saludar
nombre = input("Dime tu nombre: ")
apellido1 = input("Dime tu primer apellido: ")
apellido2 = input("Dime tu segundo apellido: ")

saludar(nombre, apellido1, apellido2)
```

Ver solución en GitHub

Actividad 4

Escriba un programa en Python que haga uso de una función llamada `calcular_area_circulo` que

cumpla los siguientes requisitos:

- *Entrada*: Tiene 1 parámetro de entrada, que será un número real con el valor del radio del círculo del que queremos calcular su área.
- *Salida*: Tiene 1 parámetro de salda, que será un número real con el valor del área del círculo.
- *Funcionalidad*: Calcula el área de un círculo. Tenga en cuenta que el valor del radio no se pide dentro de la función, sino que se recibe como parámetro de entrada.

Solución

```
import math

# Definimos la función
def calcular_area_circulo(radio):
    area = math.pi * radio * radio
    return area

#------------------------------------------------------
# Programa Principal
radio = float(input("Introduce el radio del círculo: "))
area = calcular_area_circulo(radio)
print(f"El área del círculo es {area}")
```

Ver solución en GitHub

Actividad 5

Escriba un programa en Python que haga uso de una función llamada `calcular_area_triangulo` que cumpla los siguientes requisitos:

- *Entrada*: Tiene 2 parámetros de entrada, que serán dos números reales con el valor de la base y la altura del triángulo que queremos calcular su área.
- *Salida*: Tiene 1 parámetro de salida, que será un número real con el valor del área del triángulo.
- *Funcionalidad*: Calcula el área de un triángulo. Tenga en cuenta que el valor de la base y la altura no se piden dentro de la función, sino que se reciben como parámetros de entrada.

Solución

```python
# Definimos la función
def calcular_area_triangulo(base, altura):
    area = (base * altura) / 2
    return area

#-----------------------------------------------

# Programa Principal (Main)
base = float(input("Introduce el valor de la base: "))
altura = float(input("Introduce el valor de la altura: "))
area = calcular_area_triangulo(base, altura)
print(f"El área del triángulo es {area}")
```

Ver solución en GitHub

Actividad 6

Escriba un programa en Python que haga uso de una función llamada `calcular_area_rectangulo` que cumpla los siguientes requisitos:

- *Entrada*: Tiene 2 parámetros de entrada, que serán dos números reales con el valor de la base y la altura del rectángulo que queremos calcular su área.
- *Salida*: Tiene 1 parámetro de salida, que será un número real con el valor del área del rectángulo.
- *Funcionalidad*: Calcula el área de un rectángulo. Tenga en cuenta que el valor de la base y la altura no se piden dentro de la función, sino que se reciben como parámetros de entrada..

Solución

```python
# Definimos la función
def calcular_area_rectangulo(base, altura):
    area = base * altura
    return area

#-----------------------------------------------

# Programa Principal (Main)
base = float(input("Introduce el valor de la base: "))
altura = float(input("Introduce el valor de la altura: "))
area = calcular_area_rectangulo(base, altura)
print(f"El área del rectángulo es {area}")
```

Ver solución en GitHub

Actividad 7

Escriba un programa en Python que muestre un menú que permita al usuario seleccionar qué

operación desea realizar. Las operaciones que puede realizar serán calcular el área de un círculo, un triángulo o un rectángulo. El menú que se le muestra al usuario será similar al siguiente:

```
1. Calcular el área de un círculo
2. Calcular el área de un triángulo
3. Calcular el área de un rectángulo
4. Salir
Introduce una opción (1-4):
```

El programa se estará ejecutando de forma indefinida hasta que el usuario seleccione la opción 4.

Tendrá que diseñar las siguientes funciones:

- `mostrar_menu`
  - *Descripción*: Muestra el menú por pantalla con todas las opciones disponibles.
- `principal`
  - *Descripción*: Lee por teclado la opción seleccionada por el usuario, valida que la opción está entre 1 y 4, y una vez que ha leído una opción válida llamará a la función correspondiente en función de la opción seleccionada.
- `opcion1`
  - *Descripción*: Lee por teclado el valor del radio del círculo, valida que el radio siempre sea mayor que 0 y una vez que ha validado el radio llamará a la función `calcular_area_circulo`.
- `opcion2`
  - *Descripción*: Lee por teclado el valor de la base y la altura del triángulo, valida que los dos datos son mayores que 0 y una vez que los ha validado llamará a la función `calcular_area_triangulo`.
- `opcion3`
  - *Descripción*: Lee por teclado el valor de la base y la altura del rectángulo, valida que los dos datos son mayores que 0 y una vez que los ha validado llamará a la función `calcular_area_rectangulo`.

Solución

```
import areas
import os

# Definición de funciones
def mostrar_menu():
    print("1. Calcular el área de un círculo")
    print("2. Calcular el área de un triángulo")
    print("3. Calcular el área de un rectángulo")
    print("4. Salir")

def principal():
    opcion = 1
```

```
    while opcion != 4:
        os.system("clear")
        mostrar_menu()

        opcion = int(input("Introduce una opción (1-4): "))
        while opcion < 1 or opcion > 4:
            print("Error. La opción no es válida")
            opcion = int(input("Introduce una opción (1-4): "))

        if opcion == 1:
            opcion1()
        elif opcion == 2:
            opcion2()
        elif opcion == 3:
            opcion3()

        if opcion != 4:
            input("\nPulse una tecla para continuar...")

def opcion1():
    print("\nOpción 1")
    radio = float(input("Introduce el radio del círculo: "))
    area = areas.calcular_area_circulo(radio)
    print(f"El área del círculo es {area}")

def opcion2():
    print("\nOpción 2")
    base = float(input("Introduce el valor de la base: "))
    altura = float(input("Introduce el valor de la altura: "))
    area = areas.calcular_area_triangulo(base, altura)
    print(f"El área del triángulo es {area}")

def opcion3():
    print("\nOpción 3")
    base = float(input("Introduce el valor de la base: "))
    altura = float(input("Introduce el valor de la altura: "))
    area = areas.calcular_area_rectangulo(base, altura)
    print(f"El área del rectángulo es {area}")

#-----------------------------------------
# Programa Principal (Main)

principal()
```

Ver solución en GitHub

7. Entornos virtuales

7.1. Crear un entorno virtual

```
python3 -m venv mi-entorno
```

7.2. Activar un entorno virtual

Activar un entorno virtual en **Linux** o **macOS**.

```
source mi-entorno/bin/activate
```

Activar un entorno virtual en **Windows** desde PowerShell.

```
.\mi-entorno\Scripts\Activate.ps1
```

Puede ser que en **Windows** necesite habilitar la ejecución de scripts en PowerShell.

```
Set-ExecutionPolicy Unrestricted
```

Referencia:

- Configuración de PowerShell para crear el entorno virtual en Windows

7.3. Desactivar un entorno virtual

```
deactivate
```

8. Web scraping con Request y BeautifulSoup

8.1. Instalación de la librería `requests`

```
python3 -m pip install requests
```

Referencia:

- https://requests.readthedocs.io/en/latest/

8.2. Ejemplo

```
# Importamos los módulos necesarios
import requests

# Hacemos una petición HTTP a una URL
url = "https://www.marca.com"
page = requests.get(url)

# Mostramos el contenido de la respuesta
print(page.text)
```

8.3. Instalación de la librería `beautifulsoup`

```
python3 -m pip install beautifulsoup4
```

Referencia:

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/

8.4. Ejemplo 1

En este ejemplo vamos a obtener el texto que hay encerrado en todas las etiquetas `<h2>` de un sitio web.

```
# Importamos los módulos necesarios
import requests
from bs4 import BeautifulSoup

# Hacemos una petición HTTP a una URL
url = "https://www.marca.com"
page = requests.get(url)

# Convertimos la respuesta en un objeto BeautifulSoup
soup = BeautifulSoup(page.content, "html.parser")

# Obtenemos todas los elementos de tipo h2
titulares = soup.find_all("h2")

# Mostramos el texto de los titulares
for titular in titulares:
    print(titular.text.strip())
```

8.5. Ejemplo 2

En este ejemplo vamos a obtener el valor de los atributos `src` y `alt` de todas las etiquetas `<img>` de un sitio web.

```
# Importamos los módulos necesarios
import requests
from bs4 import BeautifulSoup

# Hacemos una petición HTTP a una URL
url = "https://www.marca.com"
page = requests.get(url)

# Convertimos la respuesta en un objeto BeautifulSoup
soup = BeautifulSoup(page.content, "html.parser")

# Obtenemos todos las etiquetas img del documento
imagenes = soup.find_all("img")

# Obtenmos los atributos 'src' y 'alt' de cada imagen
for imagen in imagenes:
    try:
        print(f"{imagen['src']} {imagen['alt']}")
        #print(f"{imagen.get('src')} {imagen.get('alt')}")
    except KeyError:
        pass
```

9. Descargar archivos de una URL con `wget`

9.1. Instalación de la librería `wget`

```
python3 -m pip install wget
```

Referencia:

- https://pypi.org/project/wget/

9.2. Ejemplo

En este ejemplo vamos a descargar todas las imágenes de un sitio web que tengan la extensión `.jpg` o `.png`.

```
# Importamos los módulos necesarios
import requests
from bs4 import BeautifulSoup
import wget

# Hacemos una petición HTTP a una URL
url = "https://www.marca.com"
page = requests.get(url)

# Convertimos la respuesta en un objeto BeautifulSoup
soup = BeautifulSoup(page.content, "html.parser")

# Obtenemos todos las etiquetas img del documento
imagenes = soup.find_all("img")

for imagen in imagenes:
    try:
        print(imagen['src'])
        url_imagen = imagen['src']

        if url_imagen.endswith('.jpg') or url_imagen.endswith('.png'):
            wget.download(url_imagen)
    except KeyError:
        pass
```

10. Descargar vídeos de YouTube

10.1. Instalación de la librería `pytube`

```
python3 -m pip install pytube
```

Referencia:

- https://pytube.io/en/latest/index.html

10.2. Ejemplo

En este ejemplo vamos a descargar un vídeo de YouTube a partir de su URL.

```
from pytube import YouTube

url = "https://www.youtube.com/watch?v=fzI9FNjXQ0o"
video = YouTube(url)
video.streams.filter(progressive=True, file_extension='mp4').order_by('resolution').desc().first().download()
```

11. Repositorio en GitHub con ejemplos de web scraping

En GitHub puede encontrar un repositorio con ejemplos de cómo realizar web scraping en Python.

- https://github.com/josejuansanchez/web-scraping-python

12. Ejemplo de un bot de Telegram en Python

Bot de Telegram para recibir las últimas noticias publicadas en la página web de la Consejería de Educación de la Junta de Andalucía.

- https://github.com/josejuansanchez/novedades-educacion-andalucia

13. Processing.py (Processing + Python)

Processing.py es un proyecto open source desarrollado por Jonathan Feinberg que permite escribir sketches para Processing utilizando el lenguaje de programación Python.

- https://josejuansanchez.org/processing-python/

14. Jupyter Notebooks

```
docker run -it --rm -p 8888:8888 -v "$PWD/work":/home/jovyan jupyter/scipy-notebook
```

15. Autor

Este material ha sido desarrollado por José Juan Sánchez Hernández.

16. Licencia

17. Referencias

- Python 3 Documentation. Web oficial de Python.
- The Python Tutorial. Web oficial de Python.
- Introducción a la programación con Python. Bartolomé Sintes Marco.
- Curso de Introducción a Python. Javier Cózar del Olmo. Rafael Cabañas de Paz. Andrés R. Masegosa Arredondo.
- Introducción a python3. PLEDIN. José Domingo Muñoz.
- Repositorio en GitHub con ejercicios resueltos.
- Introducción a la programación con Python 3. Universitat Jaume I. Andrés Marzal, Isabel García y Pedro García.
- Google's Python Class. Google.
- El libro de Python.
- Python intermedio.
- Introduction and Intermediate Python. Nina Zakharenko.