Introducción a la programación

Profesor: Lic. Sergio Eduardo Torres

UNPAZ
Universidad Nacional de José C. Paz

Licenciatura en Gestión de Tecnología de la Información

Clase XII

Temario

Funciones
Funciones Recursivas

Licenciatura en Gestión de Tecnología de la Información

UNPAZ
Universidad Nacional de José C. Paz

Funciones
- **Retorno de la función**
- **Parámetros**

Funciones Recursivas

Funciones. ¿Que es una función?

Una función es un grupo de declaraciones e instrucciones (código) relacionadas que realizan una tarea específica. El objetivo de una función es **no repetir trozos** de código durante nuestro programa y **reutilizar** el código para distintas situaciones.

Sintaxis de una función

```
def nombre_Funcion( parametros ):
    ''' cadena de domumentacion '''
    instrucciones
    instrucciones
    instrucciones
    return
```

def: Palabra clave para definir funciones.

nombre de función: Sirve para identificar de forma única la función.

Parámetros: Parámetros (argumentos) a través de los cuales pasamos valores a una función. Son **opcionales**.

Dos puntos: indican el inicio del cuerpo de la función.

Cadena de documentación: Es la documentación que indica el funcionamiento de la función.

Instrucciones: Instrucciones (código) válidas que componen el cuerpo de la función, deben están indentadas.

Return: instrucción return se utiliza para devolver un valor de la función. Es **opcional**.

Funciones. ¿Cómo hago para ejecutar(llamar) a una función?

Para ejecutar , invocar o llamar a una función tendremos que escribir su nombre seguido de paréntesis debemos prestar atención si la función necesita valores (parámetros) o no. Las funciones pueden ser llamadas desde un programa u otra función.

```
def saludo():
    #Cuerpo de la función

    #Instrucciones
    print("Hola usuario, le damos la bienvenida...\n")


#Llamamos la función
saludo()
```

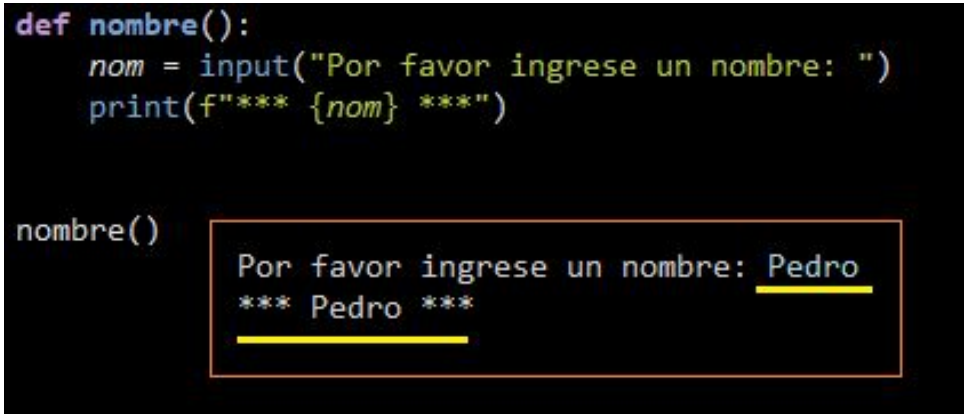
```
Hola usuario, le damos la bienvenida...
```

Funciones. Tipos de funciones

Las funciones se dividen en:

- **Funciones integradas** : funciones integradas en Python.
  - print() ,input() , nt(), str() ,float(), len(), list(), enumerate() ....
- **Funciones definidas por el usuario** : funciones definidas por los propios usuarios . Por ejemplo en el ejercicio anterior definimos la función saludo() .

Ejercicio 1: Escriba un programa que utilice una función que solicite su nombre y lo muestre por pantalla entre 3 asteriscos.



Funciones. Tipos de funciones

Las funciones se dividen en:

- **Funciones integradas** : funciones integradas en Python.
  - print() ,input() , nt(), str() ,float(), len(), list(), enumerate() ....

- **Funciones definidas por el usuario** : funciones definidas por los propios usuarios . Por ejemplo en el ejercicio anterior definimos la función saludo() .

Ejercicio 1: Escriba un programa que utilice una función que se llame nombre, que solicite su nombre y lo muestre por pantalla entre 3 asteriscos.

Ejercicio 2: Escriba un programa que utilice una función que se llame suma, que calcule la suma entre 4 y 10 , debe mostrar por pantalla el resultado.

Funciones. Solución de los ejercicios

Ejercicio 1:

```
def nombre():
    nom = input("Por favor ingrese un nombre: ")
    print(f"*** {nom} ***")


nombre()
```

```
Por favor ingrese un nombre: Pedro
*** Pedro ***
```

Ejercicio 2:

```
def suma ():
    resultado = 4 + 10
    print(f"La suma de 4 + 10 es {resultado}")


suma()
```

```
La suma de 4 + 10 es 14
```

Funciones. Ayuda de las funciones (docstrings)

Cuando creamos una función se puede agregar, la documentación o ayuda de la función, que puede indicarnos cómo funciona, si necesitas parámetros, si devuelve algún valor, dependerá de cada programador qué información a incluir aunque es opcional es una buena práctica hacer uso del docstrings.

El docstring se sitúa debajo del nombre de la función, es la primera línea del cuerpo de la función y la información se suele encerrar entre comillas triples.

Para visualizar la ayuda podemos utilizar la instrucción help(nombre de la función)

Ejemplo:

```
def saludo_con_ayuda():
    ''' Funcion para saludar al usuario '''
    print(f"Hola usuario, le damos la bienvenida...\n")

help(saludo_con_ayuda)
```

```
saludo_con_ayuda()
    Funcion para saludar al usuario
```

```
saludo_con_ayuda()
```

```
Hola usuario, le damos la bienvenida...
```

Funciones. Devolución de valores (Return). Ejercicios

La declaración return es una declaración especial que se utiliza dentro de una función o método para devolver el resultado de la función cuando es llamada.

Sintaxis: **return** **valor de retorno** que puede ser int, str, float, list, tuple, dict, set. entre otros

Puede omitir el valor de retorno de una función y usar return sin valor de retorno o también puede omitir return, En estos casos, el valor devuelto será None.

```
def suma():
    a = 45
    b = 56
    return a + b

valor = suma()
print(f"El resultado de la suma es {valor}")
```

```
El resultado de la suma es 101
```

```
def suma():
    return 45 + 56

print(f"El resultado de la suma es {suma()}")
```

```
El resultado de la suma es 101
```

```
def suma():
    resultado = 45 + 56

valor = suma()
print(f"El resultado de la suma es {valor}")
```

```
El resultado de la suma es None
```

Funciones. Devolución de valores (Return). Ejercicios

Ejercicio 1: Escribir un programa que contenga una función llamada resta que **devuelve la resta de 45 -10** , luego sume el valor devuelto a 90 y muestre el resultado final por pantalla.

```
def resta():
    return 45 - 10

valor = resta() + 90
print(f"El resultado es {valor}")
```

```
El resultado es 125
```

Ejercicio 2: Escribir un programa que contenga una función llamada saludo que muestre la palabra **hola** y devuelva la palabra **chau**, debe mostrarla fuera de la función

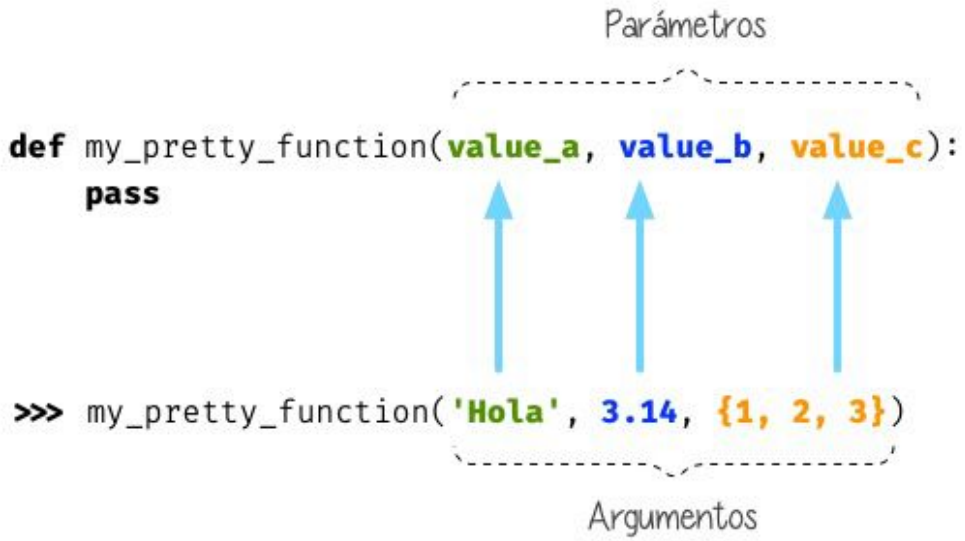
.

```
def saludo():
    print("Hola")
    return 'Chau'

texto = saludo()
print(texto)
```

```
Hola
Chau
```

Funciones. Parámetros y argumentos

Se pueden crear funciones que requieran o reciban uno o más valores, cuando se crea la función se utilizan los parámetros para luego utilizarlos en la función, cuando se llama a la función se cargan los valores (argumentos) para cumplir con los parámetros de nuestra función.



Delgado Quintero, DQS (2022). *Parámetros y argumentos de una función* , https://aprendepython.es

Funciones. Alcance de las variables

Variables globales

Cualquier variable declarada fuera de la función se denomina variable global. Tal variable tiene alcance global, es decir, es accesible desde cualquier lugar, ya sea dentro o fuera de la función.

```
def suma(valor1):

    return (valor1 + numero )

numero = 10

print(suma(8))          18

print(numero + 1)       11
```

Variables locales

Una variable declarada dentro del cuerpo de la función se denomina variable local. Se dice que una variable de este tipo tiene un alcance local, es decir, sólo es accesible dentro de la función y no fuera de ella.

```
def suma(valor1):
    numero = 20
    print(id(numero))   1862003344
    valor2 = 30
    return (valor1 + numero )
numero = 10
print(id(numero))       1862003504

print(suma(8))          28

print(numero + 1)       11

print(5 + valor2)

    print(5 + valor2)
NameError: name 'valor2' is not defined
```

Funciones. Parámetros y argumentos

Ejercicio 1 : Escriba un programa que contenga una función con 1 parámetro, llamada saludo_al_usuario, la funcion debera poder recibir una nombre (string), y luego deberá imprimir “Hola ..argumento(nombre)..buen día”

```
def saludo_al_usuario(nombre):
    print(f"Hola {nombre} buen día")

saludo_al_usuario('Alejandro')
```

```
Hola Alejandro buen día
```

Ejercicio 2 : Escriba un programa que contenga una función con 2 parámetros, llamada multiplicación , la función deberá poder recibir dos números enteros (int), y luego debe realizar la multiplicación y retornar el valor, luego debe mostrarla.

```
def multiplicacion(num1,num2):
    return num1 * num2

resultado = multiplicacion(8,6)
print(f" 8 x 6 = {resultado} ")
```

```
8 x 6 = 48
```

```
def multiplicacion(num1,num2):
    return num1 * num2

print(f" 8 x 6 = {multiplicacion(8,6)} ")
```

```
8 x 6 = 48
```

Funciones. Parámetros y argumentos. Ejercicio

Ejercicio 3: Escriba un programa permite ingresar dos números enteros por teclado, luego debe resolver la división de los dos números por medio de una función, la función deberá corroborar que los números (argumentos) no sean ceros antes de realizar la división, si los valores son correctos debe retornar el resultado de la división con comas pero si alguno de los valores es cero deberá retornar un string con la palabra Error. muestre por pantalla el resultado

```
def division(num1,num2):
    if num1 == 0 or num2 == 0:
        return 'Error'
    else:
        return num1 / num2

valor1 = int(input("Ingrese un valor: "))
valor2 = int(input("Ingrese otro valor: "))

resultado = division(valor1,valor2)
if  resultado == 'Error':
    print(f"{resultado}, ingreso un cero")
else:
    print(f"El resultado es {resultado:.2f}")
```

```
Ingrese un valor: 63
Ingrese otro valor: 2
El resultado es 31.50
```

```
Ingrese un valor: 0
Ingrese otro valor: 12
Error, ingreso un cero
```

Funciones. Parámetros y argumentos. Ejercicio

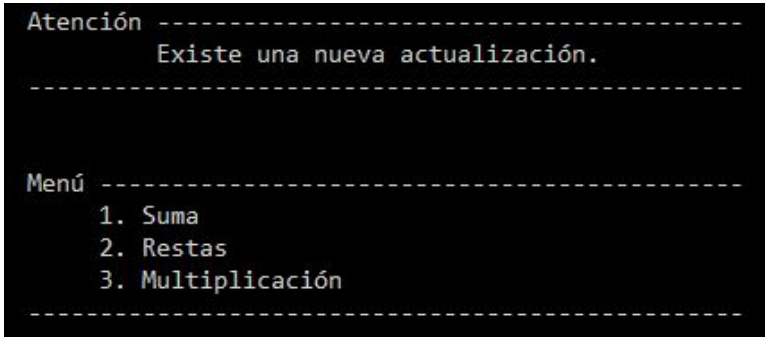
Ejercicio 4: Escriba una función para cargar un vector de n cantidad de elementos con valor 0 cero , debe retornar el vector.

Luego escriba un programa que llame a esa función y muestre el vector por pantalla

```
def crea_vector(elementos):

    import array as arr
    vector = arr.array('i',[])

    for i in range(elementos):
        vector.append(0)
    return vector

print(crea_vector(5))

print(*crea_vector(5))
```

```
array('i', [0, 0, 0, 0, 0])

0 0 0 0 0
```

Funciones. Ejemplo



Ejemplo de una función para mostrar mensajes.

Escriba una función que permita mostrar Mensajes:
la función debe:

1. Tener dos parámetros uno para el Título del aviso y otro para el mensaje a transmitir.
2. El título debe ser un string que no supere los 10 caracteres, si supera debe mostrar el mensaje "Mensaje incorrecto, El título no debe superar los 10 caracteres y el mensaje no puede superar los 50 caracteres",
   El título debe ser mostrado a partir de 20 espacios luego viene el título un espacio y termina con guiones hasta alcanzar los 50 caracteres.
3. El mensaje ser una Lista dentro puede contener cualquier tipo de dato string numero float.
   Deberá recorrer la lista si tienen un solo elemento mostrará el mensaje centrado entre 50 caracteres en total
   pero si tiene más elementos deberá recorrer la lista y mostrar cada elemento uno debajo de otro, alineado a la izquierda y a una distancia de 25 espacios desde el margen izquierdo.
4. El mensaje debe finalizar con 50 guiones, que se empezaran a mostrar a partir 20 espacios desde el margen izquierdo
5. Esta función sólo tiene retorno al encontrar un error sino imprime el mensaje desde la función

Funciones. Ejemplo

```python
def mensajes(titulo,mensaje):
    largo_menu = len(titulo)

    superior= 49 - largo_menu

    if len(titulo) > 10:
        print("Mensaje incorrecto, El titulo no debe superar los 10 caracteres")
        return

    print(f"\n{' ':20s}{titulo} {'-' * superior}")

    elementos = len(mensaje)
    for i in range(elementos):
        if elementos == 1:
            print(f"{' ':20s}{mensaje[i]:^50s}")
        else:
            print(f"{' ':25s}{mensaje[i]}")

    print(f"{' ':20s}{'-' * 50}\n")


mensajes('Atención', ['Existe una nueva actualización.'])
mensajes('Menú', ['1. Suma ','2. Restas','3. Multiplicación'])
mensajes('Atención', ['Se acerca el 2do parcial'])
```

```
Atención ----------------------------------------
         Existe una nueva actualización.
--------------------------------------------------
```

```
Menú --------------------------------------------
     1. Suma
     2. Restas
     3. Multiplicación
--------------------------------------------------
```

```
Atención ----------------------------------------
             Se acerca el 2do parcial
--------------------------------------------------
```

Introducción a la programación

Licenciatura en Gestión de Tecnología de la Información