# Tabla de contenido

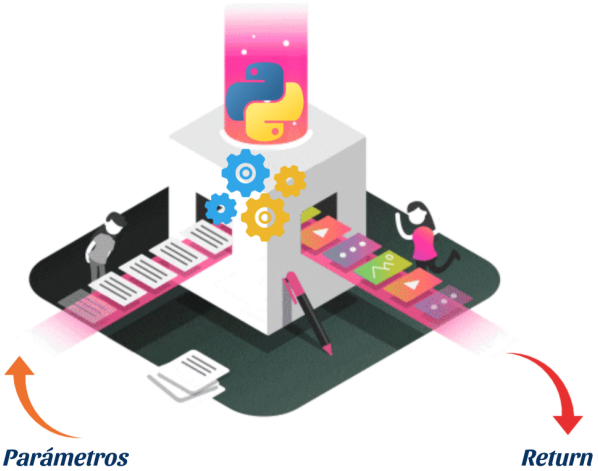
# 1. Objetivo de la unidad 2

Reconocer la sintaxis de Python, incluyendo la declaración de variables, tipos de datos, expresiones, estructuras de datos, control de flujo y bucles, para implementar programas simples y realizar tareas repetitivas. Asimismo, comprender la creación y el uso de funciones en Python, para desarrollar programas con código más modular, eficiente y reutilizable.

# 2. Tema 3: Funciones en Python

## 2.1 ¿Qué es una función en Python?

Una función en Python es un bloque de código que se ejecuta cuando es llamado. Se define utilizando la palabra clave **def**, seguida del nombre de la función y, opcionalmente, parámetros que pueden ser pasados a la función. Esta función realiza una tarea específica y puede devolver un resultado utilizando la declaración return.




**Características**

- **Reutilización de Código**: Las funciones permiten escribir una vez y reutilizar múltiples veces el mismo bloque de código, evitando la duplicación y promoviendo la modularidad.
- **Abstracción:** Las funciones ayudan a abstraer detalles complejos, permitiendo a los programadores centrarse en la lógica de alto nivel sin preocuparse por los detalles de implementación.
- **Modularidad:** Al dividir el código en funciones más pequeñas y manejables, se facilita la comprensión, depuración y mantenimiento del programa en su conjunto, lo cual promueve la limpieza y claridad del código.
- **Parámetros y Argumentos:** Las funciones pueden aceptar parámetros, que son valores que se pasan a la función cuando es llamada, y devolver resultados utilizando la declaración return.

### Ventajas

- Promueven la reutilización y la organización del código.
- Facilitan la abstracción y la comprensión del programa.
- Mejoran la modularidad y la mantenibilidad del código.

### Sintaxis de una función

A continuación, puedes observar los componentes en la sintaxis para definir una función en Python:

```
def nombre_de_la_funcion(parametro1, parametro2, ...):
    """
    Documentación opcional de la función
    """
    # Cuerpo de la función
    # Instrucciones que la función ejecuta
    return valor_de_retorno
```

- **def:** Palabra clave que indica el inicio de la definición de la función.
- **nombre_de_la_funcion:** Nombre que identifica a la función.
- parámetros: Variables que la función espera recibir como entrada. Pueden ser opcionales.
- **Cuerpo de la función:** Bloque de código que ejecuta una tarea específica.

- **return:** Declaración opcional que devuelve un valor calculado por la función como resultado.

### Identación en una función

En la definición de una función, la identación o sangría en las líneas posteriores a los dos puntos (:) delimita el bloque que se ejecuta adentro de la función.

### Ejemplo:

La siguiente función define la suma de dos números. Esta celda de código debe ser ejecutada para poder usar posteriormente la función.

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

Para utilizar la función, sólo debemos pasar como parámetros los números que deseamos sumar. Así obtenemos el resultado en el output:

```
suma(3,5)
Output: 8
```

Otra opción es guardar el resultado en una variable:

```
x=suma(3,5)
x
Output: 8
```

### Parámetros y argumentos

En Python, los términos "**parámetros**" y "**argumentos**" se utilizan para referirse a los valores que se pasan a una función, pero tienen significados ligeramente diferentes.

Los parámetros son nombres de variables que se utilizan en la definición de una función. Representan los valores que la función espera recibir cuando es llamada.

Los argumentos son los valores reales que se pasan a una función cuando es llamada. Son los datos específicos que se utilizan para completar los parámetros definidos en la función.

**Ejemplo:**

En este ejemplo, **nombre** es el parámetro de la función **saludar**, mientras que "**Miguel**" es el argumento pasado a la función cuando es llamada.

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

saludar("Miguel")
```

### Variables locales y globales

**Locales:** Son aquellas definidas dentro de una función. Sólo son accesibles desde dentro de esa función. No pueden ser accedidas desde fuera de la función en la que fueron definidas. Son creadas cuando la función es llamada y destruidas cuando la función termina su ejecución.

```
def mi_funcion():
    x = 10 # Esta es una variable local
    print(x)

mi_funcion() # Esto imprimirá 10
print(x) # Esto causará un error, ya que x no está definida fuera de la función
```

**Globales:** Son aquellas definidas fuera de todas las funciones. Son accesibles desde cualquier lugar del código, incluidas las funciones. Pueden ser accedidas y modificadas desde cualquier función.

### Retorno de valores

Permite que las funciones devuelvan un resultado o valor después de realizar algún tipo de procesamiento.

- **Declaración de retorno:** En Python, se utiliza la palabra clave **return** para devolver un valor desde una función. Puedes devolver cualquier tipo de dato, incluyendo números, cadenas, listas, tuplas, diccionarios, objetos personalizados, etc.
  **Ejemplo:**

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

resultado = suma(3, 5)
print(resultado) # Esto imprimirá 8
```

- **Valores de retorno múltiples:** Python permite que una función devuelva múltiples valores separados por comas. Estos valores se pueden almacenar en una sola variable como una tupla o se pueden desempaquetar en variables individuales.

  **Ejemplo:**

```
def operaciones(a, b):
    suma = a + b
    resta = a - b
    multiplicacion = a * b
    division = a / b
    return suma, resta, multiplicacion, division

resultados = operaciones(10, 5)
print(resultados) # Esto imprimirá (15, 5, 50, 2.0)
```

- **Valores de retorno en condicionales:** Puedes usar declaraciones **return** condicionales para devolver diferentes valores según una condición específica dentro de la función.

**Ejemplo:**

```
def es_positivo(numero):
    if numero > 0:
    return "Es positivo"
    elif numero < 0:
    return "Es negativo"
    else:
    return "Es cero"

print(es_positivo(10)) # imprimirá "Es positivo"
```

## 2.2 Ejemplos de funciones

**Ejemplo:**

Esta función toma un conjunto de datos almacenados en una lista y calcula el rango.

```
def calcular_rango(numeros):
    return max(numeros) - min(numeros)

# Ejemplo de uso:
lista_numeros = [10, 20, 30, 40, 50]
rango = calcular_rango(lista_numeros)
print("El rango de la lista es:", rango)
```

**Ejemplo:**

Esta función toma un conjunto de datos almacenados en una lista y calcula el promedio

```python
def calcular_promedio(numeros):
    return sum(numeros) / len(numeros)

# Ejemplo de uso:
lista_numeros = [10, 20, 30, 40, 50]
promedio = calcular_promedio(lista_numeros)
print("El promedio de la lista es:", promedio)
```

### 2.3 Autoevaluación

**Pregunta 1:**

¿Qué palabra clave se utiliza para definir una función en Python?

Opción 1: define

Opción 2: function

Opción 3: def

Opción 4: fun

**Pregunta 2:**

¿Cuál de las siguientes opciones describe mejor el propósito de los parámetros en una función?

Opción 1: Son valores devueltos por la función.

Opción 2: Son variables locales dentro de la función.

Opción 3: Son nombres de variables utilizados en la llamada a la función.

Opción 4: Son valores pasados a la función cuando es llamada.

## 3. Bibliografía

✓ Álvarez, C. A. (2020). Introducción al Jupyter Notebook y aplicaciones básicas. [Objeto_virtual_de_Informacion_OVI].

Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/35681

- ✓ Hinojosa Gutiérrez, Á. (2015). Python paso a paso: ( ed.). RA-MA Editorial, (pp. 21-24). https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/107213?page=1
- ✓ Hinojosa Gutiérrez, Á. (2015). Python paso a paso: ( ed.). RA-MA Editorial, (pp. 39). https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/107213?page=1
- ✓ Hinojosa Gutiérrez, Á. (2015). Python paso a paso: ( ed.). RA-MA Editorial, (pp. 49-88). https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/107213?page=1
- ✓ Hinojosa Gutiérrez, Á. (2015). Python paso a paso: ( ed.). RA-MA Editorial, (pp. 89-104). https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/107213?page=1
- ✓ Hinojosa Gutiérrez, Á. (2015). Python paso a paso: ( ed.). RA-MA Editorial, (pp. 105-122). https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/107213?page=1
- ✓ Cuevas Álvarez, A. (2016). Python 3: curso práctico. RA-MA Editorial, (pp. 145-182). https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/106404?page=1
- ✓ Código Marzal Varó, A. García Sevilla, P. & Gracia Luengo, I. (2016). Introducción a la programación con Python 3. D - Universitat Jaume I. Servei de Comunicació i Publicacions, (pp. 228-336). https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/51760?page=1
- ✓ Noguera, A. d. (2021). Lenguaje de programación Python. [Objeto_virtual_de_Informacion_OVI]. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/39346
- ✓ Aldana, J. M. (2023). *Estructuras de control cíclicas*. [Objeto_virtual_de_aprendizaje_OVA]. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/58883
- ✓ Amórtegui, M. P. (2021). Algoritmos y Programación. [Objeto_virtual_de_Informacion_OVI]. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/44535