Análisis de datos ambientales con Python I

Marvin J. Quispe Sedano

Mayo 2021

Contenido

- Operadores relacionales y lógicos
- Declaraciones (Python's Statements)
- Condicionales y Loops
- Funciones y métodos

Operadores relacionales

- Los operadores relacionales sirven para comparar o establecer una relación entre objetos, devolviendo un resultado booleano (True o False) según el comparativo realizado.

Operador	Descripción	Ejemplo
==	¿son iguales a y b?	`>>> 5 == 3` `False`
!=	¿son distintos a y b?	`>>> 5 != 3` `True`
<	¿es a menor que b?	`>>> 5 < 3` `False`
>	¿es a mayor que b?	`>>> 5 > 3` `True`
<=	¿es a menor o igual que b?	`>>> 5 <= 5` `True`
>=	¿es a mayor o igual que b?	`>>> 5 >= 3` `True`

Operadores lógicos

- Los operadores lógicos nos permiten hacer comparativos entre valores de tipo booleano (True o False).

Operador	Descripción	Ejemplo

a and b	¿se cumple a y b?	5 == 5 and b==1 # b=1 ... verdadero
		z == 8 and 10==10 # z=9 ... falso

a or b	¿se cumple a o b?	5 == 5 or b==1 # b=8 ... verdadero
		z == 7 and 10==8 # z=7 ... falso

not a	No a	not (b == 1) # b=5 ... verdadero not (10==10) # falso

Declaraciones (Python's Statements)

- las declaraciones, en términos simples, son las líneas de código que permiten decirle a Python lo que sus programas deberían hacer.

Statement	Role	Example
Assignment	Creating references	`a, *b = 'good', 'bad', 'ugly'`
Calls and other expressions	Running functions	`log.write("spam, ham")`
print calls	Printing objects	`print('The Killer', joke)`
if/elif/else	Selecting actions	`if "python" in text:` `  print(text)`
for/else	Sequence iteration	`for x in mylist:` `  print(x)`
while/else	General loops	`while X > Y:` `  print('hello')`
pass	Empty placeholder	`while True:` `  pass`
break	Loop exit	`while True:` `  if exittest(): break`
continue	Loop continue	`while True:` `  if skiptest(): continue`
def	Functions and methods	`def f(a, b, c=1, *d):` `  print(a+b+c+d[0])`
return	Functions results	`def f(a, b, c=1, *d):` `  return a+b+c+d[0]`
yield	Generator functions	`def gen(n):` `  for i in n: yield i*2`

global	Namespaces	`x = 'old'`
		`def function():`
		`  global x, y; x = 'new'`

nonlocal	Namespaces (3.0+)	`def outer():`
		`  x = 'old'`
		`  def function():`
		`  nonlocal x; x = 'new'`

import	Module access	`import sys`
from	Attribute access	`from sys import stdin`

class	Building objects	`class Subclass(Superclass):`
		`  staticData = []`
		`  def method(self): pass`

Declaraciones: if, else y elif

- La declaración "if", "else" y "elif" se utilizan para ejecutar un bloque de código si se cumple una determinada condición y otras instrucciones si no se cumple con dicha condición.

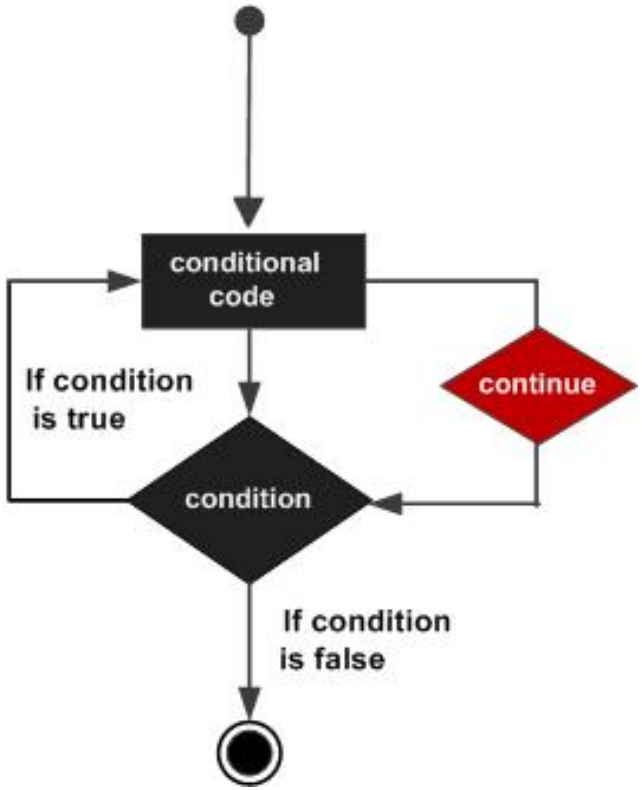
```
if EXPRESION:
    STATEMENTS
```

```
if = « SÍ »
else = « DE LO CONTRARIO»
elif = « DE LO CONTRARIO, SI »
```

- Los dos puntos (:) son significativos y obligatorios.
- Las líneas de código después de los dos puntos deben tener sangría.
- Las líneas de código con sangría se ejecutan después que la expresión sea verdadera.

Condicionales y Loops

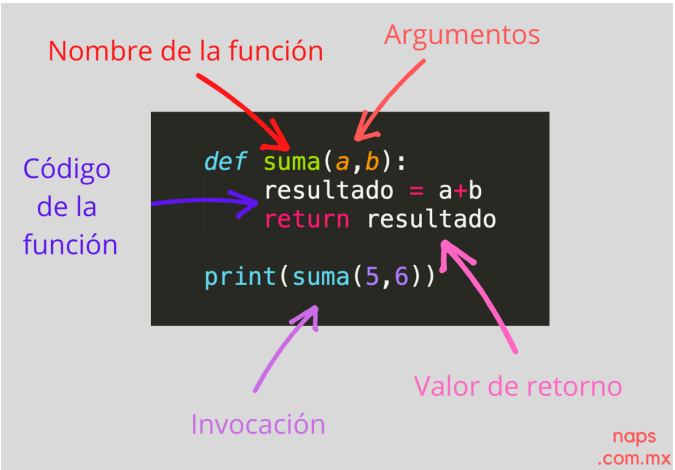
- Los Loops o "bucles" son la manera de repetir alguna función o líneas de código, mientras se cumplan determinadas condiciones.



Funciones y métodos

- Una función constituye un fragmento de código, que se llama por su nombre, y que se ejecuta en un orden determinado. Una función opera con datos de entrada (parámetros) y puede devolver un valor de retorno.

- Un método es un fragmento de código, que se llama por un nombre, y que está asociado con un objeto. Un método puede operar con datos que están contenidos dentro de la clase o tipo de objeto.



GRACIAS