lez_02

January 21, 2026

1 Cicli

Come nella stragrande maggioranza dei linguaggi di programmazione, in python troviamo due tipi di loop: i cicli **for** e i cicli **while**:

- Il ciclo **while** viene ripetuto finché una condizione non smette di essere vera;
- Il ciclo **for** viene eseguito un certo numero di volte, ma viene utilizzato principalmente per 'scorrere' gli elementi di una lista (vedremo poi)

1.1 Il ciclo while

La sintassi del ciclo while è perlopiù come ce la si aspetta:

```
while condizione:     codice
```

Finché la condizione viene valutata come **True**, il blocco di codice nel corpo del while verrà rieseguito. Quindi è necessario che il codice modifichi in qualche modo la variabile alla quale la condizione fa riferimento, o il ciclo non terminerà mai. Ovviamente esistono delle eccezioni, in cui il ciclo while viene interrotto all'interno del suo blocco di codice con l'istruzione **break**.

[3]:
```
#Stampo i numeri da 0 a 10

i = 0 #inizializzo il contatore

while i < 11:
    print(i)
    i += 1 #incremento il contatore
```

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

[1]:
```
#Stessa cosa, ma interrompendo il ciclo con break

i = 0

while True:
    print(i)
    i += 1
    if i > 10:
        break
```

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

Ovviamente in un caso così semplice è decisamente preferibile definire una condizione chiara subito dopo while, senza ricorrere all'istruzione break. Tuttavia possono presentarsi casi in cui il suo utilizzo è necessario.

1.2 Il ciclo for

Il ciclo for viene spesso usato in coppia con la funzione **range()**, quindi prima di descrivere il ciclo, vediamo cosa fa la funzione:

[4]:
```
intervallo = range(0, 10)
print(intervallo, type(intervallo))
```

```
range(0, 10) <class 'range'>
```

Questa funzione ritorna un oggetto della classe *range*, e per ora basti sapere che questo oggetto è un **iterabile** che contiene i numeri da 0 a 9. L'intervallo inferiore è compreso, quello superiore no.

Il ciclo for in python serve per scorrere un iterabile. Gli iterabili comprendono le liste (vettori), le tuple, i dizionari e altri tipi di variabile che vedremo più avanti.

Quindi, dato che la nostra funzione *range()* ritorna un iterabile, possiamo "*scorrerla*" con un ciclo for:

[5]:
```
for pippo in range(0, 10):
    print(pippo)
```

```
0
1
2
```

```
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

[ ]:

La sintassi generale del *for* è quella rappresentata nel codice sopra, ossia:

`for indice in iterabile:     codice(valore dell'indice)`

La funzione range funziona anche con un solo parametro: in quel caso il parametro indicherà **l'estremo superiore** dell'intervallo:

[7]:
```
for i in range(5):
    print('ciao')
```

```
ciao
ciao
ciao
ciao
ciao
```

E in aggiunta, può anche prendere 3 parametri, tali che: - Il primo parametro sarà l'estremo inferiore; - Il secondo parametro sarà l'estremo superiore; - Il terzo parametro sarà il passo, o l'incremento, del range

Cerchiamo di chiarire questo aspetto con un esempio:

[20]:
```
#for i in range(0, 11):
#    print(i)
```

```
seq = 'ATCGATCGATCGCTAGCTG'
for i in range(len(seq)):
    print(seq[i])
```

```
A
T
C
G
A
T
C
G
A
T
C
G
C
T
A
G
C
T
G
```

Quindi ad ogni passaggio del ciclo, il contatore viene incrementato del valore del passo. È anche possibile definire un passo negativo, procedendo quindi al contrario, purché estremo superiore ed inferiore siano invertiti:

[21]:
```
for i in range(10, -1, -2):
    print(i)
```

```
10
8
6
4
2
0
```

Per chiudere, dimentichiamoci per un attimo di non aver parlato di liste, definiamone una, e utilizziamo il ciclo for per scorrere i suoi elementi:

[23]:
```
lista_animali = ['Cane', 'Gatto', 'Toporagno']

for bestia in lista_animali:
    print(len(bestia))
    print(bestia)
```

```
4
Cane
5
```

```
Gatto
9
Toporagno
```

Ora passiamo ad un semplice esercizio che ci porterà ad utilizzare i loop e introdurrà il nostro primo **import**. La keyword **import** è utilizzata per caricare del codice python presente in un altro file.

Generalmente questo significa importare una libreria o un modulo *famosi*, come pandas o numpy o scipy, ma viene anche utilizzata per importare file scritti da noi.

Nel seguente esempio lanceremo due dadi, forniremo all'utente il risultato ottenuto sommando i loro punteggi, chiederemo all'utente se vuole giocare ancora e in caso affermativo ripeteremo l'operazione.

Per calcolare il punteggio del singolo dado utilizzeremo la funzione **randint()** contenuta nel modulo **random**:

[25]:
```
import random

continuare = 'si'

while continuare == 'si':
    print('Lancio il primo dado....')
    d1 = random.randint(1, 6)
    print('Lancio il secondo dado...')
    d2 = random.randint(1, 6)
    print('Il risultato è {} + {} = {}'.format(d1, d2, d1 + d2))
    continuare = input('Vuoi giocare ancora? [si/no]')
print('Alla prossima!')
```

```
Lancio il primo dado…
Lancio il secondo dado…
Il risultato è 3 + 2 = 5

Vuoi giocare ancora? [si/no] si

Lancio il primo dado…
Lancio il secondo dado…
Il risultato è 6 + 5 = 11

Vuoi giocare ancora? [si/no] pippo

Alla prossima!
```

1.3 Esercizi

1. Lanciare due dadi finché entrambi non danno 6 contemporaneamente, stampare il numero di tentativi impiegati;
2. Stampare tutti i numeri primi compresi tra 0 e un numero intero n;
3. Calcolare il fattoriale di un numero prima con il ciclo while e poi con il ciclo for;
4. Lanciare un dado 1000 volte e contare quante volte esce ciascun valore;

5. Utilizzando un ciclo *for*, produrre e stampare la sequenza complementare di 'ATCGCGTAGCTGATG'

1.4 Soluzioni

..

..

..

..

..

..

..

..

..

..

..

..

1.5 Esercizio 1

[31]:
```
from random import randint

tentativi = 0

while True:
    d1 = randint(1, 6)
    d2 = randint(1, 6)
    if d1 == 6 and d2 == 6:
        print('tentativi necessari: ', tentativi)
        break
    tentativi += 1
```

```
tentativi necessari:  22
```

1.6 Esercizio 2

[34]:
```
n = 100

for i in range(n, 1, -1):
    primo = True
    for j in range(2, i):
        if i % j == 0:
            primo = False
```

```
    if primo:
        print(i)
```

```
97
89
83
79
73
71
67
61
59
53
47
43
41
37
31
29
23
19
17
13
11
7
5
3
2
```

1.7 Esercizio 3 - While

[36]:
```
n = 9
fattoriale = 1
while n > 1:
    fattoriale *= n
    n -= 1
print(fattoriale)
```

```
362880
```

1.8 Esercizio 3 - For

[38]:
```
n = 9

for i in range(1, n):
    n *= i
print(n)
```

```
362880
```

1.9 Esercizio 4

[1]:
```
from random import randint

x1 = 0
x2 = 0
x3 = 0
x4 = 0
x5 = 0
x6 = 0

for i in range(1000):
    x = randint(1, 6)
    if x == 1:
        x1 += 1
    elif x == 2:
        x2 += 1
    elif x == 3:
        x3 += 1
    elif x == 4:
        x4 += 1
    elif x == 5:
        x5 += 1
    else:
        x6 += 1

print(x1, x2, x3, x4, x5, x6)
```

```
156 171 167 186 167 153
```

1.10 Esercizio 5

[1]:
```
seq = 'ATCGCGTAGCTGATG'
comp = ''
for nucleotide in seq:
    if nucleotide == 'A':
        comp += 'T'
    elif nucleotide == 'T':
        comp += 'A'
    elif nucleotide == 'G':
        comp += 'C'
    else:
        comp += 'G'
print(seq)
print(comp)
```

```
ATCGCGTAGCTGATG
TAGCGCATCGACTAC
```