Introduzione al linguaggio Python

Laboratorio di Matematica Computazionale

Massimo Pasquetto

I.T.S. "Cangrande della Scala" – Verona

3 novembre 2019

Notizie introduttive di Python

1 Tipi di dato e variabili

tratto da https://hpc-forge.cineca.it/files/CoursesDev/public/2016/Milan/Introduction

Python è un linguaggio dinamico e fortemente tipato.

Esempi di tipi di dato

```
>>> a = '123'           # tipo string
>>> b = 127             # tipo int
>>> check = True        # tipo bool
>>> piGreco = 3.14159   # tipo float
>>> c = 1000 + int(a)   # casting ad un intero
>>> print(c)
1123
>>> d = b + piGreco     # casting automatico ad un float
>>> print(d)
130.14159
>>> e = b + a
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    e = b + a
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

È anche possibile visualizzare il tipo di dato

```
>>> type(a)
<class 'str'>
>>> type(b)
<class 'int'>
>>> type(check)
<class 'bool'>
```

1.1 Variabili e assegnamento

Come in algebra anche in Python utilizziamo le variabili per rappresentare i numeri, ma anche le lettere e altri tipi di dato. Le variabili vengono utilizzate per memorizzare un "valore"

```
>>> a = 23
>>> b = a + 5
>>> print(a)
23
>>> print(b)
28
```

Nella riga 1 del precedente codice compare una delle istruzioni più semplici dei linguaggi di programmazione ma anche una delle più importanti. Il simbolo = non indica l'uguaglianza come in matematica ma piuttosto va letto come

"assegna alla variabile a sinistra del simbolo = il risultato dell'espressione a destra dello stesso simbolo"

Ogni volta che il nome di una variabile compare a destra del simbolo di assegnamento = si intende il contenuto della variabile, pertanto nella riga 2 del codice precedente alla variabile `b` viene assegnato il risultato dell'espressione "contenuto della variabile `a` sommato al numero intero 5"

Anche i successivi comandi nelle righe 3 e 5 fanno riferimento al contenuto della variabile.

Il tipo di una variabile è determinato dall'assegnamento e può cambiare

1.2 Identificatore

In matematica i nomi delle variabili sono di solito composti da una lettera e spesso si utilizzano le lettere x, y, z, mentre per i parametri si preferiscono le lettere k, t talvolta a, b, c, m, q.

Gli informatici preferiscono chiamare le variabili con nomi più esplicativi che in qualche modo facciano capire a cosa serve quella variabile, per esempio `altezza`, `totale` e `sconto` sono da preferire a `h`, `t` e `s`.

I nomi delle variabili devono essere degli identificatori validi. Gli identificatori sono i nomi utilizzabili nel linguaggio di programmazione. Ci sono delle regole molto rigide per la scelta degli identificatori.

Gli identificatori sono i nomi dati per identificare qualcosa e si ottengono seguendo queste regole:

1. Un identificatore deve contenere almeno un carattere.
2. Il primo carattere di un identificatore deve essere una lettera dell'alfabeto (maiuscola o minuscola) o un carattere "_" (underscore)

   `ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_`

3. I caratteri successivi, se esistono, possono essere lettere dell'alfabeto maiuscole o minuscole, caratteri di sottolineatura (carattere underscore) o cifre

   `ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_0123456789`

4. Nessun altro carattere, incluso lo spazio, è permesso.
5. I nomi degli identificatori sono case-sensitive (cioè il sistema distingue tra lettere minuscole e maiuscole)
6. Le parole riservate del linguaggio non possono essere usate come identificatori (vedi fig. 1).

Alcuni esempi di identificatori validi sono: `i`, `__mio_nome`, `totale_1` e `a1b2_c3`

and	del	from	None	try
as	elif	global	nonlocal	True
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

Figura 1 – Parole riservate di Python

Non sono identificatori validi i seguenti nomi:

- **2cose**, inizia con una cifra;
- **mio nome**, contiene uno spazio;
- **mio-nome**, **mio "nome"**, contengono caratteri non mermessi;
- **and**, è una parola riservata del linguaggio Python

In Python non esistono dichiarazioni esplicite di variabili. Vengono create nel momento dell'assegnamento e distrutte all'uscita dell'ambito in cui vengono utilizzate ("scope").

Per conoscere il tipo di valori che si possono assegnare ad una variabile e soprattutto quali operazioni si possono utilizzare si utilizza la funzione predefinita (built-in) `type()`.

```
>>> a = 300
>>> type(a)
<class 'int'>
>>> scuola = 'Cangrande della Scala'
>>> type(scuola)
<class 'str'>
```

1.3 Dati semplici e contenitori

Python dispone di due tipologie di dato, dati semplici

- `int`
- `bool`
- `float`
- `str`
- `complex`

e contenitori

- `tuple ( )`
- `list [ ]`
- `dict { }`

1.4 Il tipo int

È l'equivalente dell'insieme dei numeri interi. Non ha ne massimo ne minimo. Il massimo valore rappresentabile dipende dalla potenza del computer.

Programma 1 – Il tipo int

```
>>> a = 300
>>> b = 2 ** 1024
>>> print("Il prodotto di a e b vale: ", a * b)
Il prodotto di a e b vale: 539307940458694772318791557236707420085393093682691
971820290243473198027416502889398125431967222608063360341639614180072976369306
443249867478542291918422373133303680274596455828906658803738282358359248856255
017306514452047027388644421739331622481711490051532053758894719841737815439148
914506068988872672411648 00
```

La divisione intera tra due interi si indica con un doppio slash `15 // 6` e il risultato è 2. Il resto della divisione intera si indica con il simbolo di percentuale.

Se a diviso b è uguale a q con il resto di r, significa che

$$a = q \cdot b + r$$

```
>>> q = 15 // 6    # divisione intera
>>> r = 15 % 6     # resto della divisione
>>> print('Il quoziente intero di 15 diviso 6 è: ', q)
Il quoziente intero di 15 diviso 6 è: 2
>>> print('Il resto della divione intera di 15 diviso 6 è: ', r)
Il resto della divione intera di 15 diviso 6 è: 3
```

1.5 Il tipo bool

Una proposizione può assumere solo due valori di verità: "Vero" o "Falso".

p	not(p)
False	True
True	False

p	q	p or q	p and q
False	False	False	False
False	True	True	False
True	False	True	False
True	True	True	True

Tabella 1 – Tavole di verità

Il tipo `bool` ha solo due valori possibili, le costanti `True` e `False`. Le principali operazioni possibili sono `not`, `or`, `and`.

Per esempio

```
>>> p = -12 < -5
>>> print(p)
True
>>> q = -7 == 7
>>> print(q)
False
>>> r = not(q)
>>> print(r)
True
>>> type(r)
<class 'bool'>
```

Esercizio 1 Calcolare tutti i possibili valori di verità delle proposizioni `not(p)`, `p and q`, `p or q` al variare di `p` e `q`.

Il tipo `bool` viene utilizzato ogni volta una espressione ha come risultato un valore di verità vero o falso.

Per controllare se un certo valore indicato dall'identificatore `x` è compreso tra i numeri interi -5 e $+3$ possiamo scrivere l'espressione booleana

```
>>> x = -2
>>> p = (x > -5) and (x < 3)
>>> print(p)
True
>>> x = 5
>>> p = (x > -5) and (x < 3)
>>> print(p)
False
```

Altrimenti per controllare se `x` è un numero negativo oppure maggiore o uguale di 7 utilizziamo la seguente espressione

```
>>> x = -9
>>> q = (x < 0) or (x >= 7)
>>> print(q)
True
```

Esercizio 2 Calcolare i valori di verità della proposizione `(x**2 - 16) > 0` utilizzando 10 valori di `x` diversi, come nell'esempio che segue.

```
>>> x = -3
>>> p = (x**2 - 16) > 0
>>> print(p)
False
```

1.6 Il tipo float

Il tipo `float` rappresenta i numeri reali. Per rappresentare i numeri decimali si usa il punto al posto della virgola. È possibile inserire i numeri anche utilizzando la notazione esponenziale.

```
>>> a = 12.347
>>> b = 3.7e-2
>>> c = 5 / 7
>>> print(c)
0.7142857142857143
>>> type(b)
<class 'float'>
```

Quando si utilizzano i numeri di tipo float bisogna sempre considerare i problemi di approssimazione dovuti alla rappresentazione dei numeri reali in virgola mobile

```
>>> a = 7 / 5
>>> print(a)
1.4
>>> print("{:.30f}".format(a))
1.399999999999999911182158029987
>>> x = 1 / 10
>>> y = 31 / 10 - 3
>>> print("x={0:.5f} y={1:.5f}".format(x, y))
x=0.10000 y=0.10000
>>> print("x={0:.15f} y={1:.15f}".format(x, y))
x=0.100000000000000 y=0.100000000000000
>>> print("x={0:.25f} y={1:.25f}".format(x, y))
x=0.1000000000000000055511151 y=0.1000000000000000888178420
```

Questi errori di approssimazione non devono stupire abbiamo lo stesso problema ogni volta che facciamo una divisione e otteniamo per risultato un numero periodico e quindi siamo costretti ad approssimarlo scrivendo un numero finito di cifre.

Utilizzare la console di Python per calcolare l'espressione

$$\left(1-\frac{1}{2}\right)^2+\left(1+\frac{1}{2}\right)^2-3\left[\left(1-\frac{1}{2}\right)\left(1+\frac{1}{2}\right)\right]^{-1}$$

il primo metodo consiste nell'inserire l'intera espressione esplicitando gli operatori mancanti e ricordando che si possono utilizzare solo le parentesi tonde

```
>>> (1 - 1 / 2) ** 2 + (1 + 1 / 2) ** 2 - 3 * ((1 - 1 / 2) * (1 + 1 / 2)) ** (-1)
-1.5
```

un secondo metodo può consistere nel ricercare nell'espressione degli schemi ricorrenti

```
>>> a = 1 - 1 / 2
>>> b = 1 + 1 / 2
>>> a ** 2 + b ** 2 - 3 * (a * b) ** (-1)
-1.5
```

Talvolta si utilizza la notazione scientifica per rappresentare i numeri decimali, soprattutto quei numeri molto grandi o molto piccoli.

```
>>> massaElettrone = 9.109e-31
>>> massaTerra = 5.972e24
>>> c = 2.998e8   # in fisica c indica la velocità della luce misurata in m/s
```

Arrotondamento

```
>>> x = 1693.57836
>>> round(x)      # il risultato è di tipo <int>
1694
>>> round(x, 0)   # il risultato è di tipo <float>
1694.0
>>> round(x, 1)
1693.6
>>> round(x, 2)
1693.58
>>> round(x, -1)
1690.0
>>> round(x, -2)
1700.0
```

1.7 String

In informatica si indica con il termine "stringa" una qualsiasi sequenza di caratteri alfanumerici oppure caratteri speciali anche non stampapili.

```
>>> str1 = "0123456789abcdefghijkABCD £$%&/()=?^;,:._-+*@#ˇ"
>>> str2 = 'Marco Pantani "il pirata"'
>>> a = 3.14
>>> type(a)
<class 'float'>
>>> str3 = str(a)
>>> print(str3)
3.14
>>> type(str3)
<class 'str'>
>>> len(str2)
25
```

La funzione predefinita `len()` restituisce la lunghezza di una stringa cioè il numero di caratteri che la compongono.

Sulle stringhe si possono compiere alcune operazioni come per esempio la concatenazione di stringhe. L'operatore che indica la concatenazione di due stringhe si indica con il simbolo +. Una stringa può anche essere moltiplicata per un intero positivo, in questo caso si ottiene una concatenazione ripetuta. Moltiplicando una stringa per un numero intero non positivo si ottiene la stringa nulla.

```
>>> s1 = 'My favourite sport is '
>>> s2 = 'tennis'
>>> s3 = s1 + s2
>>> print(s3)
My favourite sport is tennis
>>> s4 = 'Ciao' * 7
>>> print(s4)
'CiaoCiaoCiaoCiaoCiaoCiaoCiao'
>>> 'Ciao' * (-1)
''
```

Una stringa è composta da una successione ordinata di singoli caratteri ognuno dei quali occupa una precisa posizione; il primo carattere della stringa è in posizione 0, il secondo in posizione 1 e così via.

Possiamo ottenere una sottostringa utilizzando il concetto di intervallo indicato con il simbolo :.

```
>>> s1 = 'Istituto Tecnico Statale "Cangrande della Scala" - Verona'
>>> s1[0]
'I'
>>> s1[5]
'u'
>>> s1[9:16]  # sottostringa dalla posizione 9 e formata da (16-9) caratteri
'Tecnico'
```

Esercizio 3 Qual è l'output della seguente sequenza di istruzioni?

```
>>> s1 = 'Istituto Tecnico Statale "Cangrande della Scala" - Verona'
>>> s2 = 'the best school'
>>> s3 = s1[25:48] + ' is ' + s2 + ' in ' + s1[51:57]
>>> print(s3)
```

1.8 Complex

Python permette di operare anche con l'insieme dei numeri complessi. L'unità immaginaria viene indicata con la lettera `j` e le operazioni algebriche definite nell'insieme dei numeri complessi sono l'addizione, la sottrazione, la moltiplicazione, la divisione e l'elevamento a potenza.

```
>>> a = -1 + 2j
>>> type(a)
<class 'complex'>
>>> b = 1 + 1j
>>> print(a + b)
3j
>>> print(a - b)
(-2+1j)
>>> print(a * b)
(-3+1j)
>>> print(a / b)
(0.5+1.5j)
>>> print(a**2)
(-3-4j)
```

1.9 Operazioni su dati numerici

Consideriamo le operazioni eseguibili sui tipi di dato `<int>` e `<float>`. Il tipo di dato del risultato dipende dal tipo di dato degli operatori.

In particolare:

- addizione(+), sottrazione(-), moltiplicazione(*)
  - **se** entrambi gli operandi sono di tipo `<int>` **allora** anche il risultato è `<int>`
  - **se** almeno uno degli operandi è di tipo `<float>` **allora** anche il risultato è `<float>`
- divisione tra interi(//)
  - **se** entrambi gli operandi sono di tipo `<int>` **allora** anche il risultato è `<int>`
- divisione(/)
  - il risultato è sempre di tipo `<float>`
- resto della divisione tra interi(%)
  - **se** entrambi gli operandi sono di tipo `<int>` **allora** anche il risultato è `<int>`
- elevamento a potenza(**)
  - **se** la base è di tipo `<int>` e l'esponente è un `<int>` non negativo **allora** il risultato è `<int>`
  - **se** la base è di tipo `<int>` e l'esponente è un `<int>` negativo **allora** il risultato è `<float>`
  - **se** la base o l'esponente sono di tipo `<float>` **allora** il risultato è `<float>`

Operatore	Funzione
+	Addizione
-	Sottrazione
*	Moltiplicazione
//	Divisione intera
%	Resto della divisione intera
/	Divisione

Tabella 2 – Operazioni numeriche

Le precedenti operazioni si possono applicare agli insiemi numerici <int>, <float>, <complex>.

```
>>> -5 - (-7)          # il risultato è di tipo <int>
2
>>> 8 + (-3.5)         # il risultato è di tipo <float>
4.5
>>> 13 // 3            # il risultato è di tipo <int>
4
>>> 13 / 3             # il risultato è di tipo <float>
4.333333333333333
>>> 12 // 3            # il risultato è di tipo <int>
4
>>> 12 / 3             # ATTENZIONE! il risultato è di tipo <floaT>
4.0
>>> 13 % 3             # il risultato è di tipo <int>
1
>>> (-2)**(3)          # il risultato è di tipo <int>
-8
>>> (2)**(-3)          # il risultato è di tipo <float>
0.125
>>> (2.0)**(3)         # il risultato è di tipo <float>
8.0
```

1.10 Operatori logici di confronto

Gli operatori di confronto restituiscono un valore di tipo <bool>.

Operatore	Funzione
>	Maggiore
>=	Maggiore o uguale
<	Minore
<=	Minore o uguale
==	Uguaglianza
!=	Diverso

Tabella 3 – Operazioni di confronto

2 Input Output

Scrivere un programma Python per determinare la media aritmetica di 5 numeri letti da tastiera.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
  Autore: Massimo Pasquetto
  Data: 29 Ottobre 2019
  Classe: 1B
"""

print("-"*60)
print("Programma per calcolare la MEDIA ARITMETICA di 5 numeri")
print("-"*60)

print("Inserire il valore n. 1: ", end="")
x1 = float(input())
print("Inserire il valore n. 2: ", end="")
x2 = float(input())
print("Inserire il valore n. 3: ", end="")
x3 = float(input())
print("Inserire il valore n. 4: ", end="")
x4 = float(input())
print("Inserire il valore n. 5: ", end="")
x5 = float(input())

somma = x1 + x2 + x3 + x4 + x5
media = somma / 5

print("La MEDIA ARITMETICA dei cinque numeri è:", media)
```

Modificare il programma in modo tale che si possa calcolare la media aritmetica di 10 numeri, cercando di limitare il numero delle variabili utilizzate.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
  Autore: Massimo Pasquetto
  Data: 29 Ottobre 2019
  Classe: 1B
"""

print("-"*60)
print("Programma per calcolare la MEDIA ARITMETICA di 10 numeri")
print("-"*60)

somma = 0

print("Inserire il valore n. 1: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 2: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 3: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 4: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 5: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 6: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 7: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 8: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 9: ", end="")
x = float(input())
somma = somma + x
print("Inserire il valore n. 10: ", end="")
x = float(input())
somma = somma + x

media = somma / 10

print("La MEDIA ARITMETICA dei dieci numeri è:", media)
```

Si osservi che le istruzioni si ripetono secondo uno schema ben preciso. È possibile inserirle in un blocco iterativo che ripeta le istruzioni per 10 volte.

Le istruzioni che seguono il ciclo `for i in range(1,11,1)` e allineate con 4 spazi a destra(identazione) rispetto alla parola riservata `for`, vengono eseguite al variare di `i` dal numero 1 al numero 10.

L'iterativo `range(1,11,1)` assume in ordine tutti i numeri interi da 1 a 11-1 con incrementi di 1.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
  Autore: Massimo Pasquetto
  Data: 29 Ottobre 2019
  Classe: 1B
"""

print("-"*60)
print("Programma per calcolare la MEDIA ARITMETICA di 10 numeri")
print("-"*60)

somma = 0

for i in range(1, 11, 1):
    print("Inserire il valore n.", i,": ", end="")
    x = float(input());
    somma = somma + x

media = somma / 10

print("La MEDIA ARITMETICA dei dieci numeri è:", media)
```

Modificare il programma in modo tale che sia possibile leggere da input il numero di valori di cui si vuole calcolare la media aritmetica.

Programma 2 – Calcolo della media aritmetica (vers. 4)

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
  Autore: Massimo Pasquetto
  Data: 29 Ottobre 2019
  Classe: 1B
"""

print("-"*60)
print("Programma per calcolare la MEDIA ARITMETICA di n numeri")
print("-"*60)

somma = 0

print("Di qunti numeri vuoi calcolare la media? ", end="")
n = int(input())

for i in range(1, n+1, 1):
    print("Inserire il valore n.", i,": ", end="")
    x = float(input());
    somma = somma + x

media = somma / n

print("La MEDIA ARITMETICA dei", n, "numeri è:", media)
```

3 Blocco for

Esercizio 4 Scrivere un programma che calcoli e stampi la somma dei numeri interi da 1 a 100.

```python
somma = 0
for i in range(1, 101, 1):
    somma = somma + i
print("La somma dei numeri interi da 1 a 100 è:", somma)
```

Il processo di determinare il massimo e/o il minimo di una sequenza di numeri è utilizzato frequentemente nelle applicazioni per computer. Si pensi per esempio al problema di determinare il miglior e il peggior venditore in una azienda oppure il vincitore in una gara cioè colui che ha totalizzato il punteggio migliore.

Esercizio 5 Scrivere un programma che legga 10 numeri interi in input e determini il maggiore e il minore dei numeri letti.

```
print("Inserire il valore 1): ", end="")
maggiore = int(input())
minore = maggiore

for i in range(2, 11, 1):
    print("Inserire il valore ", i, "): ", sep="", end="")
    numero = int(input())
    if ( numero > maggiore ):
        maggiore = numero
    if ( numero < minore ):
        minore = numero

print("Il maggiore è:", maggiore)
print("Il minore è:", minore)
```

Riferimenti bibliografici

[1] RICHARD L. HALTERMAN (September 5, 2019). *Fundamentals of Python Programming*, Southern Adventist University, https://python.cs.southern.edu/pythonbook/pythonbook.pdf