

A graphic on the left side of the slide consisting of four stacked, 3D-style rectangular blocks in purple, orange, yellow, and blue. An orange arrow points to the right, passing through the middle of the blocks.

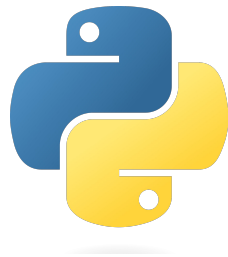
Agencia de
Aprendizaje
a lo largo
de la vida

CODO A CODO INICIAL

Clase 17

Python 5

Funciones I: Introducción



Les damos la bienvenida

Vamos a comenzar a grabar la clase



Clase 16

Python 4 - Condicionales II

- Estructuras condicionales.
- Uso de "if", "elif" y "else" para tomar decisiones en un programa.
- Uso de operadores lógicos.
- Uso de Match.
- Diferencias de sintaxis entre Python y PSeint

Clase 17

Python 5 - Funciones I: Introducción

- Definición.
- Propósito. Ventajas.
- Sintaxis para declarar funciones.
- Ejemplos de funciones sencillas.

Clase 18

Python 6 - Funciones II: Argumentos

- Parámetros
- Argumentos
- Ejemplos de funciones con parámetros y argumentos.
- Funciones integradas en el lenguaje (len(), print(), input(), etc.)

Funciones

En **Python**, una **función** es un bloque de código que constituyen una unidad lógica dentro del programa. Resuelve un problema específico, y permiten la modularidad del código.

Una función puede definir opcionalmente **parámetros** de entrada, que permiten pasar argumentos a la función en el momento de su llamada. Además, una función **también puede devolver un valor** como salida.

Las funciones nos permiten dividir el trabajo que hace un programa en tareas más pequeñas, separadas del código principal. Ese es el concepto de función en programación.

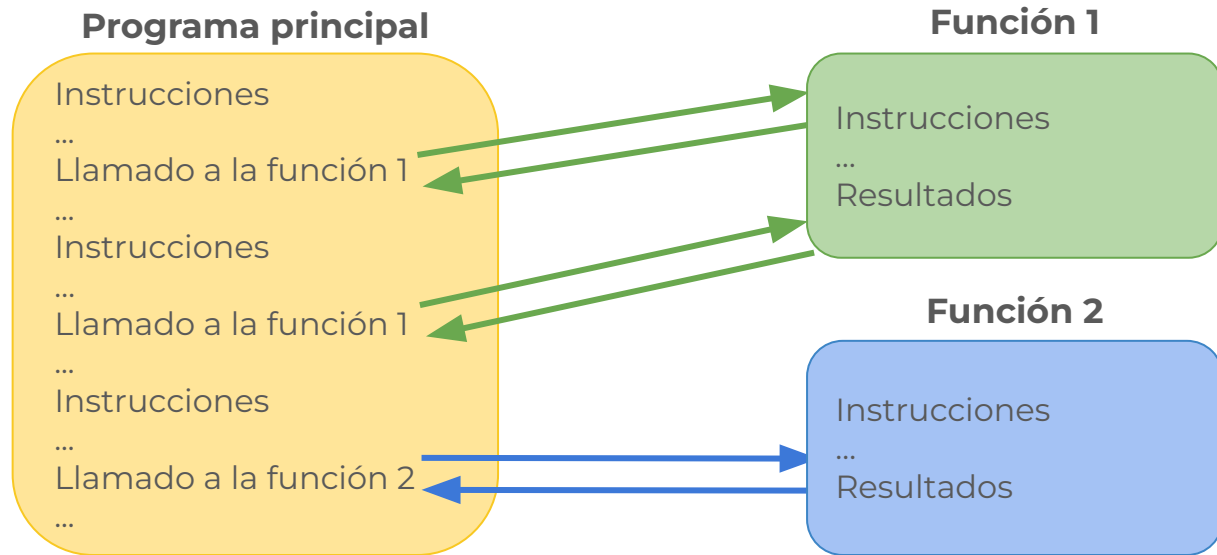
Funciones

Beneficios de la programación funcional:

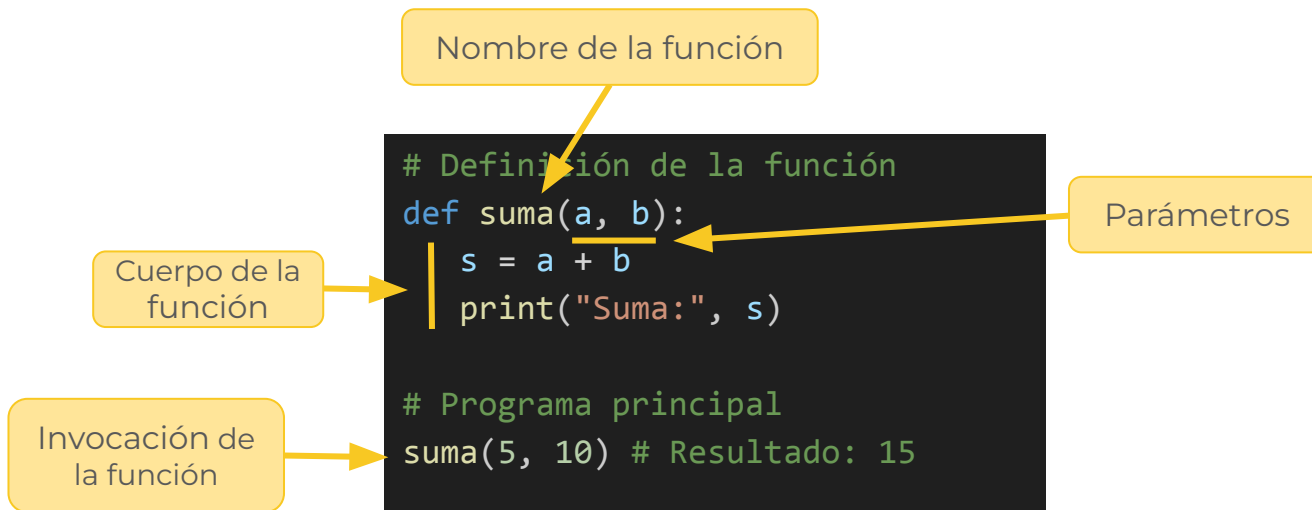
- **Facilita el trabajo en equipo.** Al modularizar el código permite trabajar en unidades lógicas separadas.
- **Encapsulamiento.** Se divide y organiza el código en partes más sencillas que se pueden encapsular en funciones y ser reutilizado a lo largo del proyecto.
- **Simplifica la lectura.** El código estructurado en funciones tiene un cuerpo principal reducido y funciones bien delimitadas.
- **Reutilización.** El código encapsulado en una función puede utilizarse en diferentes proyectos.
- **Mantenimiento:** El software que utiliza funciones es más fácil de mantener

Funciones

Las funciones permiten ejecutar código fuera del flujo normal del programa, recibiendo y devolviendo datos:



Funciones | Definición



Funciones | Definición

Elementos de una función:

- **Primera línea:** cabecera o definición de la función:
 - `def`: palabra reservada que define una función.
 - *nombre o identificador*: se utiliza para invocar la función.
 - *parámetros*: encerrados entre paréntesis, son opcionales (los veremos más adelante).
 - *dos puntos*: indican el cierre de la cabecera.
- **Cuerpo:** se delimita con la indentación, y constituye el código que se encapsula en la función.
- **Return:** Es opcional, permite a la función devolver valores al cuerpo principal del programa (lo veremos más adelante).

Funciones | Definición

Los nombres de las funciones siguen las mismas pautas vistas para nombrar variables, aunque utilizando verbos en infinitivo.

```
# Definición de la función
def saludar():
    print("Bienvenido al sistema!")
    print("Gracias por elegirnos!")

# Invocación o llamada de la función
saludar()
```

```
Bienvenido al sistema!
Gracias por elegirnos!
```

La función se debe definir antes de ser invocada por primera vez. Desde el programa principal se invoca a la función escribiendo su nombre.

Funciones | Reglas para el nombre

Las reglas para el **nombre de una función** son las mismas que para la de una variable. Las recordamos:

- El nombre debe comenzar con una letra
- Puede contener letras, números y guiones bajos
- No puede llamarse igual que una palabra reservada del lenguaje
- No debe contener caracteres especiales como letras acentuadas (á, é, í, ó, ú) ni caracteres especiales como la virgulilla (ñ).

Funciones | Ejemplo I

Cada vez que es invocada, la función **dibujar_rectángulo()** genera en la terminal un rectángulo de 10x3 asteriscos:

```
# Definición de la función
def dibujar_rectangulo():
    print(f"*****")
    print(f"*****")
    print(f"*****")

# Invocación o llamada de la función
dibujar_rectangulo()
print()
dibujar_rectangulo()
```

```
*****
*****
*****

*****
*****
*****
```

La línea en blanco que separa un rectángulo del otro proviene de la orden "print()" que está entre las dos llamadas a la función.


Funciones | Ejemplo II

La función **mostrar_informacion()** define tres variables (*nombre*, *edad* y *ciudad*) y luego imprime un mensaje mostrando su contenido:

```
# Definición de la función
def mostrarInformacion():
    nombre = "Juan"
    edad = 25
    ciudad = "Buenos Aires"

    print(f"Nombre: {nombre}")
    print(f"Edad: {edad} años")
    print(f"Ciudad: {ciudad}")

# Invocación o llamada de la función
mostrarInformacion()
```



```
Nombre: Juan
Edad: 25 años
Ciudad: Buenos Aires
```

La llamada a la función imprime la información almacenada en esas variables.

Funciones | Ejemplo III

```
# Definición de las funciones
def mostrarHorarioMatutino():
    print("El horario del turno de la mañana es de 08 a 12 horas")

def mostrarHorarioVespertino():
    print("El horario del turno de la tarde es de 14 a 18 horas")

def mostrarHorarioNocturno():
    print("El horario del turno de la noche es de 19 a 23 horas")

def mostrarMenu():
    print("Bienvenidos a la Universidad! ¿Qué horarios desea consultar?: ")
    print("1. Mañana")
    print("2. Tarde")
    print("3. Noche")
```

Funciones | Ejemplo III

```
# Programa principal
mostrarMenu()
opcion_elegida = int(input("Elige una opción (1, 2 o 3): "))

if opcion_elegida == 1:
    mostrarHorarioMatutino()
elif opcion_elegida == 2:
    mostrarHorarioVespertino()
elif opcion_elegida == 3:
    mostrarHorarioNocturno()
else:
    print("Opción no válida. Por favor, elige 1, 2 o 3.")
```

Este programa llamará a una función para mostrar el menú, luego pedirá un valor al usuario y llamará a una función distinta de acuerdo a la elección del usuario.

Desafíos

Desafío 1: Saludos Personalizados

Crear una función que imprima un saludo personalizado. La función no debe recibir ningún valor y siempre imprimirá el mismo mensaje, por ejemplo, *"¡Hola, bienvenidos al curso de Python!"*.

!No olvides escribir en el programa una llamada a la función que creaste!

```
¡Hola, bienvenidos al curso de Python!
```

Desafío 2: Generador de Patrones

Crear una función que imprima un patrón simple en la consola (como una pirámide). La función simplemente imprimirá el patrón cada vez que se llame.

La salida puede ser algo así:

```
  *
 ***
*****
*****
*****
*****
*****
```

Desafío 3: Impresión de Tabla de Multiplicar

Desarrollar una función que imprima la tabla de multiplicar de un número fijo (por ejemplo, la tabla del 3). Cada vez que se llame a la función, imprimirá la tabla completa de ese número ($3 \times 1 = 3$, $3 \times 2 = 6$, ..., $3 \times 10 = 30$).

La salida podría ser algo así:

```
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
...
3 x 10 = 30
```

Desafío 4: Funciones para una empresa

Una empresa requiere que desarrollemos estas funciones:

- Generar un encabezado de página con los datos de la empresa.
- Generar un pie de página que imprima una línea de asteriscos.
- Imprimir una fecha.
- Imprimir un saludo de bienvenida formal.
- Imprimir un saludo de despedida formal.

Luego invocarlas en el programa principal en un orden coherente.

Material extra

Artículos de interés

Material extra:

- [Funciones en Python](#) | Escuela Superior Politécnica del Litoral
- [Guía básica de funciones en Python](#) | Pablo Londoño

Videos:

- [Funciones en Python](#) | Tutoriales sobre Ciencia y Tecnología
- [Curso de Python](#), en Píldoras informáticas

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizar el Ejercicio de Repaso.

Todo en el Aula Virtual.

Muchas gracias por tu atención.

Nos vemos pronto