

# U3.5

# Funciones



# Indice

# Funciones en Python

- Las funciones son estructuras esenciales de código.
- Unidades lógicas del programa.
- Dividen y organizan el código para mayor legibilidad y reutilización.
- Agrupan instrucciones para resolver un problema concreto.
- Objetivos:
  - Dividir y organizar el código.
  - Encapsular código repetitivo.

# Cómo definir una función en Python

- Utiliza la palabra reservada `def`.
- Nombre de la función seguido de paréntesis y lista de parámetros (opcional).
- Cabecera termina con dos puntos.
- Cuerpo de la función con sangrado mayor.
- Opcional: instrucción `return` para devolver un resultado.
- docstring se utiliza para documentar la función.

# Cómo usar una función en Python

- Para invocarla, escribe su nombre como una instrucción.
- Pasa los argumentos según los parámetros definidos.

python [Copy code](#)

```
def multiplica_por_5(numero):  
    print(f'{numero} * 5 = {numero * 5}')    # 7 * 5 = 35  
  
multiplica_por_5(7)
```

# Parámetros y argumentos

# Diferencia entre parámetro y argumento

- Parámetro: definido en la función.
- Argumento: valor pasado a la función al ser invocada.

python [Copy code](#)

```
def multiplica_por_5(numero): # 'numero' es un parámetro  
    print(f'{numero} * 5 = {numero * 5}')
```

```
multiplica_por_5(7) # '7' es un argumento
```



# Parámetros

- Paso por valor: copia el valor de las variables.
- Paso por referencia: copia la dirección de memoria.

# Retorno de valores

# Sentencia return

- El retorno de valores es opcional, puede devolver o no un valor.
- Termina la ejecución de la función y continúa el programa.

python [Copy code](#)

```
def cuadrado_de_par(numero):  
    if not numero % 2 == 0:  
        return  
    else:  
        print(numero ** 2)    # 64
```

```
cuadrado_de_par(8)
```

# Devolver más de un valor

- En Python, se puede devolver más de un valor con `return`.
- En tal caso, por defecto, se devuelve una tupla de valores.

python [Copy code](#)

```
def cuadrado_y_cubo(numero):  
    return numero ** 2, numero ** 3
```

```
cuad, cubo = cuadrado_y_cubo(4)  
cuad      # 16  
cubo      # 64
```

# Devolver resultados en una lista

- Se pueden devolver diferentes resultados en una lista.


python [Copy code](#)

```
def tabla_del(numero):  
    resultados = []  
    for i in range(11):  
        resultados.append(numero * i)  
    return resultados  
  
res = tabla_del(3)  
res  # [0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
```

# Python siempre devuelve un valor

- A diferencia de otros lenguajes, Python no tiene procedimientos.
- Si no hay `return`, se devuelve automáticamente `None`.

python

 Copy code

```
def saludo(nombre):  
    print(f'Hola {nombre}')
```

```
print(saludo('j2logo')) # Hola j2logo \n None
```

# Variables y ámbito

# Ámbito y ciclo de vida de las variables

- Local: dentro de una función, no accesible fuera.
- Global: definidas fuera de funciones, visibles dentro.

python [Copy code](#)

```
def muestra_x():  
    x = 10  
    print(f'x vale {x}') # x vale 10  
  
x = 20  
muestra_x()  
print(x) # 20
```



# Ámbito y ciclo de vida de las variables

- Ámbito local: dentro de una función.
- Variables desaparecen al finalizar la función.

# Variables globales

- Ámbito global: fuera de funciones.
- Pueden ser consultadas dentro de funciones.

python [Copy code](#)

```
y = 20
def muestra_x():
    global x
    x += 1
    print(f'x vale {x}')
    print(f'y vale {y}')
```

# Funciones Lambda en Python

- Funciones anónimas y pequeñas.
- No requieren definición con `def`.
- **Sintaxis:** `lambda argumentos: expresion.`

python [Copy code](#)

```
doble = lambda x: x * 2
```

# Resumen

- Funciones en Python son unidades lógicas de código.
- Dividen y organizan el código, facilitando la reutilización.
- Se definen con `def`, seguido del nombre y parámetros (opcional).
- Se invocan escribiendo el nombre y pasando argumentos.
- `return` opcional para devolver resultados.
- Ámbito de las variables: local y global.
- Las funciones lamda son utiles para casos simples y breves.

¡Gracias por su atención!