Unidad No.2 Lenguajes Altamente Dinámicos/débilmente Tipados

TI2011 – Introducción a la Programación

Funciones

Funciones en Python.

Objetivo:

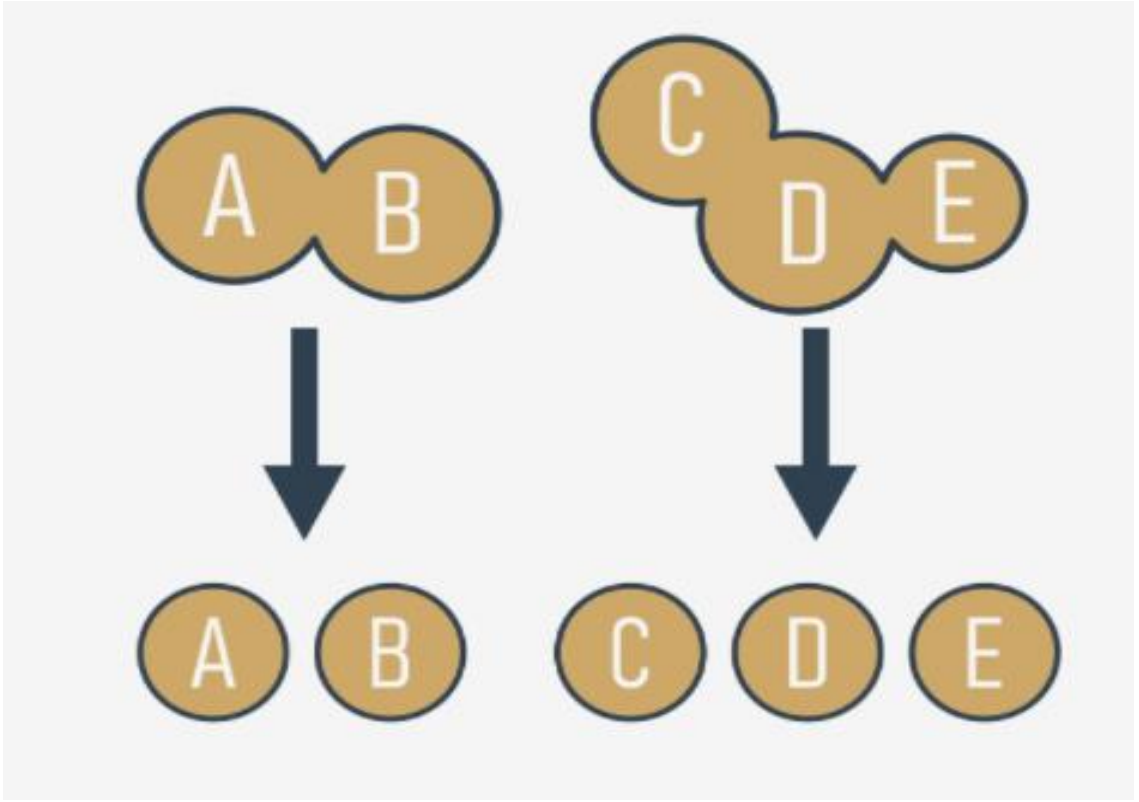
- ✓ Definir y utilizar Funciones

¿Por qué necesitamos funciones?

- ✓ Utilizamos funciones cuando queremos reutilizar varias veces una o más línea de nuestro código.
- ✓ Las funciones las utilizamos como herramientas, con el fin de hacer la vida mas fácil, y para simplificar tareas tediosas y repetitivas.
- ✓ Muy a menudo ocurre que un cierto fragmento de código **se repite muchas veces en un programa**. Se repite de manera literal o, con algunas modificaciones menores, empleando algunas otras variables dentro del programa.

Si un fragmento de código comienza a aparecer en más de una ocasión, considera la posibilidad de aislarlo en la forma de una función invocando la función desde el lugar en el que originalmente se encontraba.

¿Por qué necesitamos funciones?



Esto simplifica considerablemente el trabajo del programa, debido a que cada pieza se codifica por separado y consecuentemente se prueba por separado.

A este proceso se le llama comúnmente **descomposición**.

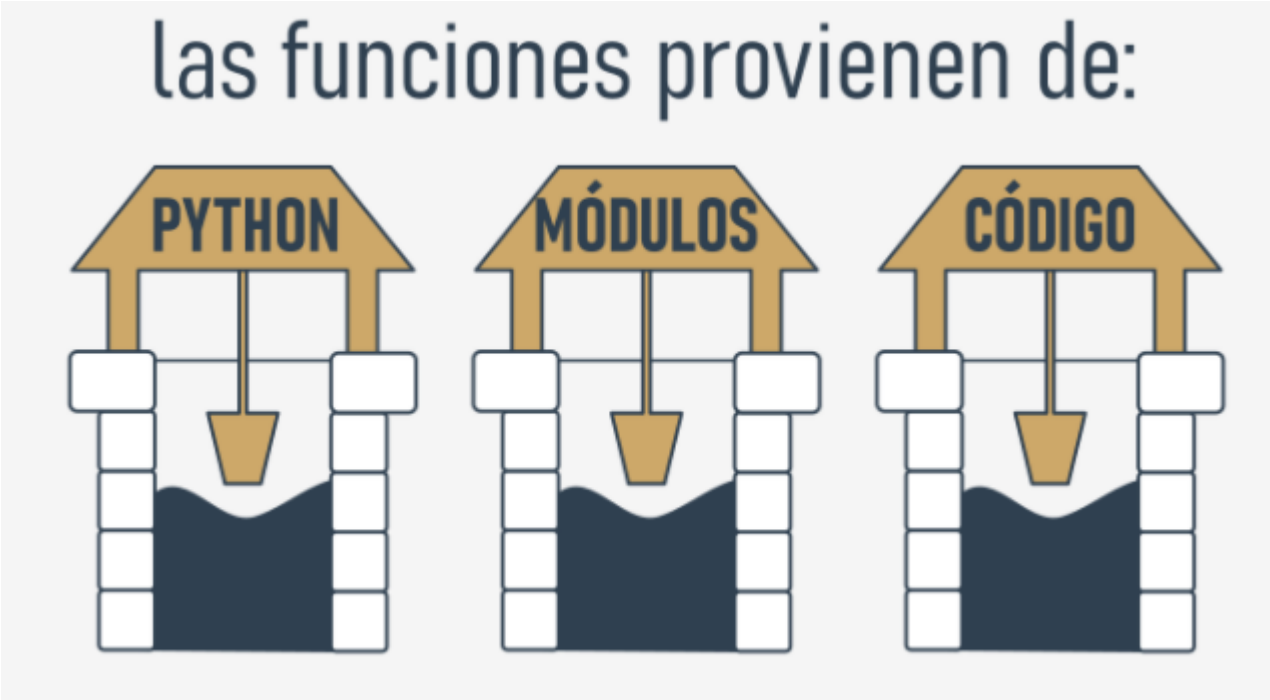
¿De dónde provienen las funciones?

En general, las funciones provienen de al menos tres lugares:

De Python mismo: varias funciones (como print()) son una parte integral de Python, y siempre están disponibles, a estas funciones se les conoce como “funciones integradas”.

De los **módulos preinstalados** de Python: muchas de las funciones, las cuales comúnmente son menos utilizadas que las integradas, están disponibles en módulos instalados juntamente con Python; para poder utilizar estas funciones el programador debe realizar algunos pasos adicionales

Directamente del código: puedes escribir tus propias funciones, colocarlas dentro del código, y usarlas libremente.



Ejemplo:

```
print("Ingresa un valor: ")
a = int(input())

print("Ingresa un valor: ")
b = int(input())

print("Ingresa un valor: ")
c = int(input())
```

El mensaje enviado a la consola por la función print() es siempre el mismo. El código es funcional y no contiene errores, sin embargo imagina tendrías que hacer debes cambiar el mensaje para que fuese mas cortés, por ejemplo, que comience con la frase "Por favor,".

¿Es posible separar ese código *repetido*, darle un nombre y hacerlo reutilizable? Significaría que **el cambio hecho en un solo lugar será propagado a todos los lugares donde se utilice**.
Para que esto funcione, dicho código debe ser invocado cada vez que se requiera.

Se comienza con la palabra reservada, def

Nombre de la función, se utiliza el mismo sistema de codificación para el nombre de variables.

•La línea debe de terminar con **dos puntos**.

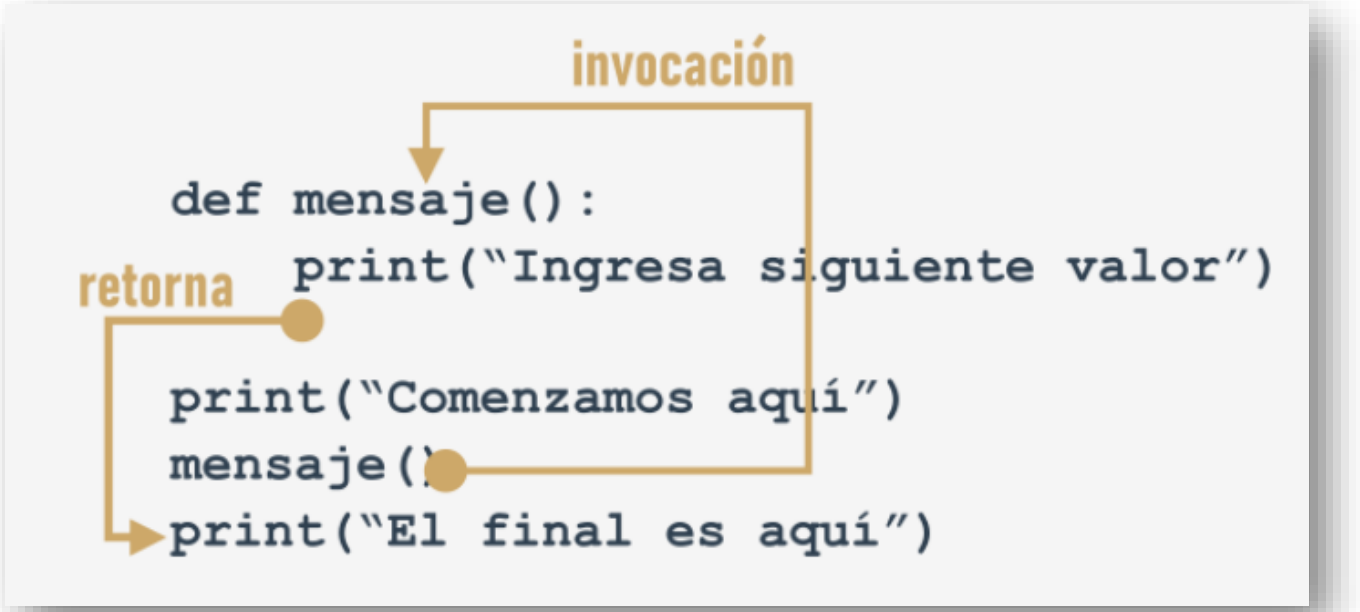
```
def NombreFunción(argumentos):
cuerpoFunción
```

cuerpo de la función - donde varias o (al menos una). **instrucción anidada**, será ejecutada cada vez que la función sea invocada; nota: la **función termina donde el anidamiento termina**,

```
def mensaje_1():
    print(" Por favor, Ingresa un valor \n")


mensaje_1()
a = int(input())

mensaje_1()
b = int(input())

mensaje_1()
c = int(input())
```

Llamada a función

```
def mensaje():
        print("Ingresa un valor: ")
print("Se comienza aquí.")
mensaje()
print("Se termina aquí.")
```



La imagen intenta mostrar el proceso completo:
•Cuando se **invoca** una función, Python recuerda el lugar donde esto ocurre y *salta* hacia dentro de la función invocada.
•El cuerpo de la función es entonces **ejecutado**.
•Al llegar al final de la función, Python **regresa** al lugar inmediato después de donde ocurrió la invocación.

Puntos Clave:

Una función es un bloque de código que realiza una tarea especifica cuando la función es llamada (invocada).

Las funciones son útiles para hacer que el código sea reutilizable, que este mejor organizado y más legible. Las funciones contienen parámetros y pueden regresar valores.

Existen al menos cuatro tipos de funciones básicas en Python:

a. **Funciones integradas:** las cuales son partes importantes de Python (como lo es la función print()).

Lista completa de las funciones integradas de Python en el siguiente link: https://docs.python.org/3/library/functions.html.

b. También están las que se encuentran en **módulos pre-instalados**.

c. **Funciones definidas por el usuario,** las cuales son escritas por los programadores para los programadores.

d. **Las funciones lambda.**

Se puede definir una función sin que haga uso de argumentos, por ejemplo:

```
def mensaje():    # definiendo una función
    print("Hola")    # cuerpo de la función

mensaje()    # invocación de la función
```

También es posible definir funciones con argumentos, como la siguiente que contiene un solo parámetro:

```
def hola(nombre):    # definiendo una función
    print("Hola,", nombre)    # cuerpo de la función


nombre = input("Ingresa tu nombre: ")

hola(nombre)    # invocación de la función
```

Ejercicio 1

La función input() es un ejemplo de:

a) Una función definida por el usuario
b) Una función integrada
c) Una función pre-instalada

Ejercicio 2

¿Qué es lo que ocurre cuando se invoca una función antes de ser definida? Ejemplo:

```
hola()

def hola():
    print("hola!")
```

R: Se genera una excepción (la excepción NameError)

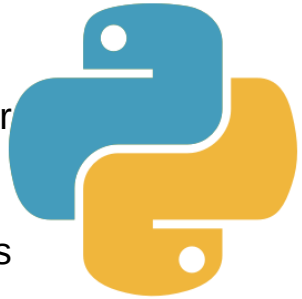
Ejercicio 3

¿Qué es lo que ocurrirá cuando se ejecute el siguiente código?

```
def hola():
    print("hola")

hola(5)
```

R: Se genera una excepción (la excepción TypeError) - la función hola() no toma argumentos.

Funciones con parámetros:



El potencial completo de una función se revela cuando puede ser equipada con una interface que es capaz de aceptar datos provenientes de la invocación.

Dichos datos pueden modificar el comportamiento de la función, haciéndola mas flexible y adaptable a condiciones cambiantes.

Un parámetro es una variable, pero existen dos factores que hacen a un parámetro diferente:

Los parámetros solo existen dentro de las funciones en donde han sido definidos, y el único lugar donde un parámetro puede ser definido es entre los paréntesis después del nombre de la función, donde se encuentra la palabra reservada def.

La asignación de un valor a un parámetro de una función se hace en el momento en que la función se manda llamar o se invoca, especificando el argumento correspondiente.

Ejemplo:

```
def mensaje(numero):
        print("Ingresa el número:", numero)
Mensaje()
```

TypeError: mensaje() missing 1 required positional argument: 'numero'

Ejercicios:

a. Construya una función que pida el nombre a un usuario, y que devuelva como mensaje “Hola + el nombre ingresado”

b. Construya una función que retorne la edad de una persona ingresando el año de nacimiento.

c. Construya una función que sume 2 números enteros.

d. Construya una función que entregue el valor absoluto de un número ingresado.

e. Construya una función que muestre el digito menor de una decena ingresada, en caso de que el número no sea de 2 dígitos o si son dígitos iguales como el 22 el programa dirá “Número no permitido” como mensaje.