lez_01

January 21, 2026

1 Variabili numeriche ed espressioni condizionali

Nella lezione precedente abbiamo parlato di variabili ed abbiamo approfondito le stringhe.

In questa lezione vedremo le variabili numeriche e poi passeremo alle variabili booleane per introdurre le espressioni condizionali. Contestualmente vedremo come definire i blocchi di codice in python.

1.1 *Variabili numeriche*

1.1.1 Numeri interi e float

In python troviamo due tipi di variabili numeriche: - Numeri interi (class Int) - Numeri a virgola mobile (class Float)

I due tipi di variabile si definiscono banalmente come segue:

[2]:
```
x = 1
y = 2
a = 1.3
b = 12.5

print(type(x), type(a))
```

```
<class 'int'> <class 'float'>
```

Le operazioni elementari sono definite con i soliti operatori:

[3]:
```
print('somma: ', (x + y))
print('differenza: ', (x - y))
print('prodotto: ', (x * y))
print('divisione: ', (b / a))
print('divisione senza resto: ', (b // a))
print('elevamento a potenza: ', (y ** 2))
print('modulo (resto divisione)', (y % 1.3))
```

```
somma:  3
differenza:  -1
prodotto:  2
divisione:  9.615384615384615
divisione senza resto:  9.0
```

```
elevamento a potenza:  4
modulo (resto divisione) 0.7
```

1.1.2 Conversione da stringhe a numeri

È possibile convertire una stringa in una variabile numerica, a patto che questa contenga dei valori numerici, con le funzioni *int()* e *float()*

[4]:
```
stringa = '43'
print(stringa, type(stringa))
numero = int(stringa)
print(numero, type(numero))
print(17 + int(stringa))
```

```
43 <class 'str'>
43 <class 'int'>
60
```

[5]:
```
stringa = '43.3'
print(float(stringa))
```

```
43.3
```

Attenzione: affinché una stringa venga convertita in un intero, è necessario che la stringa rappresenti INEQUIVOCABILMENTE un intero. Tutte le seguenti conversioni, infatti danno errore (de-commentare per credere):

[20]:
```
int('17.5')
int('17b')
int('a17')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[20], line 1
----> 1 int('17.5')
      2 int('17b')
      3 int('a17')

ValueError: invalid literal for int() with base 10: '17.5'
```

1.2 Variabili Booleane e variabile None

1.2.1 Variabili Booleane

Una variabile Booleana può assumere solo due valori: **True** o **False**. Sono generalmente il prodotto di un confronto tra due variabili. Gli operatori di confronto sono gli stessi che si trovano in matematica e negli altri linguaggi di programmazione:

- Uguale: $==$
- Maggiore: $>$

- Minore: $<$
- Minore o uguale / Maggiore o uguale: $<=$ / $>=$

[18]:
```
a = 1
b = 2
c = 1

print(b > a)
print(a > b)
print(a == c)
```

```
True
False
True
```

Le operazioni di confronto possono essere combinate utilizzando gli operatori logici **and**, **or**, e **not**

[19]:
```
print((a > b) or (c == a)) #True se almeno una delle due condizioni è␣
 ↪soddisfatta
print((a > b) and (c == a)) #True solo se entrambe le condizioni sono␣
 ↪soddisfatte
print(not a > b) # True solo se la condizione NON è soddisfatta
```

```
True
False
True
```

E osserviamo anche come sia possibile **sommare** le variabili Booleane. I **True** verranno infatti interpretati come **1**, i **False** come **0**

[16]:
```
print(int(True), int(False))
print(True + True)
print(True + True + False + True)
```

```
1 0
2
3
```

Per finire, introduciamo la variabile **None**, ossia il valore nullo in python.

Questa non viene quasi mai assegnata esplicitamente ad una variabile, tuttavia può essere ritornata da alcune funzioni in particolari situazioni.

Ad esempio, supponiamo di voler scrivere una funzione che cerchi un k-mero in una sequenza nucleotidica: nel caso in cui non lo trovasse potremmo farci ritornare **None** dalla funzione.

[15]:
```
a = None
print(a)
```

```
None
```

1.3 Espressioni condizionali

Con le espressioni condizionali introduciamo il concetto di **flow control**, ossia la gestione dell'ordine, della modalità, e delle volte nelle quali le operazioni in un programma vengono eseguite.

In particolare, con le espressioni condizionali siamo chiamati a scegliere se un'operazione sarà eseguita o meno dal programma data una particolare condizione, e se esistono una o più alternative nel caso in cui la condizione non sia soddisfatta.

Contestualmente, vedremo anche la sintassi dei blocchi di codice in python.

Le **keywords** per l'implementazione delle espressioni condizionali sono 3: ***if***, ***elif*** ed ***else***

[14]:
```
a = 4

if a % 2 == 0:
    print('a è un numero pari')
```

```
a è un numero pari
```

La sintassi di base dell'if è identica a quella della maggior parte dei linguaggi di programmazione:
`> if <condizione>: >  codice`

Se la condizione è valutata come **True**, ossia **se è vera**, il codice viene eseguito, altrimenti viene saltato.

Affinché una condizione sia valida, questa deve essere rappresentata da un espressione che viene valutata come True o come False.

Notare come in python non si utilizzino parentesi per delimitare i blocchi di codice come ad esempio in C:

```
if(condizione){  Codice}
```

Ma si utilizzino invece i duepunti **':'** seguiti **dall'indentazione**. Un blocco di codice finisce quando si ritorna all'indentazione precedente:

[13]:
```
a = 2
b = 4

if a > 2:
    print('Questa frase è nel blocco if')
    print('Anche questa')
print('Questa frase NO!')
```

```
Questa frase NO!
```

[12]:
```
if a < b:
    print('Questa frase è nel blocco if')
    print('Anche questa')
print('Questa frase NO!')
```

```
Questa frase è nel blocco if
Anche questa
Questa frase NO!
```

Come negli altri linguaggi di programmazione è possibile combinare più condizioni con gli operatori logici:

[11]:
```
a = 1
b = 2
c = 1

if (a > b) or (a == c):
    print('Ok1')

if (a < b) and (a ==c):
    print('Ok2')

if not a > b:
    print('Ok3')
```

```
Ok1
Ok2
Ok3
```

Se invece vogliamo valutare più condizioni separatamente, possiamo ricorrere alle altre due keywords delle espressioni condizionali: **elif** ed **else**

- **elif** va usato dopo un'**if**, e anche lui ha bisogno di una condizione da valutare;
- **else** viene usato alla fine di una serie più o meno lunga di **if** ed **elif** e non richiede condizioni

[10]:
```
a = 2
b = 1

if a > b:
    print('A è maggiore di B')
elif a == b:
    print('A è uguale a B')
else:
    print('A è minore di B')
```

```
A è maggiore di B
```

Per finire, possiamo scrivere un'espressione condizionale su una sola riga, utilizzando la sintassi nel codice qui sotto. Questi metodi di scrivere operazioni "complesse" in una sola riga è definito un *one-liner* ed è molto comune nel linguaggio python:

[9]:
```
a = 1
b = 2

print('A è maggiore') if a > b else print('A è minore o uguale')
```

```
A è minore o uguale
```

Attenzione: se utilizziamo questo costrutto omettendo l'else, questo non funzionerà e darà errore (*SyntaxError*).

[8]:
```python
a = 1
b = 2

if a:
    print('a')
if a:
    print('b')

print('B è maggiore di A') if b > a
```

```
  Cell In[8], line 9
    print('B è maggiore di A') if b > a
    ^
SyntaxError: expected 'else' after 'if' expression
```

In questo caso specifico possiamo scrivere l'espressione condizionale su una sola riga mettendo l'if prima dell'istruzione da eseguire, nel modo seguente:

[7]:
```python
a = 1
b = 2

if b > a: print('B è maggiore di A')
```

```
B è maggiore di A
```

> **Nota personale**: non sono un grande fan delle espressioni su una sola riga, perché **a mio parere** rendono il codice meno leggibile. Tuttavia per operazioni molto semplici come quella scritta sopra è considerato dalla maggior parte degli utenti un modo compatto ed elegante di scrivere un'espressione condizionale. A voi la scelta!

1.4 Esercizi

1. Prendere in input dall'utente un numero intero e stabilire se questo sia pari o dispari
2. Prendere in input numeratore e denominatore di una frazione e stabilire se questa sia propria, impropria o apparente
3. Prendere due numeri in input dall'utente, eseguirne la media e prenderne la parte intera. Stabilire se questa sia pari o dispari

1) prendere i due numeri in input 2) calcolare la media 3) prendere solo la parte intera della media 4) valutare se questa sia pari o dispari 5) comunicarlo all'utente! .. ..

1.5 Soluzioni

1.5.1 Esericizio 1

[ ]:
```
n = int(input('Dammi un numero intero: '))
print('Il numero è pari') if n % 2 == 0 else print('Il numero è dispari')
```

1.5.2 Esercizio 2

[ ]:
```
num = int(input('Numeratore: '))
den = int(input('Denominatore: '))

if num % den == 0:
    print('La frazione è apparente')
elif num < den:
    print('La frazione è propria')
else:
    print('La frazione è impropria')
```

1.5.3 Esercizio 3

[ ]:
```
m = int(input('Dammi un numero: '))
n = int(input('Dammi un altro numero: '))
media = int((m + n) / 2)
if media % 2 == 0:
    print('La media è pari')
else:
    print('La media è dispari')
```