Funciones

Python

Repaso
Sentencias de control

Funciones

Introducción

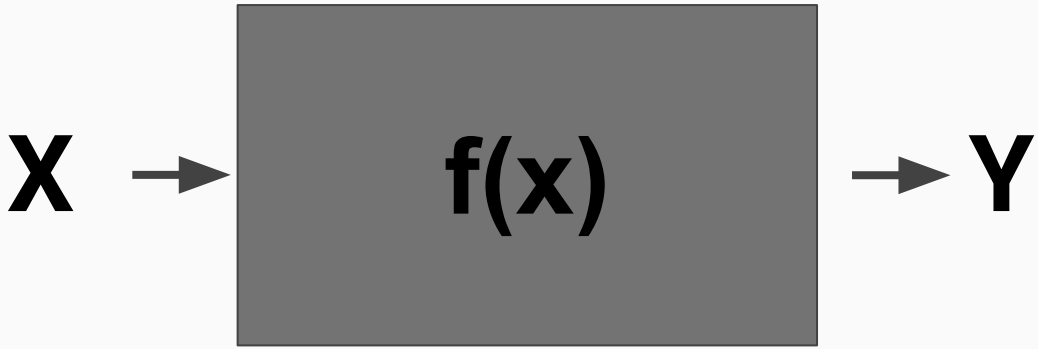
- ¿Qué pasa si quiero **reutilizar** un pedazo de **código** en diferentes partes de mi **programa**?

```
a = input("Ingresa un numero: ")
if a>0:
    print("El numero ingresado es positivo")
else:
```

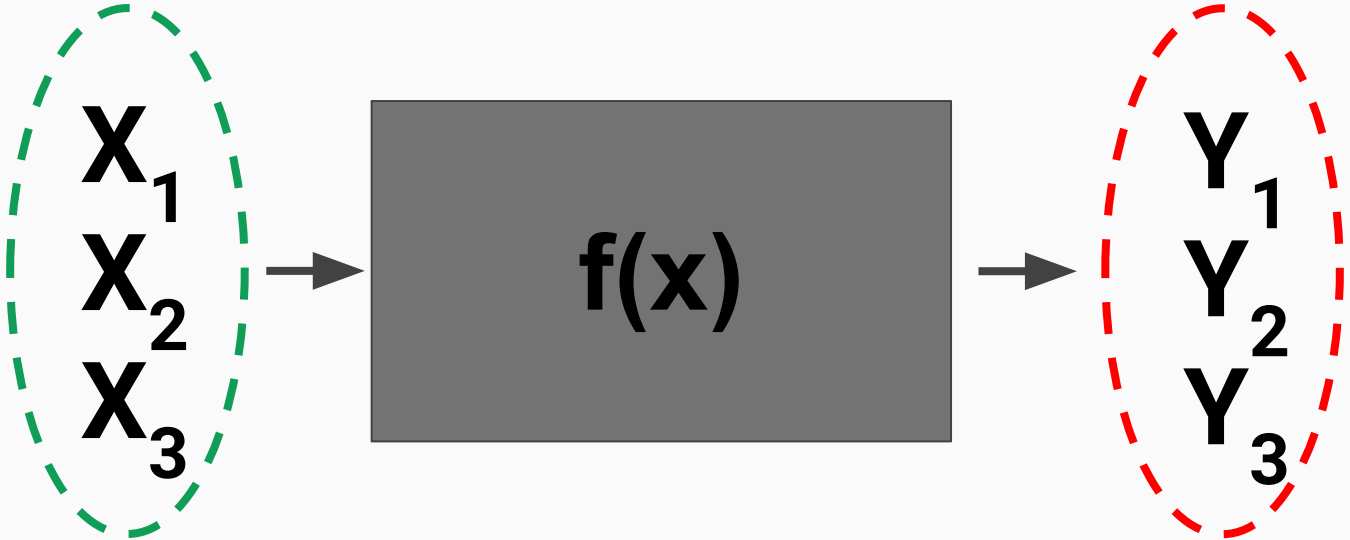
```
def positivo(numero):
    if numero>0:
        return "El numero ingreasdo es positivo"
    else:
        return "El numero ingreasdo es negativo"
a = input("Ingresa un numero: ")
print(positivo(a))
b = input("Ingresa un numero: ")
print(positivo(b))
```

```
    print("El numero ingresado es positivo")
else:
    print("El numero ingresado es negativo")
```

Introducción

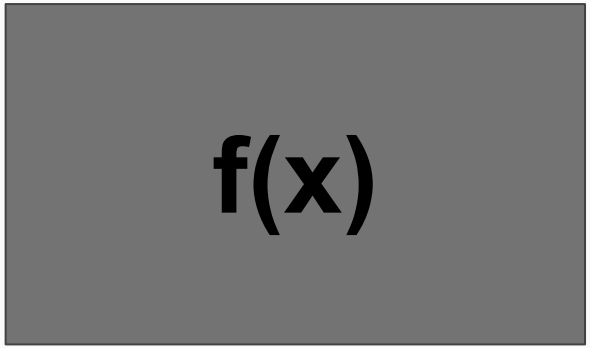


Introducción



Definición

- Una **función** es un **bloque** de **código organizado** y **reutilizable** que ejecuta cierta acción.
- Las **funciones** proveen mayor **modularidad** para nuestros **programas** y un mayor grado de **reutilización** de código.



Definición

- Utilizar **funciones** ofrece una serie de **ventajas:**
  - *Organización*
  - *Reusabilidad*
  - *Testeo*
  - *Abstracción*
  - *Extensibilidad*

Funciones en Python

Keyword Nombre Parámetro

```
def positivo (numero):
    if numero>0:
        return "El numero ingreasdo es positivo"
    else:
        return "El numero ingreasdo es negativo"
```

Return

Return

Funciones en Python: Observaciones

- Pueden **no** tener **return**

```
def print_nombre (nombre):
    print("Mi nombre es {}".format(nombre))
```

- Pueden **no** tener **parámetros**

```
def saludar ():
    print("Hola usuario")
```

- Los parámetros pueden tener valores por defecto de

```
def print_nombre (nombre="Juan"):
    print("Mi nombre es {}".format(nombre))
```

Ejemplos iniciales

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

```
def resta(a,b):
    return a-b
```

Ejemplos iniciales

```
def multiplicacion(a,b):
    return
```

```
def division(a,b):
    return
```

```
def potencia(a,b):
    return
```

```
def raiz(a,b):
    return
```

Ejercicios iniciales

- **Función** para evaluar si una palabra existe en una lista.
- **Función** para comprobar la igualdad de dos palabras sin distinguir mayúsculas y minúsculas.
- **Función** para evaluar si un alumno obtuvo un nota desaprobatoria o aprobatoria en su examen.

Funciones con sentencias de selección

Bienvenida de usuario

Hacer dos lista de cinco o más nombres de usuario cada una. Una lista para los usuarios administradores y la otra, los usuarios operadores.
Imagine que está escribiendo un código que imprimirá un saludo a cada usuario después de que inicien sesión en un sitio web.

Si el nombre de usuario es un administrador, imprima un saludo especial, como: "Hola administrador, ¿le gustaría ver un informe de estado?"
Si el nombre de usuario es un operador, imprima el saludo personal: "Hola Jaden, gracias por iniciar sesión nuevamente."
En caso contrario, imprimir el mensaje: "Usuario no existe"

Funciones con sentencias de selección

Entradas al cine

Una sala de cine cobra diferentes precios de entradas según la edad de una persona. Si una persona es menor de 3 años, el boleto es gratis; si tienen entre 3 y 12 años, el boleto es de S/. 7; y si son mayores de 12 años, el boleto es de S/. 15. Escriba un programa que le pregunte a los usuarios su edad y luego muestre el costo del boleto de cine.

Funciones con sentencias de selección

Asientos de restaurantes

Escriba un programa que le pregunte al usuario cuántas personas hay en su grupo de cena. Si la respuesta es más de ocho, mostrar un mensaje diciendo que tendrán que esperar una mesa. De lo contrario, informar que su mesa está lista.

Funciones con sentencias de iteración

Listar alumnos aprobados

Hacer un programa que permita calificar a las notas de cada alumno que exista en un diccionario que tiene la siguiente estructura de ejemplo:

```
{"Mauro": {"final": 10, "partial": 8}, "Juan": {"final": 15, "partial": 20}, ...}
```

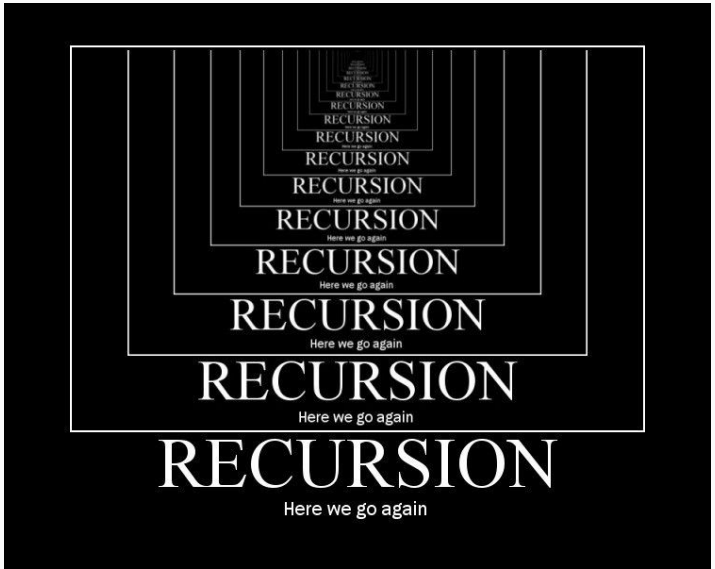
Si el promedio de las notas partial y final es mayor de 10.5, imprimir:
"Alumno Juan aprobado con promedio 17.5"
En caso contrario:
"Alumno Juan desaprobado con promedio 10.5"

Extra: Registrar las notas de los alumnos antes de evaluarlos.

Funciones: Recursividad

Concepto de recursividad

- Forma en la cual se **especifica** un **proceso** basada en su **propia definición.**
- Proceso por el cual una **función** se **llama a sí misma.**



Ejemplo: Factorial

$$1! = 1$$

$$2! = 2(1) = 2$$

$$3! = 3(2)(1) = 6$$

$$4! = 4(3)(2)(1) = 24$$

$$5! = 5(4)(3)(2)(1) = 120$$

Ejemplo: Fibonacci

The Fibonacci Sequence

1,1,2,3,5,8,13,21,34,55,89,144,233,377…

1+1=2
1+2=3
2+3=5
3+5=8
5+8=13
8+13=21
13+21=34
21+34=55
34+55=89
55+89=144
89+144=233
144+233=377

Ejercicios adicionales

- Escribe un programa en donde:
  - **Simules** una **máquina expendedora**. La máquina debe tener **5 productos** como mínimo con los siguientes **valores**:
    - **Nombre** del producto
    - **Precio**
    - **Cantidad** disponible
- Genera una **función** que reciba el **pedido** y el **dinero** de **cliente** y que **verifique si podrá obtener el producto** que quiere de la máquina expendedora.

Ejercicios adicionales

- Escribe una función que dé solución al problema de Hanoi para cualquiera cantidad de discos.
  - Listar todos los movimientos.

Resumen

- ¿Qué es una **función**?
- ¿Para qué **sirven**?
- ¿Qué **elementos** tiene en Python?
- ¿Qué es **recursividad**?