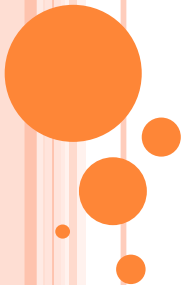


INTRODUCCIÓN A LA PROGRAMACIÓN

FUNCIONES



TEMAS

- Cómo usar funciones
- Cómo definir nuevas funciones
- Funciones con parámetros
- Funciones con valores de retorno



CÓMO USAR FUNCIONES

Python trae funciones preincorporadas para calcular casi todas las funciones matemáticas.

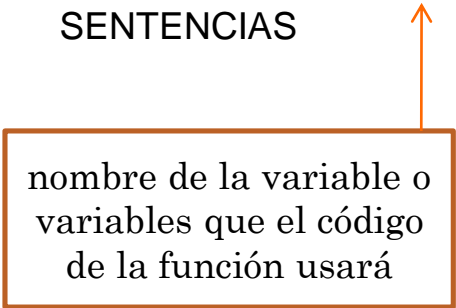
Por ejemplo:

```
import math  
raiz = math.sqrt(17)  
potencia = math.pow(2,2)
```



CÓMO DEFINIR NUEVAS FUNCIONES


- Cuando definimos funciones, estamos creando nuestras propias funciones.
- `def NOMBRE_FUNCION (parámetros):`
`SENTENCIAS`



nombre de la variable o variables que el código de la función usará

- El nombre de la función puede ser cualquiera que queramos, pero por convención, debe ser acorde a lo que resuelva.
- Una función PUEDE O NO recibir parámetros.
- Una función PUEDE O NO devolver valores.
- Ejemplos:

- ```
def SALUDAR ():
 print("Hola!!! Cómo estás?")
```



Esta  
función no  
recibe  
parámetro  
porque  
hace algo  
fijo



# EJEMPLO

- Puedo usar la función que ya creamos (saludar), en mi programa principal, de la siguiente manera:

```
n = int(input("Cuántas veces quiere saludar?"))
```

```
for i in range (1,n+1):
 SALUDAR()
```

- Si n=3, el resultado será:

Hola!!! Cómo estás?

Hola!!! Cómo estás?

Hola!!! Cómo estás?

Notar que se puede  
llamar a la función  
varias veces



## EN NUESTRAS FUNCIONES...

- Se puede incluir cualquier número de sentencias dentro de la función, pero todas deben escribirse con sangría a partir del margen izquierdo. Al igual que un for o un while.
- Dentro del código de la función, podemos usar todas las estructuras vistas.



# EJEMPLO

- `def SALUDAR_MUCHO():`  
    `for i in range(1,5):`  
        `print("Hola!! Cómo estás?")`
- El resultado será:

Hola!!! Cómo estás?  
Hola!!! Cómo estás?  
Hola!!! Cómo estás?  
Hola!!! Cómo estás?





## TAMBIÉN UNA FUNCIÓN PUEDE LLAMAR A OTRA FUNCIÓN YA DEFINIDA...

- def INVITACION():  
    SALUDAR()  
    print("Vamos a cenar hoy?")
- En el programa principal:  
    INVITACION()
- El resultado será:  
    Hola!!! Cómo estás?  
    Vamos a cenar hoy?



# PERO PARA QUÉ NOS SIRVEN LAS FUNCIONES?

- Crear una función nos permite darle un nombre a un grupo de sentencias, simplificando así un programa porque “escondemos” código detrás de un nombre simple e intuitivo.
- También hace a un programa más corto, ya que eliminamos código repetido, pues a una función se la puede usar cuantas veces necesitemos.



# FUNCIONES CON PARÁMETROS

- Algunas de las funciones pre-incorporadas que usamos en Python tienen parámetros, que son valores que se le proveen para que pueda realizar la tarea. Se los “pasamos” a la función, dentro de los paréntesis después del nombre.
- Por ejemplo:
  - ❖ `print(“cadena”)` en donde “cadena” es el parámetro.
  - ❖ `math.pow(base, potencia)` en donde base y potencia son los parámetros.



- A nuestras funciones también podemos definir las con parámetros.

- Por ejemplo, si quiero imprimir por pantalla el siguiente de un número:

```
def SIGUIENTE(n):
 print("el siguiente número es:", n+1)
```

- Luego en el programa principal:  
num=int(input("ingrese un número"))  
SIGUIENTE(num)

- Si n=9, el resultado será:  
El siguiente número es 10



# IMPORTANTE...

- Cuando **definimos a la función**, le damos **parámetros**, pero cuando **llamamos a la función**, le pasamos **argumentos**. En nuestro ejemplo, **n** es un parámetro, y **num** es el argumento.
- Además, el nombre de la variable que pasamos como argumento, no tiene nada que ver con el nombre del parámetro, lo que importa es respetar la ubicación tanto del argumento como del parámetro.



# EJEMPLO

- `def PERSONA(n, cadena):`  
    `print("la edad es:", n)`  
    `print("el nombre es:", cadena)`
- Entonces en nuestro programa principal debemos pasar primero la edad y luego el nombre:

`edad=int(input("Ingrese su edad"))`

`nombre=input("Ingrese su nombre")`

**`PERSONA(edad, nombre) # Correcto`**

**`PERSONA(nombre, edad) # Incorrecto`**



# FUNCIONES CON VALORES DE RETORNO

- Ahora vamos a crear funciones que generen un nuevo valor y lo devuelvan.
- Por ejemplo, queremos una función que nos calcule el área de un cuadrado, vamos a necesitar pasarle como parámetro la longitud del lado



- `def area(n):`  
    `x = n*n`  
    `return (x)`
- La sentencia **return** significa: “salí de la función, volvé a donde te llamaron y usá esta expresión como valor de retorno”.
- En nuestro programa principal:  
    `lado = int(input("Ingrese el lado del cuadrado"))`  
    `resultado = area(lado)`  
    `print("El área del cuadrado es:", resultado)`





# GUARDAR NUESTRAS PROPIAS FUNCIONES

- Para guardar nuestras propias funciones en un archivo, y poderlas llamar desde otro, debemos hacer lo siguiente:
  - **from** nombreArchivoFunciones **import\***
  - Tener en cuenta que ambos archivos deben estar guardados en la misma ubicación (el archivo con las funciones y el archivo en donde vamos a llamar a las funciones).



# EJEMPLO COMPLETO

- Realizar una función que dado un número, me devuelva el factorial de dicho número.

1) Primero creamos la función:

```
def factorial(n):
```

```
 factorial=1
```

```
 for i in range(1,n+1):
```

```
 factorial=factorial*i
```

```
 return factorial
```

**Acá estamos definiendo a la función**

2) Ahora vamos a llamarla en nuestro programa principal:

```
num=int(input("Ingrese un número natural"))
```

```
resultado=factorial(num)
```

**Acá llamamos a la función**

```
print("El factorial de", num, "es:", resultado)
```



**AHORA, A GENERAR NUESTRAS  
PROPIAS FUNCIONES!!!!**

