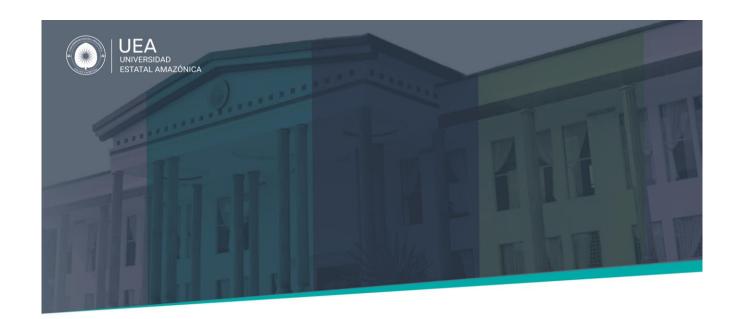




ASIGNATURA

FUNDAMENTOS DE PROGRAMACIÓN



Transformamos el mundo desde la Amazonía

Ing. Edwin Gustavo Fernández Sánchez, Mgs.

DOCENTE - PERSONAL ACADÉMICO NO TITULAR OCASIONAL

DIRECTOR DE GESTIÓN DE TECNOLOGÍAS INFORMACIÓN Y COMUNICACIÓN

UNIVERSIDAD ESTATAL AMAZÓNICA







DESARROLLO DE LA SEMANA 14: DEL LUN. 17 AL DOM. 23 DE MARZO/2025

Resultado de aprendizaje: Utilizar modularización para la simplificación de la estructura de un programa

CONTENIDOS

UNIDAD IV: Funciones, Colecciones y Archivos

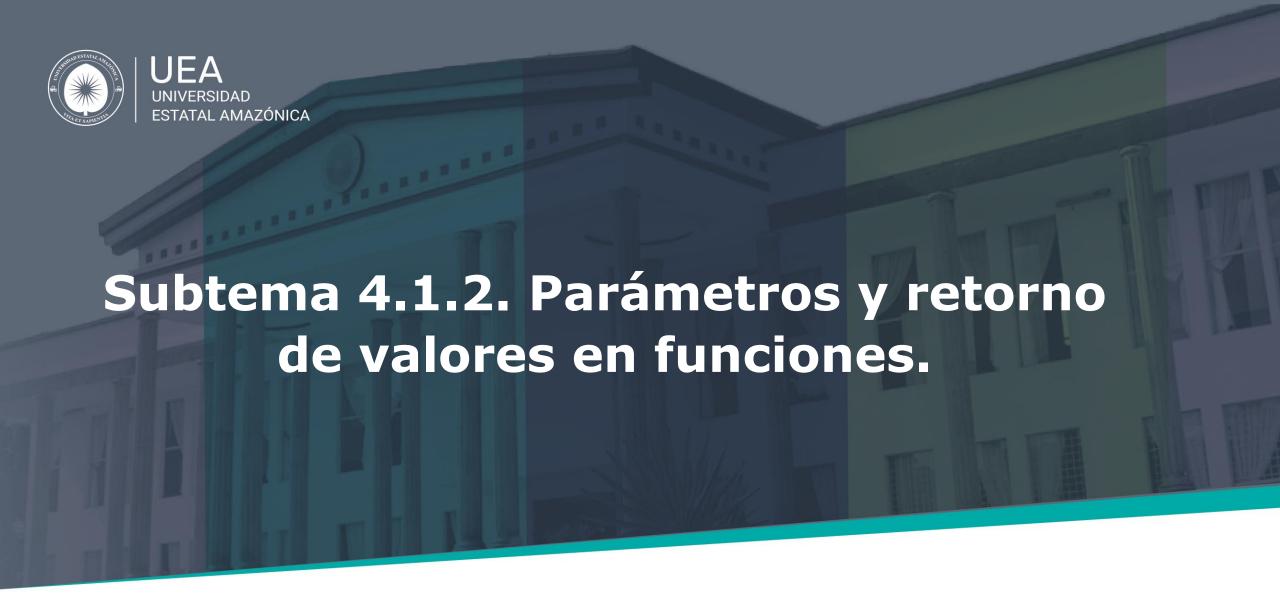
- Tema 4.1: Funciones
 - o Subtema 4.1.2: Parámetros y retorno de valores en funciones.



Unidad 4 Funciones, Colecciones y Archivos

Tema 4.1

Funciones



FUNDAMENTOS DE PROGRAMACIÓN (UEA-L-UFB-026)



Parámetros y Retorno de Valores en Funciones

- En programación, las funciones son bloques de código reutilizables que realizan tareas específicas. Hasta ahora, hemos visto cómo definir funciones y cómo utilizarlas en Python. Ahora, vamos a profundizar en cómo las funciones pueden comunicarse con el mundo exterior.
- Cuando creamos una función, a menudo queremos que haga algo con ciertos datos o información. Esto es donde entran en juego los "parámetros". Los parámetros son como los conductos de entrada de una función; permiten que la función reciba datos externos y los utilice en su interior. Piensa en ellos como <u>variables especiales que solo existen dentro</u> <u>de la función</u> y se llenan con valores cuando llamamos a la función.



Parámetros y Retorno de Valores en Funciones

- Pero no es suficiente que una función solo acepte datos; también queremos que la función pueda proporcionarnos resultados o información cuando termine su tarea.
- Aquí es donde entra en juego el <u>"retorno de valores"</u>. El retorno de valores es como el conducto de salida de una función; permite que <u>la función entregue un resultado específico cuando ha terminado su trabajo</u>.
- Esto hace que las funciones sean herramientas poderosas y flexibles en programación, ya que pueden adaptarse a diferentes situaciones y personalizarse según las necesidades del programador.



Explicación de la sintaxis

- <u>def</u>: Es la palabra clave utilizada para definir una función.
- <u>nombre de la funcion</u>: Es el nombre que elijamos para nuestra función. Debe seguir las mismas reglas de nombrado que las variables.
- <u>argumentos</u>: Son los valores que la función acepta como entrada. Pueden ser opcionales.
- # Código de la función: Aquí escribimos las instrucciones que la función ejecutará cuando sea llamada.
- <u>return</u> resultado: Opcionalmente, podemos usar la declaración return para devolver un valor como resultado de la función.

```
def nombre_de_la_funcion(argumentos):
    # Código de la función
    # Puede incluir múltiples instrucciones
    return resultado
```

```
def suma(a, b):
    resultado = a + b
    return resultado
```



Ejemplo

Este ejemplo ilustra cómo definir funciones con parámetros, cómo pasar argumentos a esas funciones y cómo usar parámetros predeterminados para hacer que las funciones sean más flexibles.

python

```
# Definición de una función con parámetros
def saludar(nombre, edad):
    print(f"Hola, {nombre} tienes {edad} años.")
# Llamada a la función y paso de argumentos
saludar("Juan", 25)
saludar("María", 30)
# Parámetros predeterminados
def saludar_con_saludo(nombre, edad, saludo="Hola"):
    print(f"{saludo}, {nombre} tienes {edad} años.")
# Llamada a la función con parámetro predeterminado
saludar_con_saludo("Pedro", 28)
saludar_con_saludo("Ana", 22, "iHola!")
```

