Universidad Peruana Cayetano Heredia
Facultad de Ciencias y Filosofía
Departamento de Ciencias Exactas

Computación 2022 Verano

Lab. 4.3. Funciones en Python. Parte I.

Enero 26, 2022

Cree una carpeta (folder) en el disco D, nómbrela con su apellido paterno seguido de su código. Ejemplo: LOPEZ12345

Funciones en Python

Las ***funciones*** son un conjunto de instrucciones que realizan una tarea determinada. Dichas instrucciones agrupadas forman una unidad (entidad, caja negra) a la cual se le da un nombre.

Podemos enumerar varias razones por la que usamos ***funciones*** en nuestros programas, pero por ahora nos concentraremos en dos de sus roles más importantes: la reutilización del código y el diseño descendente.

Maximizando la reutilización del código y minimizando la redundancia.

Cuando se escribe programas grandes de cientos de líneas de código, aparecen varias veces, a lo largo del programa, porciones de código que realizan una misma tarea determinada. También sucede que ciertas tareas ya están codificadas en otros programas. En ambos casos la porción de código que realiza una tarea determinada, es mejor encapsularlo y convertirlo en una unidad llamada **función**. Esto permite usar cuantas veces queramos y en cualquier lugar de nuestros programas, inclusive pueden ser usados por otros programas. De esta manera se logra reutilizar el código, evitar las redundancias y por consiguiente aumentar la productividad del programador y facilitar el mantenimiento.

Diseño descendente, descomposición estructural

Una manera de luchar con la complejidad de los problemas es usar la técnica "divide y vencerás", es decir el diseño descendente (top-down design). La implementación de esta técnica nos lleva a dividir un programa en varias piezas llamadas ***funciones.*** Codificar por separado cada una de las partes de un programa, es más sencillo que codificar todo el programa a la vez.

Debido a que las funciones forman una unidad (caja negra), para poder usarlos tenemos que establecer una comunicación con ellas a través de una **interfaz**. Es decir que las funciones deben poseer una **interfaz** (medio de comunicación) con el mundo exterior. La estructura de la **interfaz** es definida por la sintaxis del lenguaje de programación, en este caso Python.

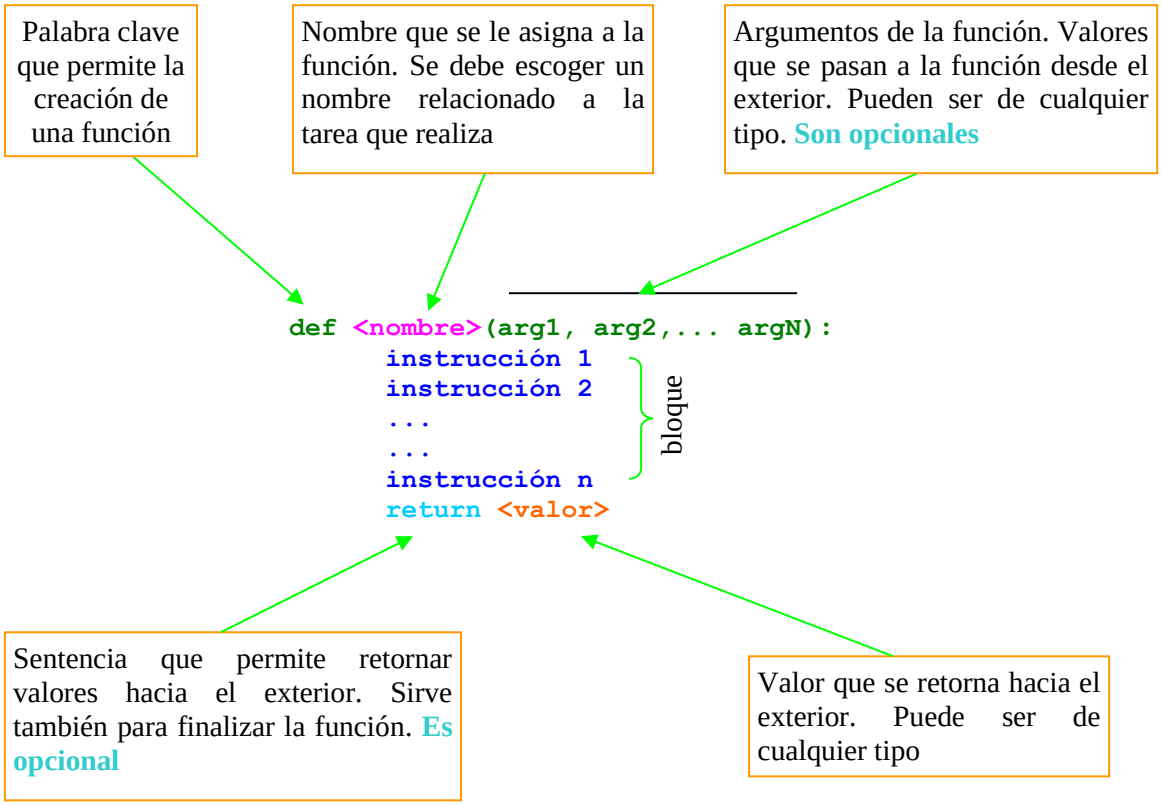
Debemos tener en cuenta, además, que las instrucciones, datos y variables contenidas en el cuerpo de las funciones, no son conocidas en el exterior, ni tampoco se puede tener acceso a ellos. La única forma de comunicarnos con las funciones es a través de su interfaz.

Cómo definir e invocar funciones simples

Para usar una función, primero tenemos que definirla y luego invocarla.

Definición de una función

La definición se realiza usando la palabra clave **def**. Esta palabra clave permite crear una función, sin embargo, exige que la asignemos un nombre y proporcionemos los argumentos. El formato general es la siguiente:



Ejemplo 1: El siguiente programa contiene una función que devuelve el producto de dos números. Corra el programa y analice cómo se define una función y la forma de invocarla (llamarla).

```
# Ejemplo 4.3.1.py
# Creación y uso de una función definida por el usuario

# Definición de la función:
def square( x,y ):
    z=x*y
    return z
# ********* Fin de la función ************
a=int(input("ingrese un valor:"))
b=int(input("ingrese un valor:"))

p=square( a,b )

print( p )
```

Invocación de la función. Se invoca colocando el nombre de la función y pasándole los parámetros si son necesarios. El valor devuelto se guarda en la variable p.

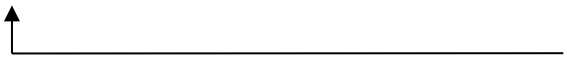
¿Qué sucede si colocamos `print(z)` después de `print(p)`?, ¿por qué?

Ejemplo 2: El siguiente programa modifica el ejemplo 1. El valor retornado no se guarda en otra variable, se imprime directamente.

```
# Ejemplo 4.3.2.py
# Creación y uso de una función definida por el usuario

# ********* Definición de la función: ***************
def square( x,y ):
    z=x*y
    return z
# ********* Fin de la function **********************
a=int(input("ingrese un numero para a:"))
b=int(input("ingrese un numero para b:"))

print(square( a,b ))
```



Aquí se invoca la función y el valor devuelto se imprime directamente.

Ejemplo 3: El siguiente programa modifica el ejemplo 1 para imprimir el cuadrado de los números del 1 al 10 inclusive.

```
# Ejemplo4.3.3.py
# Creación y uso de una función definida por el usuario

# Definición de la función:
def square( x ):
    z=x*x
    return z

for c in range( 1, 11 ):
    p = square( c )
    print(p)
```

Antes de iniciar con los ejercicios, responder:

a) ¿En qué lugar se define una función, antes de la invocación, después de la invocación o en cualquier lugar?
b) ¿Es necesario la instrucción return? ¿qué usos tiene?
c) ¿Una función siempre tiene argumentos?
d) ¿Se puede definir más de una función en un programa?
e) Las variables que están dentro de una función, ¿pueden ser usados fuera de la función?
f) ¿Cuántas veces se puede invocar una función?

Para cada uno de los siguientes ejercicios implemente el programa Python correspondiente. Guarde vuestros programas en vuestra carpeta de trabajo.

1. Implemente una función que reciba tres valores diferentes y determine el mayor de ellos. Escriba un programa para probar la función.
   a) la función debe devolver el valor encontrado
   b) la función no devuelve el valor encontrado, lo imprime directamente en pantalla

2. Implemente las siguientes funciones:
   a) Una función llamada `left` que elimine los espacios sobrantes al comienzo de un texto.
   b) Una función llamada `right` que elimine los espacios sobrantes al final de un texto.
   c) Una función llamada `clean` que elimine todos los espacios sobrantes en un texto, del comienzo, del final y entre palabras y palabras. La función debe de usar las funciones, `left` y `right`.
   d) Una función llamada `count` que reciba una cadena y determine la cantidad de palabras. Previamente la función debe de "limpiar" la cadena usando las funciones recién implementadas.

   Implemente un programa para probar las funciones.

```
Probando la función, left:
Ingrese un texto:   ␣␣␣␣␣Hola␣␣␣␣como␣␣estas␣␣␣␣
Longitud del texto inicial: 25
Texto resultante:   Hola␣␣␣␣como␣␣estas␣␣␣␣
Longitud del texto resultante: 21

Probando la función, right:
Ingrese un texto:   ␣␣␣␣␣Hola␣␣␣␣como␣␣estas␣␣␣␣
Longitud del texto inicial: 25
Texto resultante:   ␣␣␣␣␣Hola␣␣␣␣como␣␣estas
Longitud del texto resultante: 22

Probando la función, clean:
Ingrese un texto:   ␣␣␣␣␣Hola␣␣␣␣como␣␣estas␣␣␣␣
Longitud del texto inicial: 25
Texto resultante:   Hola␣como␣estas
Longitud del texto resultante: 15

Probando la función, count:
Ingrese un texto:   ␣␣␣␣␣Hola␣␣␣␣como␣␣estas␣␣␣␣
La cantidad de palabras del texto es: 3
```

3. Implemente las siguientes funciones:

   a) Una función que determine la factorial de un número entero positivo, considere 0!=1.

   b) Una función que determine la potencia de un número de la forma $\mathbf{x^n}$ usando multiplicación sucesiva, considere exponentes negativos.

4. Crea un módulo que incorpore las funciones del ejercicio N° 3.
   Implemente una función que demuestre, para cualquier **a**, que $a^n < n!$ para **n** lo suficientemente grande (muestre en forma tabular los valores de a^n y **n!**)
   La función debe de invocar las funciones del módulo usando las distintas técnicas de importación de funciones de un archivo a otro archivo.

   a) Usando la forma:

   ```
   from nombreArchivo import nombreFunccion     #(pruebe en un nuevo archivo)
   ```

   b) Usando la forma:

   ```
   from nombreArchivo import *                  #(pruebe en un nuevo archivo)
   ```

   c) Usando las formas:

   ```
   import nombreArchivo                         #(pruebe en un nuevo archivo)
   import nombreArchivo as alias                #(pruebe en un nuevo archivo)
   ```

5. Dado el valor del número X. Determinar la suma de los N primeros términos de la siguiente serie. Usar las funciones implementadas previamente según corresponda.

$$S = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \cdots$$

OPCIONAL:

6. Implemente una función que reciba un número binario (en forma de string) y devuelva su equivalente en decimal.

Guarde todos vuestros programas y vuestra hoja de respuestas en una carpeta con el nombre su **Apellido** paterno seguido de vuestro **DNI**, luego comprima esta carpeta. Envíe este archivo a victor.melchor.e@upch.pe , especificando como asunto **Lab4.3**.