CONQUER BLOCKS

PYTHON

FUNCIONES: RETORNO, ARGUMENTOS ARBITRARIOS Y USO DE MODULOS

VALORES DE RETORNO

Valores simples:

```
def recibir_nombre_formateado(nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', 'hendrix')
print(musico)
```

✓ 0.0s

```
Jimi Hendrix
```

VALORES DE RETORNO

Valores simples:

```
def recibir_nombre_formateado(nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', 'hen
print(musico)
```

✓ 0.0s

```
Jimi Hendrix
```

Recordad que podemos añadir argumentos opcionales:

```
def recibir_nombre_formateado(primer_nombre, segundo_nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = primer_nombre + ' ' + segundo_nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi',  'hendrix')
print(musico)
```

⊗ 0.1s

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 6
      3     nombre_completo = primer_nombre + ' ' + segundo_nombre + ' ' + apellido
      4     return nombre_completo.title()
----> 6 musico = recibir_nombre_formateado('jimi',  'hendrix')
      7 print(musico)

TypeError: recibir_nombre_formateado() missing 1 required positional argument: 'apellido'
```

VALORES DE RETORNO

Valores simples:

```
def recibir_nombre_formateado(nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', 'her
print(musico)
```

✓ 0.0s

```
Jimi Hendrix
```

Recordad que podemos añadir argumentos opcionales:

```
def recibir_nombre_formateado(primer_nombre, segundo_nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = primer_nombre + ' ' + segundo_nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', "lee",  'hendrix')
print(musico)
```

✓ 0.0s

```
Jimi Lee Hendrix
```

VALORES DE RETORNO

Valores simples:

```
def recibir_nombre_formateado(nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', 'hen
print(musico)
```

✓ 0.0s

```
Jimi Hendrix
```

Recordad que podemos añadir argumentos opcionales:

```
def recibir_nombre_formateado(primer_nombre, apellido,  segundo_nombre = ""):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = primer_nombre + ' ' + segundo_nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', 'hendrix')
print(musico)
```

[7] ✓ 0.0s

```
Jimi  Hendrix
```

VALORES DE RETORNO

Valores simples:

```
def recibir_nombre_formateado(nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', 'he
print(musico)
```

✓ 0.0s

```
Jimi Hendrix
```

Recordad que podemos añadir argumentos opcionales:

```
def recibir_nombre_formateado(primer_nombre, apellido,  segundo_nombre = ""):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = primer_nombre + ' ' + segundo_nombre + ' ' + apellido
    return nombre_completo.title()

musico = recibir_nombre_formateado('jimi', 'hendrix', 'lee')
print(musico)
```

✓ 0.0s

```
Jimi Lee Hendrix
```

VALORES DE RETORNO

Diccionarios:

```python
def construir_persona(nombre, apellido):
    """Devuelve un diccionario con informacion de la persona."""
    persona = {'nombre': nombre, 'apellido': apellido}
    return persona

datos_persona = construir_persona("Juan", "Gomez")
print(datos_persona)
```

✓ 0.0s

```
{'nombre': 'Juan', 'apellido': 'Gomez'}
```

VALORES DE RETORNO

Diccionarios:

```python
def construir_persona(nombre, apellido, edad = ""):
    """Devuelve un diccionario con informacion de la persona."""
    persona = {'nombre': nombre, 'apellido': apellido}

    if edad:
        persona["edad"] = edad

    return persona

datos_persona = construir_persona("Juan", "Gomez", 25)

print(datos_persona)
```

✓ 0.0s

```
{'nombre': 'Juan', 'apellido': 'Gomez', 'edad': 25}
```

VALORES DE RETORNO

Diccionarios:

```python
def construir_persona(nombre, apellido, edad = ""):
    """Devuelve un diccionario con informacion de la persona."""
    persona = {'nombre': nombre, 'apellido': apellido}

    if edad:
        persona["edad"] = edad

    return persona

datos_persona = construir_persona("Juan", "Gomez")

print(datos_persona)
```

✓ 0.0s

```
{'nombre': 'Juan', 'apellido': 'Gomez'}
```

USO DE FUNCIONES EN BUCLES

```python
def recibir_nombre_formateado(nombre, apellido):
    """Devuelve el nombre completo bien formatead"""
    nombre_completo = nombre + ' ' + apellido
    return nombre_completo.title()

# Esto es un bucle infinito
while True:
    print("\nDime tu nombre completo:")
    nombre = input("Nombre: ")
    apellido = input("Apellido: ")

    nombre_formateado = recibir_nombre_formateado(nombre, apellido)

    print("\nHola, " + nombre_formateado + "!")
```

```
Dime tu nombre completo:

Hola, Juan Gomez!

Dime tu nombre completo:

Hola, Carlos Jimenez!
```

LISTAS EN FUNCIONES

Pasar una lista:

```
def saludar_usuarios(lista_nombres):
    """Imprime un saludo simple a todos los elementos de la lista."""
    for nombre in lista_nombres:
        mensaje = "Hola, " + nombre.title() + "."
        print(mensaje)

usuarios = ['juan', 'diego', 'lucas']
saludar_usuarios(usuarios)
```

✓ 0.0s

```
Hola, Juan.
Hola, Diego.
Hola, Lucas.
```

LISTAS EN FUNCIONES

Modificar una lista:

```
# Lista de diseños que deben ser impresos en 3D
encargos = ['funda iphone', 'robot', 'triangulo']
finalizados = []
# Simulamos que se imprimen los diseños hasta que no
# queda ninguno
# Movemos los diseños impresos a la lista de finalizados
while encargos:
    diseño_actual = encargos.pop()
    # Simulamos la creacion de un modelo 3D
    print("Imprimiendo modelo: " + diseño_actual)
    finalizados.append(diseño_actual)
# Mostrar diseños impresos
print("\nLos siguientrs modelos han sido impresos:")
for modelo_finalizado in finalizados:
    print(modelo_finalizado)
```

```
Imprimiendo modelo: triangulo
Imprimiendo modelo: robot
Imprimiendo modelo: funda iphone

Los siguientrs modelos han sido impresos:
triangulo
robot
funda iphone
```

CONQUER BLOCKS

LISTAS EN FUNCIONES

Modificar una lista:

```
# Lista de diseños que deben ser impresos en 3D
encargos = ['funda iphone', 'robot', 'triangulo']
finalizados = []
# Simulamos que se imprimen los diseños hasta que no
# queda ninguno
# Movemos los diseños impresos a la lista de finaliza
while encargos:
    diseño_actual = encargos.pop()
    # Simulamos la creacion de un modelo 3D
    print("Imprimiendo modelo: " + diseño_actual)
    finalizados.append(diseño_actual)
# Mostrar diseños impresos
print("\nLos siguientrs modelos han sido impresos:")
for modelo_finalizado in finalizados:
    print(modelo_finalizado)
```

```
def imprimir_modelos(encargos, finalizados):
    """
    Simula la impresion de ada diseño hasta que todos han sido completados.
    Mueve cada diseño a la lista de finalizados tras imprimirlo.
    """
    while encargos:
        diseño_actual = encargos.pop()

        # Simular la creacion de un modelo 3D
        print("Imprimiendo modelo: " + diseño_actual)
        finalizados.append(diseño_actual)

def muestra_modelos_completados(finalizados):
    """ Muestra los modelos impresos."""
    print("\nLos siguientes mdelos han sido impresos:")
    for modelo_finalizado in finalizados:
        print(modelo_finalizado)


modelos_encargados = ['iphone case', 'robot pendant', 'dodecahedron']
modelos_completados= []
imprimir_modelos(modelos_encargados, modelos_completados)
muestra_modelos_completados(modelos_completados)
```

✓ 0.0s

```
Imprimiendo modelo: dodecahedron
Imprimiendo modelo: robot pendant
Imprimiendo modelo: iphone case

Los siguientes mdelos han sido impresos:
dodecahedron
robot pendant
iphone case
```

LISTAS EN FUNCIONES

Prevenir la modificación de una lista

```
def imprimir_modelos(encargos, finalizados):
    """
       Simula la impresion de ada diseño hasta que todos han sido completados.
       Mueve cada diseño a la lista de finalizados tras imprimirlo.
    """
    while encargos:
        diseño_actual = encargos.pop()

        # Simular la creacion de un modelo 3D
        print("Imprimiendo modelo: " + diseño_actual)
        finalizados.append(diseño_actual)

def muestra_modelos_completados(finalizados):
    """ Muestra los modelos impresos."""
    print("\nLos siguientes mdelos han sido impresos:")
    for modelo_finalizado in finalizados:
        print(modelo_finalizado)


modelos_encargados = ['iphone case', 'robot pendant', 'dodecahedron']
modelos_completados= []
imprimir_modelos(modelos_encargados, modelos_completados)
muestra_modelos_completados(modelos_completados)

print("----")
print(modelos_encargados)
```

```
Imprimiendo modelo: dodecahedron
Imprimiendo modelo: robot pendant
Imprimiendo modelo: iphone case

Los siguientes mdelos han sido impresos:
dodecahedron
robot pendant
iphone case
----
[]
```

LISTAS EN FUNCIONES

Prevenir la modificación de una lista

```
def imprimir_modelos(encargos, finalizados):
    """
       Simula la impresion de ada diseño hasta que todos han sido completados.
       Mueve cada diseño a la lista de finalizados tras imprimirlo.
    """
    while encargos:
        diseño_actual = encargos.pop()

        # Simular la creacion de un modelo 3D
        print("Imprimiendo modelo: " + diseño_actual)
        finalizados.append(diseño_actual)

def muestra_modelos_completados(finalizados):
    """ Muestra los modelos impresos."""
    print("\nLos siguientes modelos han sido impresos:")
    for modelo_finalizado in finalizados:
        print(modelo_finalizado)


modelos_encargados = ['iphone case', 'robot pendant', 'dodecahedron']
modelos_completados= []
imprimir_modelos(modelos_encargados[:], modelos_completados)
muestra_modelos_completados(modelos_completados)

print("----")
print(modelos_encargados)
```

```
Imprimiendo modelo: dodecahedron
Imprimiendo modelo: robot pendant
Imprimiendo modelo: iphone case

Los siguientes modelos han sido impresos:
dodecahedron
robot pendant
iphone case
----
['iphone case', 'robot pendant', 'dodecahedron']
```

USAR UN NUMERO ARBITRARIO DE ARGUMENTOS

Ejemplo: Pedido restaurante

```
def hacer_pizza(*ingredientes):
    """Imprimir la lista de ingredientes del pedido"""
    print(ingredientes)

hacer_pizza("pepperoni")
hacer_pizza("champignons", "pimiento verde", "aceitunas")
```

✓ 0.0s

```
('pepperoni',)
('champignons', 'pimiento verde', 'aceitunas')
```

Ingredientes es una tupla

USAR UN NUMERO ARBITRARIO DE ARGUMENTOS

Ejemplo: Pedido restaurante

```python
def hacer_pizza(dimension, *ingredientes):
    """Resumen del pedido"""
    print("Has pedido una pizza de", dimension, "cm.")
    print("La pizza contiene los siguientes ingredientes:")
    for ingrediente in ingredientes:
        print("-",ingrediente)
    print("\n")

hacer_pizza(16, "pepperoni")
hacer_pizza(12, "champignons", "pimiento verde", "aceitunas")
```

Ingredientes es una tupla

```
Has pedido una pizza de 16 cm.
La pizza contiene los siguientes ingredientes:
- pepperoni


Has pedido una pizza de 12 cm.
La pizza contiene los siguientes ingredientes:
- champignons
- pimiento verde
- aceitunas
```

USAR UN NUMERO ARBITRARIO DE ARGUMENTOS DE PALABRA CLAVE

```python
def construir_perfil(nombre, apellido, **informacion_usuario):
    """Construir un diccionario conteniendo todo
    lo que sabemos del usuario"""
    perfil = {}
    perfil[nombre] = nombre
    perfil[apellido] = apellido

    print(informacion_usuario)
    print(type(informacion_usuario))

    for clave, valor in informacion_usuario.items():
        perfil[clave] = valor
    return perfil

perfil_usuario = construir_perfil("alberto", "lopez",
                                  ubicacion="Madrid",
                                  trabajo = "programador")
```

```
5]  ✓  0.0s
{'ubicacion': 'Madrid', 'trabajo': 'programador'}
<class 'dict'>
```

USAR UN NUMERO ARBITRARIO DE ARGUMENTOS DE PALABRA CLAVE

```python
def construir_perfil(nombre, apellido, **informacion_usuario):
    """Construir un diccionario conteniendo todo
    lo que sabemos del usuario"""
    perfil = {}
    perfil[nombre] = nombre
    perfil[apellido] = apellido

    print(informacion_usuario)
    print(type(informacion_usuario))

    for clave, valor in informacion_usuario.items():
        perfil[clave] = valor
    return perfil

perfil_usuario = construir_perfil("alberto", "lopez",
                                  ubicacion="Madrid",
                                  trabajo = "programador")
```

```
5]  ✓  0.0s
{'ubicacion': 'Madrid', 'trabajo': 'programador'}
<class 'dict'>
```

USAR UN NUMERO ARBITRARIO DE ARGUMENTOS DE PALABRA CLAVE

```python
def construir_perfil(nombre, apellido, **informacion_usuario):
    """Construir un diccionario conteniendo todo
    lo que sabemos del usuario"""
    perfil = {}
    perfil["nombre"] = nombre
    perfil["apellido"] = apellido

    for clave, valor in informacion_usuario.items():
        perfil[clave] = valor
    return perfil

perfil_usuario = construir_perfil("alberto", "lopez",
                                  ubicacion="Madrid",
                                  trabajo = "programador")
print(perfil_usuario)
```

✓ 0.0s

```
{'nombre': 'alberto', 'apellido': 'lopez', 'ubicacion': 'Madrid', 'trabajo': 'programador'}
```

GUARDAR FUNCIONES EN MODULOS

La ventaja de las funciones es la forma en que separan bloques de código de tu programa principal. Al usar nombres descriptivos para tus funciones, tu programa principal será mucho más fácil de seguir.

➡ Puedes ir un paso más allá almacenando tus funciones en un archivo separado llamado módulo y luego importando ese módulo en tu programa principal.

GUARDAR FUNCIONES EN MODULOS

pizza.py

```python
def hacer_pizza(dimension, *ingredientes):
    """Resumen del pedido"""
    print("Has pedido una pizza de", dimension, "cm.")
    print("La pizza contiene los siguientes ingredientes:")
    for ingrediente in ingredientes:
        print("-",ingrediente)
    print("\n")
```

hacer_pizza.py

nombre función

nombre modulo

```python
import pizza

pizza.hacer_pizza(16, 'pepperoni')
pizza.hacer_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

GUARDAR FUNCIONES EN MODULOS

pizza.py

```python
def hacer_pizza(dimension, *ingredientes):
    """Resumen del pedido"""
    print("Has pedido una pizza de", dimension, "cm.")
    print("La pizza contiene los siguientes ingredientes:")
    for ingrediente in ingredientes:
        print("-",ingrediente)
    print("\n")
```

hacer_pizza.py

nombre función

nombre modulo

```python
import pizza

pizza.hacer_pizza(16, 'pepperoni')
pizza.hacer_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

GUARDAR FUNCIONES EN MODULOS

pizza.py

```python
def hacer_pizza(dimension, *ingredientes):
    """Resumen del pedido"""
    print("Has pedido una pizza de", dimension, "cm.")
    print("La pizza contiene los siguientes ingredientes:")
    for ingrediente in ingredientes:
        print("-",ingrediente)
    print("\n")
```

nombre función

nombre modulo

nombre función

hacer_pizza.py

```python
from pizza import hacer_pizza

hacer_pizza(16, 'pepperoni')
hacer_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

GUARDAR FUNCIONES EN MODULOS

pizza.py

```python
def hacer_pizza(dimension, *ingredientes):
    """Resumen del pedido"""
    print("Has pedido una pizza de", dimension, "cm.")
    print("La pizza contiene los siguientes ingredientes:")
    for ingrediente in ingredientes:
        print("-",ingrediente)
    print("\n")
```

nombre modulo nombre función

```python
from pizza import hacer_pizza as hpi

hpi(16, 'pepperoni')
hpi(12, 'mushrooms', 'green peppers', 'extra cheese')
```

alias

hacer_pizza.py

GUARDAR FUNCIONES EN MODULOS

pizza.py

```python
def hacer_pizza(dimension, *ingredientes):
    """Resumen del pedido"""
    print("Has pedido una pizza de", dimension, "cm.")
    print("La pizza contiene los siguientes ingredientes:")
    for ingrediente in ingredientes:
        print("-",ingrediente)
    print("\n")
```

hacer_pizza.py

nombre modulo

```python
import pizza as pi
```

alias

```python
pi.hacer_pizza(16, 'pepperoni')
pi.hacer_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

GUARDAR FUNCIONES EN MODULOS

pizza.py

```python
def hacer_pizza(dimension, *ingredientes):
    """Resumen del pedido"""
    print("Has pedido una pizza de", dimension, "cm.")
    print("La pizza contiene los siguientes ingredientes:")
    for ingrediente in ingredientes:
        print("-",ingrediente)
    print("\n")
```

hacer_pizza.py

nombre modulo

todas las funciones

```python
from pizza import *

hacer_pizza(16, "pepperoni")
hacer_pizza(12, "champignons", "pimiento verde", "aceitunas")
```

REPASO

1. Valores de retorno

2. Funciones en bucles

3. Numero arbitrario de argumentos + argumentos de palabra clave

4. Uso de listas y diccionarios en funciones

5. Uso de modulos

CONQUER BLOCKS