10 - Iteracions for

October 30, 2015

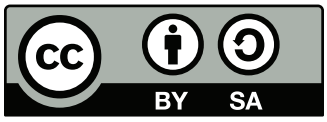

Figure 1: BY-SA

Authors : *Sonia Estradé*
José M. Gómez
Ricardo Graciani
Frank Güell
Manuel López
Xavier Luri
Josep Sabater

1 Sentència `for`

Les sentències `for` permeten executar repetidament un bloc de sentències un número de vegades controlat pel contingut d'una llista. La sintaxi bàsica és:

```
for variable_de_for in llista:
    sentència 1
    sentència 2
    ...
```

Les sentències del bloc de codi associat s'executen una vegada per a cada valor en la llista, i en cada execució la variable associada al bucle `variable_de_for` pren successivament cadascun dels valors de la llista.

Exemple bàsic de sentència *for*:

```
In [1]: # Definim una llista, en aquest cas amb valors enters
        llista = [5,2,3,4,9,6]

        # Fem un bucle for que recorri la llista; en cada iteració
        # la variable "valor" pren consecutivament cadascun dels
        # valors de la llista
        for valor in llista:
            print("Valor: " + str(valor))

Valor: 5
Valor: 2
Valor: 3
```

```
Valor: 4
Valor: 9
Valor: 6
```

```
In [2]: # La llista pot ser de qualsevol tipus, no només numèrica
        llista = ["gat","gos","ratoli"]
        for valor in llista:
            print ("Valor: " + valor)
```

```
Valor: gat
Valor: gos
Valor: ratoli
```

1.1 La funció `range()`

Molt freqüentment es necessita usar un bucle `for` per a iterar sobre una seqüència numèrica regular, per exemple:

```
0,1,2,3,4,5,6,7,8,9
0,2,4,6,8,10
```

```
In [3]: # La usem en un for
        for valor in range(0,10):
            print(valor, valor**2, valor**3)
```

```
0 0 0
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
```

Tanmateix, podem crear una llista y fer iteracions amb la llista:

```
In [4]: lista = list(range(10))
        print(lista)
        print(range(10))
        # Exemple amb range()
        for valor in lista:
            print(valor, valor**2, valor**3)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(0, 10)
0 0 0
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
```

1.2 Index i valors: La funció `enumerate()`

Usant la funció `enumerate()` es pot aconseguir que un bucle `for` ens doni simultàniament el valor de cada element d'una llista i l'índex corresponent. Això és útil per a recórrer llistes controlant al mateix temps l'índex en curs

```
In [5]: llista = ["gat","gos","ratoli"]

        # Usem enumerate per a fer un bucle for dual
        for index, valor in enumerate(llista):
            print(index,valor)

        # Usem enumerate per a fer un bucle for dual
        for enum in enumerate(llista):
            print(enum)
```

```
0 gat
1 gos
2 ratoli
(0, 'gat')
(1, 'gos')
(2, 'ratoli')
```

Això es pot aconseguir de forma alternativa usant el bucle per recórrer el rang d'índexs:

```
In [6]: llista = ["gat","gos","ratoli"]

        # Usem el for per recórrer tots els índexs
        for index in range(len(llista)):
            print(index,llista[index])
```

```
0 gat
1 gos
2 ratoli
```

1.3 Recorrent multiples llistes: `zip()`

A vegades és necessari recórrer dues llistes simultàniament. Això es pot aconseguir amb un bucle `for` que recorri l'índex de la llista més curta o bé amb la funció `zip()`.

```
In [7]: llista1 = ["A","B","C"]
        llista2 = ["1","2","3","4","5","6"]

        # Usem zip() per recórrer les varies llistes a la vegada. La funció
        # zip() retorna tuples de valors de les llistes i acaba quan una
        # d'elles no té més elements
        for valor1, valor2 in zip(llista1,llista2):
            print(valor1, valor2)

        print()
        # Podem fer-ho també recorrent els índexs de la llista més curta
        for index in range(len(llista1)):
            print(llista1[index], llista2[index])
```

```
A 1
B 2
```

```
C 3

A 1
B 2
C 3
```

1.4 Sentència `for` amb diccionaris, tuples i cadenes

Les sentències for també funcionen amb diccionaris, tuples i amb una cadena. En aquest ultim cas s'iteren els caràcters de la cadena.

```
In [8]: # Exemple amb diccionari
        preu = {'cafe':1, 'donut':2, 'refresc':1.5, 'entrepa':3}
        # element recorre els valors de les claus del diccionari
        for element in preu:
            print("Element de diccionari: " + element + " " + str(preu[element]))

        # Exemple amb cadena
        text= "abcdef"
        for caracter in text:
            print("Caracter: " + caracter)

        print(preu.keys())
        print(preu.values())
        print(preu.items())
```

```
Element de diccionari: entrepa 3
Element de diccionari: donut 2
Element de diccionari: refresc 1.5
Element de diccionari: cafe 1
Caracter: a
Caracter: b
Caracter: c
Caracter: d
Caracter: e
Caracter: f
dict_keys(['entrepa', 'donut', 'refresc', 'cafe'])
dict_values([3, 2, 1.5, 1])
dict_items([('entrepa', 3), ('donut', 2), ('refresc', 1.5), ('cafe', 1)])
```

Noteu que un diccionari no té un ordre definit. Quan utilitzem un diccionari en un bucle `for`, aquest itera sobre totes les `keys`.

També podem iterar sobre els valors

```
In [9]: #  Definim el diccionari
        preus = {'cafe':1, 'donut':2, 'xocolata':1.5, 'entrepa':3}

        for value in preus.values():
            print("El preu  es: {}".format(value))
```

```
El preu  es: 3
El preu  es: 2
El preu  es: 1.5
El preu  es: 1
```

O podem iterar explicitament sobre els parells (`key`, `value`) que formen el diccionari:

```
In [10]: #  Definim el diccionari
         preus = {'cafe':1, 'donut':2, 'xocolata':1.5, 'entrepa':3}

         for clau, value in preus.items():
             print("El preu del {} es: {}".format(clau, value))
```

```
El preu del entrepa es: 3
El preu del donut es: 2
El preu del xocolata es: 1.5
El preu del cafe es: 1
```

Nota important: Quan operem sobre seqüències mutables com llistes o diccionaris que poden ser modificades dintre de un loop, es recomana fer primer una copia d'aquest ja que una modificació de la llista dintre del bucle implica una modificació de la llista inicial.

```
In [11]: #  Tenim la llista...
         animals = ['gat', 'gos', 'gos']

         for animal in animals:
             if animal =='gos':
                 animals.remove(animal)
             print('animal de la llista: ', animal)
         print (animals)
```

```
animal de la llista:  gat
animal de la llista:  gos
['gat', 'gos']
```

En el cas de les llistes utilitzades en bucles `for` es pot fer servir slicing. Fent servir aquesta notació, list[:] genera una nova llista amb els items seleccionats, tots en aquest cas concret.

```
In [12]: #  Tenim la llista...
         animals = ['gat', 'gos', 'gos']
         print(animals)
         for animal in animals[:]:
             if animal =='gos':
                 animals.remove(animal)
             print('animal de la llista: ', animal)
         print (animals)
```

```
['gat', 'gos', 'gos']
animal de la llista:  gat
animal de la llista:  gos
animal de la llista:  gos
['gat']
```

En el cas dels diccionaris, les funcions `nom.keys()`, `nom.values()` o `nom.items()` proporcionen objectes que ens permeten iterar sobre les `keys`, `values` o tuples del tipus (`key,value`)

```
In [13]: diccionari = {'cafe':1, 'donuts':4, 'xocolatines':3}
         llista_claus = diccionari.keys()
         llista_valors = diccionari.values()
         tupla_diccionari = diccionari.items()
         print(llista_claus, type(llista_claus))
         print(llista_valors, type(llista_valors))
         print(tupla_diccionari, type(tupla_diccionari))
```

```
dict_keys(['donuts', 'xocolatines', 'cafe']) <class 'dict_keys'>
dict_values([4, 3, 1]) <class 'dict_values'>
dict_items([('donuts', 4), ('xocolatines', 3), ('cafe', 1)]) <class 'dict_items'>
```

1.5 Seqüències de valors float

Per generar seqüències de valors `float` caldrà usar la llibreria `numpy` que estudiarem més endavant. En particular la funció

```
numpy.arange([start], stop[, step], dtype=None)
```

Exemple:

```
In [14]: # Usem arange per a generar valors float entre 0 i 1 amb un pas de 0.1
         import numpy
         for valor in numpy.arange(0.,1.,0.1):
             print(valor)
```

```
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
```

1.6 `break`: interrupció de l'execució d'un bloc

La comanda `break` permet interrompre l'execució d'un bloc `for`. Quan s'executa aquesta comanda l'execució salta el què quedi del bloc for i segueix amb el programa.

Exemple:

```
In [15]: # Interrompem aquest bucle a la meitat amb la comanda "break"
         print("El programa arriba al bucle for")
         for valor in range(10):
             if valor < 5 :
                 print("Valor: " + str(valor))
             else:
                 print("Trobat valor major que 5: interrompent")
                 break

         print("Seguim amb el programa")
```

```
El programa arriba al bucle for
Valor: 0
Valor: 1
Valor: 2
Valor: 3
Valor: 4
Trobat valor major que 5: interrompent
Seguim amb el programa
```

1.7 `break-else`

La comanda `break` es pot combinar amb la sentència `else`. En un bucle `for` el bloc de codi associat a la sentència `else` s'executa al final del bucle si el bucle no s'ha interromput mitjançant un `break`.

Exemple:

```
In [16]: # Exemple 1: el bucle no s'interromp amb break
         llista = [1,1,2,3,3,4]
         for valor in llista:
             print(valor)
             if valor > 5 :
                 break
         else:
             print("No s'ha interromput el bucle for")

         # Exemple 2: el bucle s'interromp amb break
         llista = [1,1,6,3,3,4]
         for valor in llista:
             print(valor)
             if valor > 5 :
                 break
         else:
             print("No s'ha interromput el bucle for")
```

```
1
1
2
3
3
4
No s'ha interromput el bucle for
1
1
6
```

1.8 `continue`: saltant una iteració del bloc `for`

La sentència `continue` permet interrompre la iteració del bloc `for` en curs i saltar directament a la iteració següent.

Exemple:

```
In [17]: # En aquest exemple usem continue per saltar els valors parells del bucle
         for valor in range(10):
             if valor%2==0 :
                 continue
             print("Saltant valors parells: "+str(valor))
```

```
Saltant valors parells: 1
Saltant valors parells: 3
Saltant valors parells: 5
Saltant valors parells: 7
Saltant valors parells: 9
```

Exemple: ---

Escriurem el codi necessari per aconseguir que l'ordinador:

1. Imprimeixi per pantalla el quadrat de tots els enters entre -5 i +5.
2. Demani a l'usuari una cadena i la imprimeixi per pantalla, però cada lletra en majúscules i en una fila diferent.
3. Demani a l'usuari una cadena i la imprimeixi per pantalla, en ordre invers i en una fila diferent.
4. Demani a l'usuari 3 numeros enters i els guardi directament en una llista.

```
In [18]: for x in range(-5,6):
             print (x**2)

25
16
9
4
1
0
1
4
9
16
25

In [19]: cadena = input("cadena: ")
         for lletra in cadena:
             print (lletra.upper())

         print(cadena)

cadena: cadena
C
A
D
E
N
A
cadena

In [20]: cadena = input("cadena: ")

         for lletra in cadena[::-1]:
             print (lletra," ")

         print (cadena)

cadena: cadena
a
n
e
d
a
c
cadena

In [21]: llista_enters = []
         for i in range(3):
             llista_enters.append(int(input("Numero enter? ")))
         print (llista_enters)

Numero enter? 23
Numero enter? 45
Numero enter? 13
[23, 45, 13]
```

2 Comprensió de llistes

La comprensió de llistes consisteix en optimitzar la utilització de llistes i bucles `for` i clàusules `if`. El resultat dona una menor quantitat de línies de codi i una optimització en l'ús de la memòria. Consisteix en definir després del primer bracket l'expresió que volem assolir seguida de sentències `for` i `if` i tot seguit la llista sobre la que operem. El resultat dona una nova llista. Veiem un exemple:

```
In [22]: # A partir de la llista A definida com:

         A = list(range(10))

         #  Ens demanen una nova llista que faci el quadrat de cada valor de A

         quadrat_de_A = [x**2 for x in A]
         A, quadrat_de_A
```

```
Out[22]: ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 4, 9, 16, 25, 36, 49, 64, 81])
```

```
In [23]: #  Altres exemples...
         [(x,y) for x in [1,2,3] for y in [3,1,4] if x!=y]
```

```
Out[23]: [(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

Podem també operar llistes multidimensionals:

```
In [24]: matrix = [[1,2,3],
                   [4,5,6],
                   [9,8,7]]

         matriu_quadrada = [[dada**2 for dada in row] for row in matrix]

         print(matrix)
         print(matriu_quadrada)
```

```
[[1, 2, 3], [4, 5, 6], [9, 8, 7]]
[[1, 4, 9], [16, 25, 36], [81, 64, 49]]
```

Exemple: ___ Escriurem un programa que calculi el número

$$\pi$$

amb les fórmules següents:

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

$$\prod_{n=1}^{\infty} \left(\frac{2n}{2n-1} \cdot \frac{2n}{2n+1} \right) = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdots = \frac{\pi}{2}$$

```
In [25]: M = 25
         suma = 0.
         for n in range(M):
             numerador = (-1.)**n
             denominador = ((2.*n)+1)
             suma = suma + (numerador/denominador)
             print (suma)
```

```
        print("PI = {}".format(suma*4))

        # Amb Comprensió de llistes
        pi = sum([(-1.)**n/((2.*n)+1) for n in range(M)])*4

        print( "\nPI = {}".format(pi))
```

```
1.0
0.6666666666666667
0.8666666666666667
0.7238095238095239
0.8349206349206351
0.7440115440115441
0.8209346209346211
0.7542679542679545
0.8130914836797192
0.7604599047323508
0.8080789523513985
0.7646006914818333
0.8046006914818333
0.7675636544447964
0.802046413065486
0.769788348549357
0.8000913788523872
0.7715199502809587
0.7985469773079856
0.77290595166696
0.797296195569399
0.7740403816159106
0.7962626038381329
0.774986008093452
0.7953941713587581
PI = 3.1815766854350325

PI = 3.1815766854350325
```

```
In [26]: M = 25
         prod = 1.
         for n in range(1,M):
             n1 = 2.*n
             d1 = (2.*n-1)
             n2 = 2.*n
             d2 = (2.*n+1)
             prod = prod * (n1/d1) * (n2/d2)
             print (prod)

         print("PI = {}".format(prod*2.))

         # Amb Comprensió de llistes

         pi = 2.
         prod = [(2*n/(2.*n+1))*(2*n/(2.*n-1)) for n in range(1,M)]
         for n in prod:
             pi *= n
```

```
        print( "\nPI = {}".format(pi))
```

```
1.3333333333333333
1.4222222222222223
1.4628571428571429
1.4860770975056687
1.5010879772784531
1.51158509600068
1.519336814441709
1.5252949980277548
1.5300172735634443
1.5338519033217486
1.5370275801402202
1.5397006715839423
1.5419817096159185
1.5439510349155556
1.5456684442981092
1.5471793616434641
1.5485189108743243
1.5497146783730686
1.5507886317191344
1.5517584807696152
1.552638661416677
1.5534410586577192
1.5541755461559024
1.554850394417368
PI = 3.109700788834736

PI = 3.1097007888347363
```