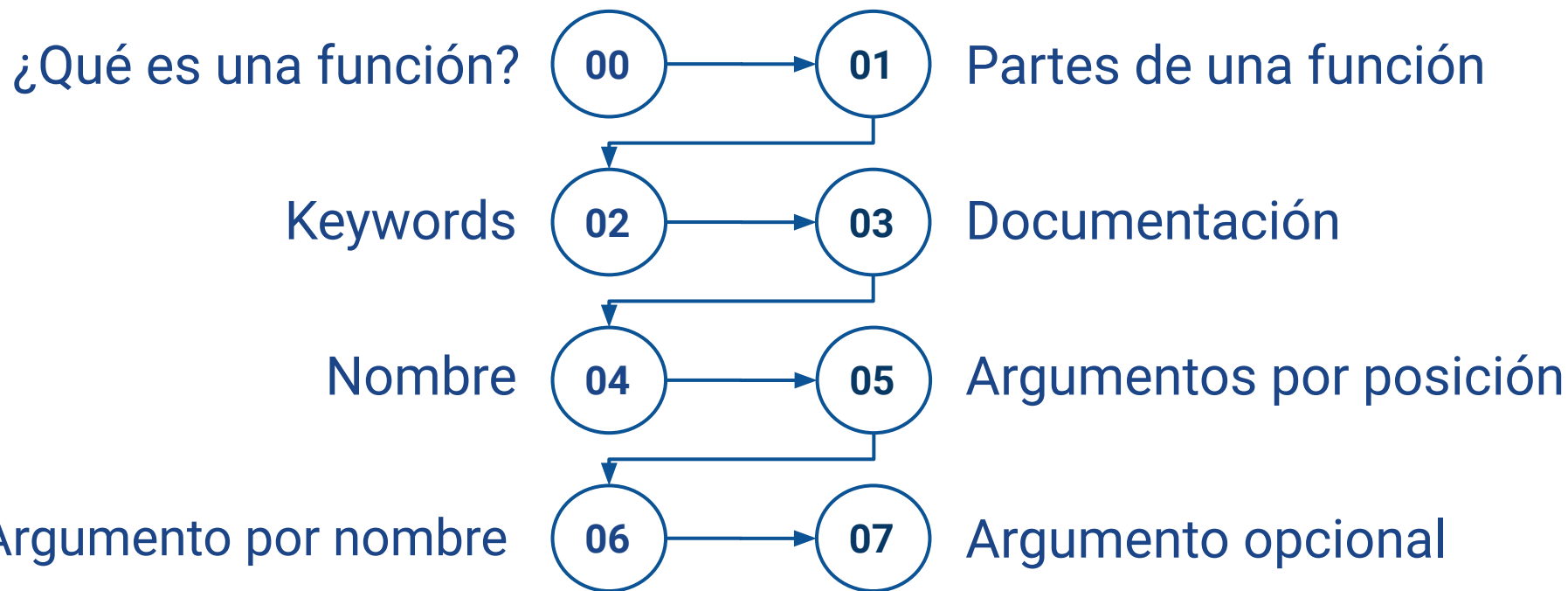


Funciones

Programación y Laboratorio I

Funciones



¿Qué es una función?

Una función es simplemente un "fragmento" de código que se puede usar una y otra vez, en lugar de escribirlo varias veces.

¿Qué es una función?

Las funciones permiten a los programadores dividir o descomponer un problema en partes más pequeñas, cada una de las cuales realiza una tarea particular.

Partes de una función

Keyword

Nombre

Argumento

Argumento opcional

```
def calcular_precio_con_iva(valor_sin_iva, iva=21):  
    '''Documentación'''  
    resultado = valor_sin_iva * (1+(iva/100))  
    return resultado
```

Documentación

Variable local

Valor de retorno

Partes de una función

```
def calcular_precio_con_iva(valor_sin_iva,iva=21):  
    '''Suma el IVA al precio, por defecto toma 21%'''  
    resultado = valor_sin_iva * (1+(iva/100))  
    return resultado  
  
print(calcular_precio_con_iva(100))           # 121.0  
print(calcular_precio_con_iva(100,10.5))      # 110.5
```

Keyword: **def** y **return**

El uso de **def** nos permite crear una función para que la función devuelva uno o varios valores, debemos usar **return**.

```
def funcion_suma(a, b):  
    return a + b
```

```
suma = funcion_suma(33, 1)  
print("La suma es", suma)
```

Es importante documentar las funciones.

```
def sumar(variable_a, variable_b):  
    '''  
    Indicar que hace  
    Que parámetros acepta  
    Que devuelve  
    '''  
    return a+b
```


Las funciones y las variables se deberán escribir en formato **snake_case**.

Las palabras separadas por barra baja (underscore) en vez de espacios y con la primera letra de cada palabra en minúscula.

`precio_con_iva`

Argumento por posición

Los argumentos posicionales son la forma más básica e intuitiva de pasar parámetros.

```
def resta(variable_a, variable_b):  
    return variable_a-variable_b
```

```
print(resta(15, 5)) # 10
```

Argumento por nombre

Otra forma de llamar a una función, es usando el nombre del argumento con = y su valor.

```
def resta(variable_a, variable_b):  
    return variable_a-variable_b  
  
print(resta(variable_b=5,variable_a=15)) # 10
```

Argumento opcional

Permite tener una función con algún parámetro opcional, que pueda ser usado o no dependiendo de diferentes circunstancias.

```
def resta(variable_a, variable_b, variable_c=1):  
    return variable_a-variable_b
```

```
aux = 12  
print(resta(variable_b = 12, variable_a=2)) # -10  
print(resta(12,2)) # 10
```

Parametros con tipo

Es posible documentar los tipos de los parámetros que espera recibir una función.

```
def resta(variable_a:int, variable_b:int=5):  
    return variable_a-variable_b
```

```
print(resta(variable_a=15)) # 10
```

Retorno con tipo

La -> (flecha) se usa para documentar el valor de retorno de una función.

```
def resta(variable_a:int)-> int:  
    return variable_a-1  
  
print(resta(15)) # 14
```