Pràctica 2: Condicionals i iteracions

8 de desembre de 2020

1 Condicionals i iteracions

1.1 Comparacions

El **Python** utilitza els símbols <, <=, ==, >=, >, != per a fer comparacions entre objectes. El resultat és `True` si la condició és certa, mentre que és `False` si no ho és.

```
[1]: 2<3
```

```
[1]: True
```

```
[2]: 2>=3
```

```
[2]: False
```

```
[3]: 7==7
```

```
[3]: True
```

```
[4]: 'Hola'=='Hola'
```

```
[4]: True
```

```
[5]: 'Hola'>'Hola!'
```

```
[5]: False
```

```
[6]: 'Hola'<'Hola!'
```

```
[6]: True
```

```
[7]: 3!=4
```

```
[7]: True
```

Podem fer composicions de comparacions amb les intruccions `and`, `or` i `not`:

```
[8]: 2<3 and 3<2
```

[8]: False

```
[9]: not 2<3
```

[9]: False

```
[10]: 2<3 or 3>2
```

[10]: True

1.1.1 Exercici

Si entrem dues condicions separades per `!=`, què en resulta? Es pot veure el resultat a les dues execucions següents:

```
[11]: (2<3) != (2<4)
```

[11]: False

```
[12]: (2<3) != (3<2)
```

[12]: True

1.2 La instrucció `if`

La instrucció `if` permet executar una (o moltes) instrucció(ns) en cas que es compleixi una condició. Per saber quina és la darrera instrucció que s'ha de executar, el **Python** utilitza el tabulardor com a marca: tot el que està amb una tabulació més que la instrucció `if`, s'executa si la condició és compleix.
A més, admet l'estructura d'*altrament si* i *altrament* mitjançant `elif` i `else` respectivament.

1.2.1 Exemple:

Suposem que volem guardar a la variable `text` la paraula *Excel·lent*, *Notable*, *Aprovat* o *Suspès* en funció de la `nota`.

```
[13]: nota=7.3
```

```
[14]: if nota >= 9:
          text='Excel·lent'
      elif nota >=7:
          text='Notable'
      elif nota >=5:
          text='Aprovat'
      else:
          text='Suspès'
```

[15]:
```
print(text)
```

```
Notable
```

1.3 Iteracions amb la instrucció `for`

La sintaxi de la instrucció `for` necessita una llista (o conjunt) d'on agafar els elements. Una manera de generar aquesta llista és amb la funció `range`. Igual que amb la instrucció `if`, tot el que volguem agrupar s'ha de fer tabulant les línies que estan afectades:

[16]:
```
for i in range(9):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
```

[17]:
```
for i in range(0,10,2):
    print(i)
```

```
0
2
4
6
8
```

[18]:
```
# Calculem els primers 10 nombres de Fibonacci
a0,a1=1,1
print(a0,a1,sep='\n')
for i in range(9):
    a0,a1=a1,a0+a1
    print(a1)
```

```
1
1
2
3
5
8
13
21
34
```

```
55
89
```

També es pot utilizar la instrucció `for` i `if` per construir llistes, amb la sintaxis d'aquests exemples:

[19]:
```
quadrats=[x**2 for x in range(11)]
```

[20]:
```
print(quadrats)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[21]:
```
quadratsparells=[x**2 for x in range(11) if x%2==0]
```

[22]:
```
print(quadratsparells)
```

```
[0, 4, 16, 36, 64, 100]
```

Equivalentment:

[23]:
```
[x**2 for x in range(0,11,2)]
```

[23]:
```
[0, 4, 16, 36, 64, 100]
```

1.3.1 Exercici

Feu una llista amb tots els nombres senars entre 0 i 1000.

1.4 Iteracions amb la instrucció `while`

La instrucció `while` serveix per executar repetidament un conjunt d'instruccions mentre es compleixi una condició (en principi, no tenim perquè saber quantes iteracions es faran):

[24]:
```
i=1
while (i<1000):
    print(i)
    i=i*2
```

```
1
2
4
8
16
32
64
128
256
512
```

[25]:
```
print(i)
```

```
1024
```

1.4.1 Exercici

Feu una llista amb els nombres de Fibonacci més petits que 10000.

La instrucció `while` adment l'opció `else` per agrupar instruccions que s'executaran un cop ja no es compleixi la condició:

[26]:
```
i=1
while (i<1000):
    print(i)
    i=i*2
else:
    print(i,' és més gran que 1000')
```

```
1
2
4
8
16
32
64
128
256
512
1024  és més gran que 1000
```

També es pot interrompre el contingut d'un `for` o un `while` amb la instrucció `break` (surt de la iteració on es troba) o bé amb `continue` (passa a la iteració següent):

[27]:
```
for n in range (1,100):
    if n % 3 != 0:
        continue
    print(n,'és múltiple de 3')
```

```
3 és múltiple de 3
6 és múltiple de 3
9 és múltiple de 3
12 és múltiple de 3
15 és múltiple de 3
18 és múltiple de 3
21 és múltiple de 3
24 és múltiple de 3
27 és múltiple de 3
30 és múltiple de 3
33 és múltiple de 3
36 és múltiple de 3
39 és múltiple de 3
```

```
42 és múltiple de 3
45 és múltiple de 3
48 és múltiple de 3
51 és múltiple de 3
54 és múltiple de 3
57 és múltiple de 3
60 és múltiple de 3
63 és múltiple de 3
66 és múltiple de 3
69 és múltiple de 3
72 és múltiple de 3
75 és múltiple de 3
78 és múltiple de 3
81 és múltiple de 3
84 és múltiple de 3
87 és múltiple de 3
90 és múltiple de 3
93 és múltiple de 3
96 és múltiple de 3
99 és múltiple de 3
```

[28]:
```python
for n in range(2,100):
    m = 2
    while m < n:
        if n % m == 0:
            print(n, 'és igual a', m, '*', n//m)
            break
        m=m+1
    if m == n:
            print(n,'és un nombre primer')
```

```
2 és un nombre primer
3 és un nombre primer
4 és igual a 2 * 2
5 és un nombre primer
6 és igual a 2 * 3
7 és un nombre primer
8 és igual a 2 * 4
9 és igual a 3 * 3
10 és igual a 2 * 5
11 és un nombre primer
12 és igual a 2 * 6
13 és un nombre primer
14 és igual a 2 * 7
15 és igual a 3 * 5
16 és igual a 2 * 8
17 és un nombre primer
18 és igual a 2 * 9
```

```
19 és un nombre primer
20 és igual a 2 * 10
21 és igual a 3 * 7
22 és igual a 2 * 11
23 és un nombre primer
24 és igual a 2 * 12
25 és igual a 5 * 5
26 és igual a 2 * 13
27 és igual a 3 * 9
28 és igual a 2 * 14
29 és un nombre primer
30 és igual a 2 * 15
31 és un nombre primer
32 és igual a 2 * 16
33 és igual a 3 * 11
34 és igual a 2 * 17
35 és igual a 5 * 7
36 és igual a 2 * 18
37 és un nombre primer
38 és igual a 2 * 19
39 és igual a 3 * 13
40 és igual a 2 * 20
41 és un nombre primer
42 és igual a 2 * 21
43 és un nombre primer
44 és igual a 2 * 22
45 és igual a 3 * 15
46 és igual a 2 * 23
47 és un nombre primer
48 és igual a 2 * 24
49 és igual a 7 * 7
50 és igual a 2 * 25
51 és igual a 3 * 17
52 és igual a 2 * 26
53 és un nombre primer
54 és igual a 2 * 27
55 és igual a 5 * 11
56 és igual a 2 * 28
57 és igual a 3 * 19
58 és igual a 2 * 29
59 és un nombre primer
60 és igual a 2 * 30
61 és un nombre primer
62 és igual a 2 * 31
63 és igual a 3 * 21
64 és igual a 2 * 32
65 és igual a 5 * 13
66 és igual a 2 * 33
```

```
67 és un nombre primer
68 és igual a 2 * 34
69 és igual a 3 * 23
70 és igual a 2 * 35
71 és un nombre primer
72 és igual a 2 * 36
73 és un nombre primer
74 és igual a 2 * 37
75 és igual a 3 * 25
76 és igual a 2 * 38
77 és igual a 7 * 11
78 és igual a 2 * 39
79 és un nombre primer
80 és igual a 2 * 40
81 és igual a 3 * 27
82 és igual a 2 * 41
83 és un nombre primer
84 és igual a 2 * 42
85 és igual a 5 * 17
86 és igual a 2 * 43
87 és igual a 3 * 29
88 és igual a 2 * 44
89 és un nombre primer
90 és igual a 2 * 45
91 és igual a 7 * 13
92 és igual a 2 * 46
93 és igual a 3 * 31
94 és igual a 2 * 47
95 és igual a 5 * 19
96 és igual a 2 * 48
97 és un nombre primer
98 és igual a 2 * 49
99 és igual a 3 * 33
```

1.4.2 Exercici

Feu una llista amb tots els nombres primers menors que 10000.