



MÓDULO 5: FUNCIONES

- Función: mecanismo para agrupar funcionalidad y permitir la reutilización del código.
- Funciones vs Métodos.
- Opcionalmente pueden recibir parámetros.
- Opcionalmente pueden proporcionar un retorno.
- Deben declararse antes del primer uso.
- Manera posicional de paso de argumentos: el argumento está determinado por el orden.

MÓDULO 5: FUNCIONES

- **Sintaxis:**

- **Declaración:**

- *def nombre_función([argumentos]):*

- Bloque de código

- **Invocación:**

- *[variable =] nombre_función([argumentos])*

MÓDULO 5: FUNCIONES

- Ejemplos:

- Sin parámetros y sin retorno:

```
def saludar():
    print("Hola")
saludar()
```

- Sin parámetros y con retorno:

```
def dime_algo():
    return "algo"
respuesta = dime_algo()
print(respuesta)
```

MÓDULO 5: FUNCIONES

- **Ejemplos:**

- **Con parámetros y sin retorno:**

```
def saludar(nombre):  
    print("Hola", nombre)  
saludar("Marty McFly")
```

- **Con parámetros y con retorno:**

```
def calcular_doble(numero):  
    doble = numero * 2  
    return doble  
resultado = calcular_doble(3)  
print(resultado)
```

MÓDULO 5: FUNCIONES

■ Parámetros:

- Son opcionales y su número puede ser variable .
- Los valores se asignan por orden.
- Pueden asignarse valores por defecto.
 - Permiten invocaciones sin indicar el valor del parámetro.
 - Puede haber varios parámetros con valor por defecto.
 - Los parámetros con valor por defecto deben ir al final.
- Ejemplo:

```
def sumar(n1, n2=8, n3=2):  
    resultado = n1+n2+n3  
    return resultado  
  
print(sumar(1))  
print(sumar(1,2))  
print(sumar(3,2,10))
```

MÓDULO 5: FUNCIONES

- Parámetros:

- Asignación del valor None por defecto.

- Ejemplo:

```
def multiplicar(n1, n2=None):  
    if n2:  
        #if n2!=None:  
        #if n2 is not None:  
            print(n1 * n2)  
    else:  
        print("No ha indicado el segundo parámetro")  
multiplicar(3)
```

MÓDULO 5: FUNCIONES

- Ámbitos de las variables en las funciones.
 - Ámbito global.
 - Está definida fuera de las funciones.
 - Es visible desde cualquier ámbito.
 - Ámbito local.
 - Está definida dentro de una función.
 - Es visible sólo desde el ámbito de la función.
 - Mismo nombre → Se impone el ámbito más reducido.

MÓDULO 5: FUNCIONES

- Ámbitos de las variables en las funciones.
 - Se puede definir una variable dentro de una función con ámbito global, declarándola previamente con la palabra clave **global**.

```
b=5  
def multiplicar(a):  
    global rdo  
    rdo = b*a  
multiplicar(3)  
print(rdo)
```

MÓDULO 5: FUNCIONES

- Ámbitos de las variables en las funciones.

- La palabra clave **global** permite indicar que una variable de una función es global de forma explícita.

```
x = 10
def incrementar():
    global x
    x = x + 5
    print("En la función",x)
```

```
incrementar()
print("En el ámbito global",x)
```

```
>>En la función 15
>>En el ámbito global 15
```

MÓDULO 5: FUNCIONES

- **Paso de parámetros por referencia:**
 - **Las modificaciones en la función afectan al origen del parámetro.**

```
lista = [1, 2, 3]
def agregar(lista_modificada, item):
    lista_modificada.append(item)
agregar(lista, 4)
print(lista)
```

MÓDULO 5: FUNCIONES

■ Paso de parámetros por valor:

- Las modificaciones en la función **NO** afectan al origen del parámetro.

```
valor=5
```

```
def sumar(valor_modificado, incremento):  
    valor_modificado+=incremento  
sumar(valor, 4)  
print(valor)
```

MÓDULO 5: FUNCIONES

■ Lista de parámetros variables.

- Admite un número indeterminado de valores separados por comas.
- Se indica con **nombre_parámetro*
- Se recoge como una tupla.
- Sólo puede haber uno y debe ser el último.
- Atención: admite parámetros con valor por defecto.
- Ejemplo:

```
def sumar(*sumandos):  
    resultado = 0  
    for s in sumandos:  
        resultado+=s  
    return resultado  
rdo = sumar(5,4,3,10)  
print(rdo)
```

MÓDULO 5: FUNCIONES

■ Variables como funciones.

- Una variable puede contener la referencia a una función.

- Asignación:

- *nombreVariable = nombre_función*

- Uso:

- *nombreVariable()*

- Ejemplo:

```
def saludar(nombre):  
    print("Hola", nombre)  
miVariable = saludar  
miVariable("Doc")
```

MÓDULO 5: FUNCIONES

■ Variables como funciones.

- Asignación de función VS Asignación de retorno

```
def dame_numero():
    return 15
miVar1 = dame_numero() #Asignación del retorno de la llamada
miVar2 = dame_numero #Asignación de la función
```

MÓDULO 5: FUNCIONES

■ Variables como funciones.

■ Asignación de listas de funciones

```
def f1():
    print("F1")

def f2():
    print("F2")

def f3():
    print("F3")

funciones = [f1,f2,f3]
funciones[1]()
for f in funciones:
    f()
```

MÓDULO 5: FUNCIONES

■ Funciones como retorno

```
def f1():
    print("F1")

def f2():
    print("F2")

def dameFuncion(indice):
    if indice==0:
        return f1
    else:
        return f2

dameFuncion(1)()
```

MÓDULO 5: FUNCIONES

- Recursividad:
 - Funciones que se llaman a sí mismas.
 - Problema de desbordamiento: *RecursionError (stack overflow)*

```
def analizador(lista):  
    if len(lista)>0:  
        #if (lista):  
            elemento = lista.pop()  
            print(elemento)  
            analizador(lista)  
lista = ["Lunes", "Martes", "Miércoles"]  
analizador(lista)
```

MÓDULO 5: FUNCIONES

- Creación de módulos:
 - Un módulo es una función o una colección de funciones almacenadas en un fichero con la extensión .py
 - E:\Python\Python39\Lib
 - Definición: modulo.py

```
def funcion1():
    print("F1")
def funcion2():
    print("F2")
```

MÓDULO 5: FUNCIONES

- Creación de módulos:

- Uso:

```
import modulo  
modulo.funcion1()
```

```
from modulo import funcion1  
funcion1()
```

```
from modulo import funcion1 as f  
f()
```

MÓDULO 5: FUNCIONES

■ Expresión yield

- Similar a return, pero no provoca salida del método
- Permite definir un generador o un generador asíncrono.
- Ejemplo:

```
def dame_dias():
    lista = ["martes", "miercoles", "jueves", "viernes"]
    for dia in lista:
        yield dia
dias_oferta = dame_dias()
for d in dias_oferta:
    print(d)
print ("Fin del proceso sincrono")
```

■ Salida del ejemplo:

```
martes
miercoles
jueves
viernes
Fin del proceso sincrono
```