Curso Introducción a la programación con Python3

Programación estructurada

OpenWebinars

Programación estructurada

Con la **programación modular y estructura**, un problema complejo debe ser dividido en varios **subproblemas más simples**, y estos a su vez en otros subproblemas más simples. Esto debe hacerse hasta obtener subproblemas lo **suficientemente simples** como para poder ser resueltos fácilmente con algún algoritmo (divide y vencerás). Además el uso de subrutinas nos proporciona la **reutilización del código** y no tener **repetido instrucciones** que realizan la misma tarea.

```
>>> def factorial(n):
...     """Calcula el factorial de un
número"""
...     resultado = 1
...     for i in range(1,n+1):
...         resultado*=i
...     return resultado
```

```
>>>
factorial(5)
120
```

Para **llamar a una función** se debe utilizar su nombre y entre paréntesis los **parámetros reales** que se mandan. La llamada a una función se puede considerar una expresión cuyo valor y **tipo** es el retornado por la función.

Ámbito de variables

Variables locales: se declaran dentro de una función y no se pueden utilizar fuera de esa función

```
>>> def operar(a,b):
...  suma = a + b
...  resta = a - b
...  print(suma,resta)
...
>>> operar(4,5)
9 -1
>>> resta
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'resta' is not defined
```

Variables globales: son visibles en todo el módulo.

```
>>> PI = 3.1415
>>> def area(radio):
...  return PI*radio**2
...
>>> area(2)
12.566
```

Parámetros formales y reales

```
def CalcularMaximo(num1,num2):
    if num1 > num2:
        return num1
    else:
        return num2
```

Parámetros formales: Son las variables que recibe la función, se crean al definir la función.

```
numero1 = int(input("Dime el número1:"))
numero2 = int(input("Dime el número2:"))
num_maximo = CalcularMaximo(numero1,numero2)
print("El máximo es ",num_maximo;)
```

Parámetros reales: Son la expresiones que se utilizan en la llamada de la función, sus valores se “copiarán” en los parámetros formales.

Paso de parámetro por valor o por referencia

En Python el **paso de parámetros es siempre por referencia**. El lenguaje no trabaja con el concepto de variables sino objetos y referencias.

Si se pasa un valor de **un objeto inmutable**, su valor **no se podrá cambiar** dentro de la función.

```
>>> def f(a):
...  a=5
>>> a=1
>>> f(a)
>>> a
1
```

Sin embargo si pasamos **un objeto de un tipo mutable**, **si podremos cambiar** su valor:

```
>>> def f(lista):
...   lista.append(5)
...
>>> l = [1,2]
>>> f(l)
>>> l
[1, 2, 5]
```

Devolución de información
Una función en python puede devolver información utilizando la instrucción **return**. La instrucción **return** puede devolver cualquier tipo de resultados, por lo tanto **es fácil devolver múltiples datos guardados en una lista, tupla o diccionario.**

Aunque podemos cambiar el parámetro real cuando los objetos pasados son de tipo mutables, **no es recomendable hacerlo en Python.**

LLamadas a una función

Cuando se **llama a una función** se tienen que indicar los **parámetros reales** que se van a pasar. **La llamada a una función se puede considerar una expresión cuyo valor y tipo es el retornado por la función.** Si la función no tiene una instrucción **return** el tipo de la llamada será **None**.

```
>>> def cuadrado(n):
...    return n*n

>>> a=cuadrado(2)
>>> cuadrado(3)+1
10
>>> cuadrado(cuadrado(4))
256
>>> type(cuadrado(2))
<class 'int'>
```

Una **función recursiva** es aquella que al ejecutarse hace **llamadas a ella misma**. Por lo tanto tenemos que tener **"un caso base"** que hace terminar el bucle de llamadas. Veamos un ejemplo:

```
def factorial(num):
     if num==0 or num==1:
          return 1
     else:
          return num * factorial(num-1)
```