Programmazione con Python

Variabili, valori, tipi ed espressioni

Corso: Tecnologie informatiche

Docente: Leonardo Essam Dei Rossi

Licenze e crediti

Questo materiale è disponibile sul sito Web del docente per il corso di Tecnologie informatiche per le studentesse e gli studenti dell'anno scolastico 2025/2026.

Versione: 1.2.5 (A)
Ultima modifica: 01/05/2026 18:36

Introduzione

- I **numeri** e le **stringhe** di caratteri (*sequenze di caratteri*) sono importanti strutture di dati in qualsiasi programma Python:
  - Questi sono anche i componenti che usiamo per costruire strutture dati più complesse.
- In questa lezione impareremo a utilizzare numeri e testi;
- Scriveremo diversi semplici programmi che li utilizzano.

Sezione #1

Le variabili

Le variabili

- Una **variabile** è una zona di memoria dotata di un nome in un programma che fa riferimento ad un **valore** specifico;
- Ci sono diversi tipi di valori, ciascuno usato per memorizzare cose diverse:

Variabili	Valori
base	6
altezza	4
area	24.0
indirizzo	"Via Brigata Acqui, 15"
citta_residenza	"Trento"

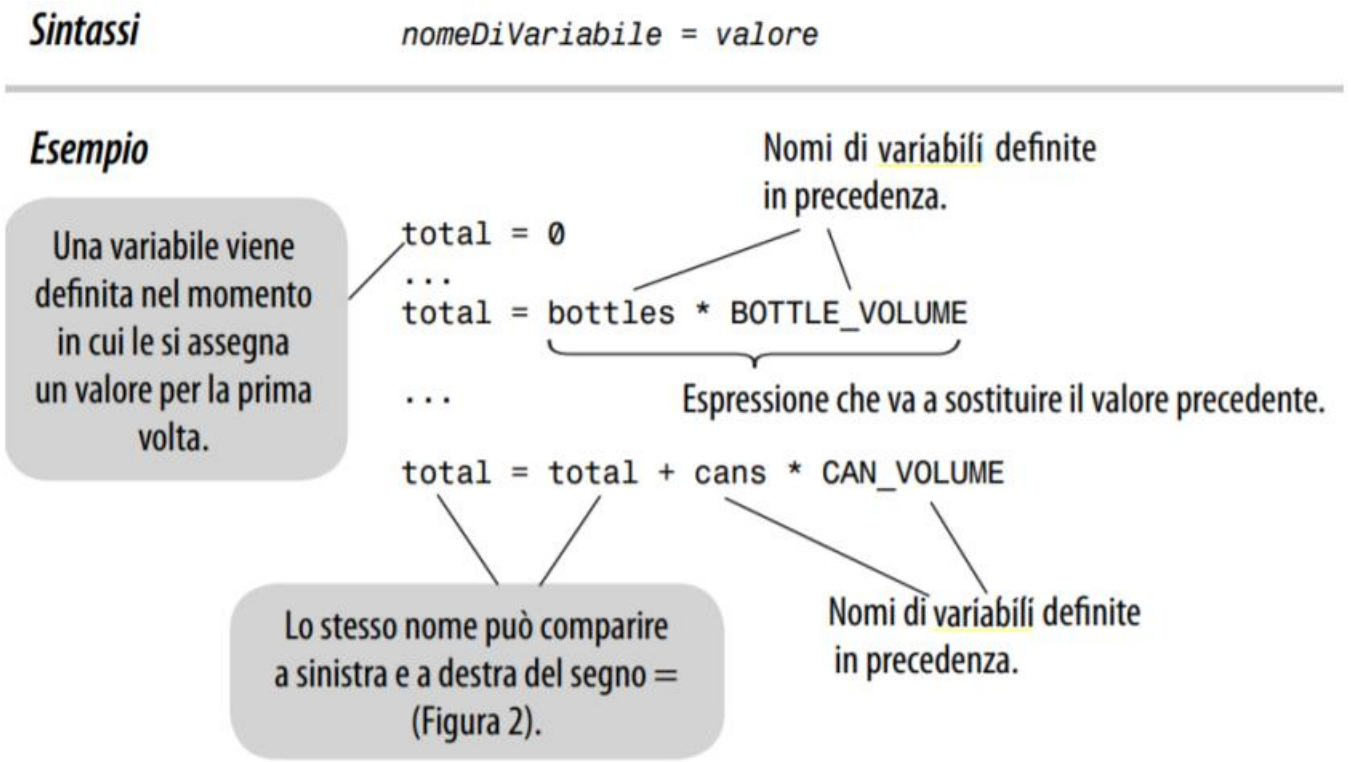
Le variabili

- Si «definisce» una variabile dicendo all'interprete:
  - Il **nome** scelto per la variabile;
  - Il **valore** iniziale della variabile.

- Si usa una **istruzione di assegnazione** per assegnare un valore alla variabile:
  - Il valore iniziale o un nuovo valore (che sostituisce il precedente).

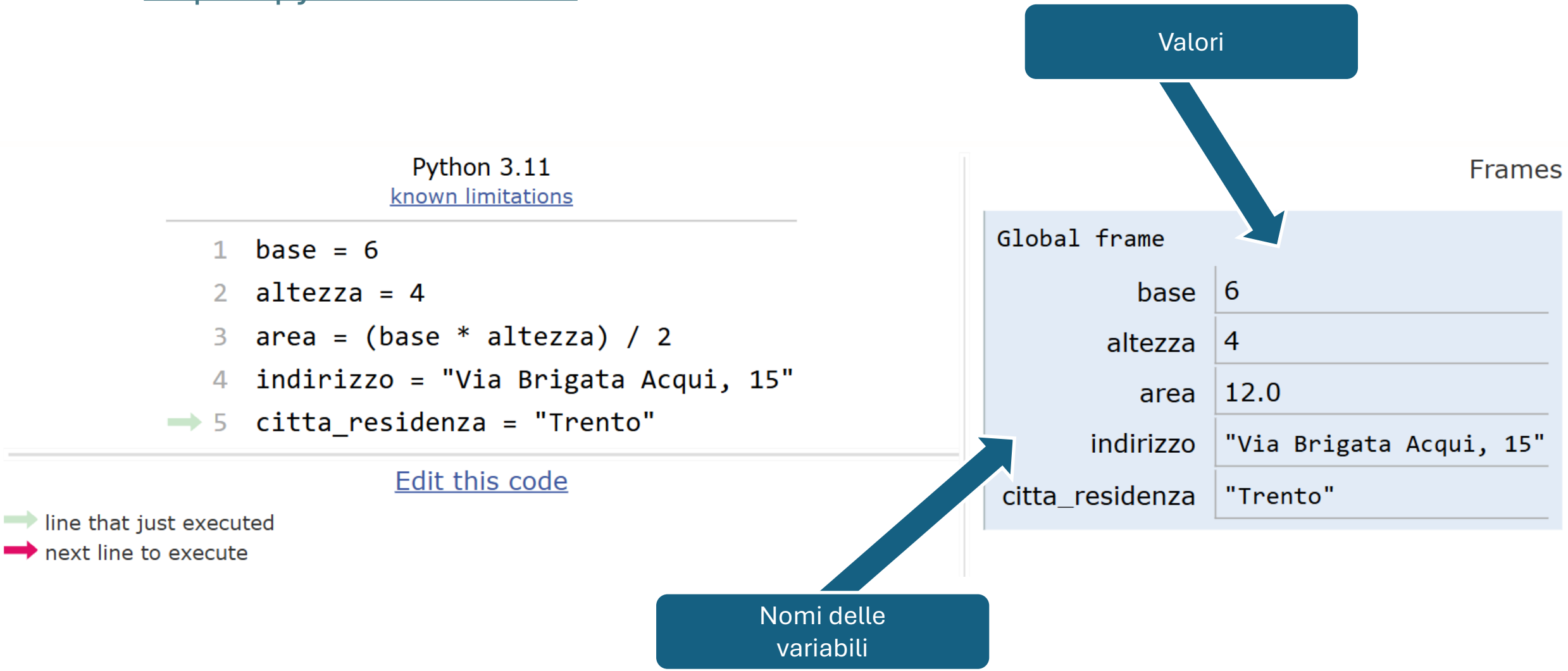
Definizione delle variabili

- Per **definire** una variabile bisogna specificarne un **nome** e un **valore iniziale:**
  - lattine = 4 # Definisce e inizializza la variabile ‘lattine’ con valore 4



Visualizzare le variabili

- Link: https://pythontutor.com/



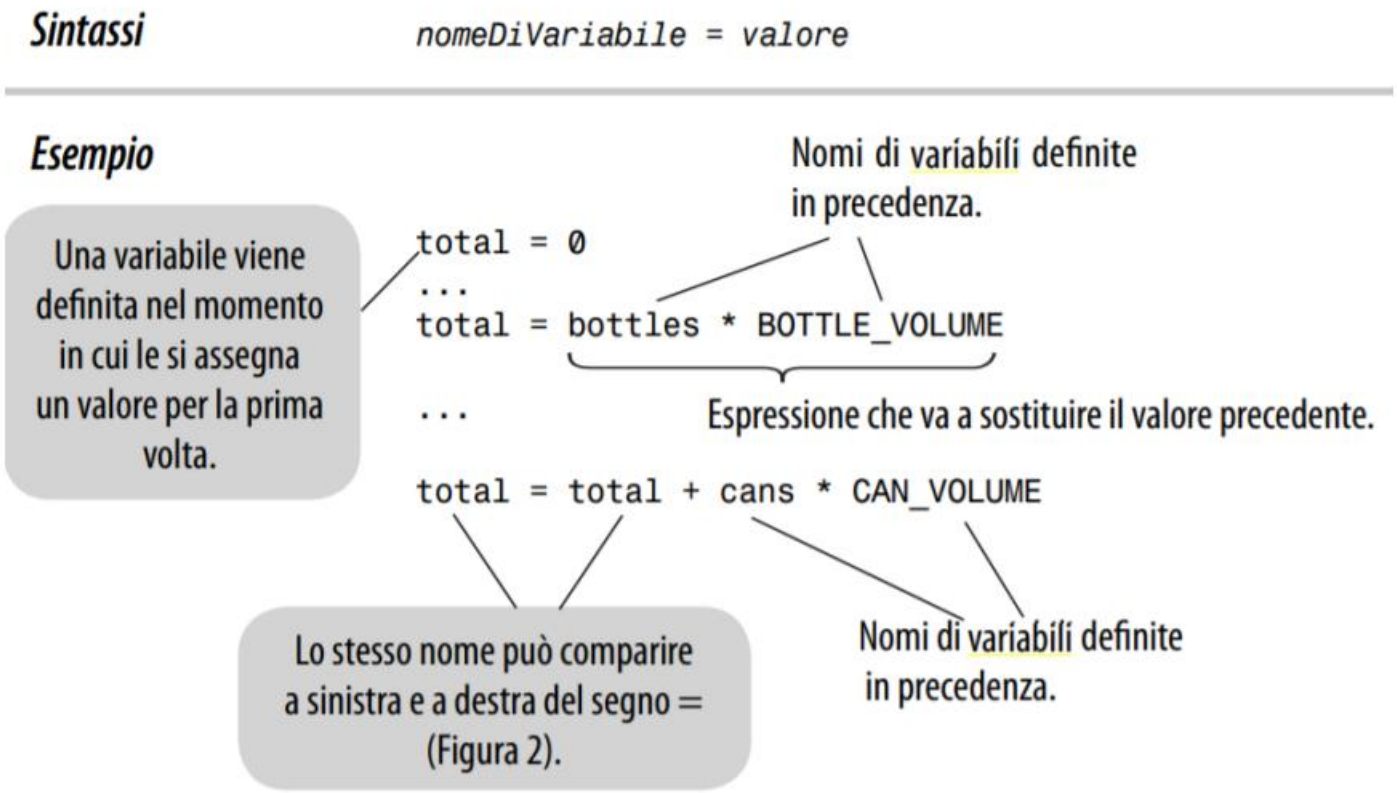
L’istruzione di assegnazione

- Usare **l’istruzione di assegnazione** (simbolo: =) per inserire un nuovo valore all’interno di una variabile:
  - Il nome della variabile si riferirà al nuovo valore:
    - `lattine = 4  # Definisce ‘lattine’ con valore 4`
    - `lattine = 6  # Cambia il valore associato a ‘lattine’ (diventa 6)`

- **Attenzione!** Il segno "=" **non rappresenta un confronto**:
  - Esso copia il valore alla destra del segno nella variabile alla sinistra del segno;
  - Vedrete l’operatore di verifica di uguaglianza nella prossima sezione.

Sintassi dell’assegnazione

- Il valore calcolato alla destra del segno "=" è assegnato alla variabile sulla sinistra.



Sezione #2

I tipi di dato

I tipi di dati

- Il **tipo** di dato è associato al valore e non alla variabile:
  - Una variabile può essere assegnata a **diversi valori di diverso tipo**, in diverse parti del programma.
  - `taxRate = 5` `# un intero`
  - `taxRate = 5.5` `# un numero con la virgola`
  - `taxRate = "Non-taxable"` `# una stringa`

- Se si usa una variabile ed essa è di un tipo non previsto, il programma darà errore.

Esempio

- Aprire VS Code e creare un nuovo progetto;
- Testare il seguente programma:
  - # Testing different types in the same variable

```
taxRate = 5                         # int
print(taxRate)

taxRate = 5.5                       # float
print(taxRate)

taxRate = "Non-taxable"             # string
print(taxRate)
```

Esempio

- Provare ora questo:
  - taxRate = "Non-taxable" # string
    print(taxRate)
    print(taxRate + 5)

```
Traceback (most recent call last):
  File "<python-input-2>", line 1, in <module>
    print(taxRate + 5)
          ~~~~~~~~~~^~~
TypeError: can only concatenate str (not "int") to str
```

- Cosa possiamo ricavare?

Esempio

- Una volta inizializzata una variabile con un valore di un particolare tipo si deve avere cura di continuare a salvare nella variabile valori del medesimo tipo!
- Vanno eseguite solamente le operazioni che sono valide a seconda del valore corrente della variabile:
  - Ricordarsi a che tipo si riferisce ogni variabile.
- Quando si utilizza l'operatore "+" con stringhe, il secondo argomento viene concatenato alla fine del primo, ma entrambi gli argomenti devono essere stringhe:
  - Le operazioni sulle stringhe verranno viste più avanti (forse).

Nomi di variabili in Python

Nome della variabile	Commento
canVolume1	I nomi delle variabili sono costituiti da lettere, da cifre e dal «trattino basso» (_).
x	In matematica si usano nomi di variabili brevi (come x o y). Anche in Python è permesso fare così, ma è una pratica poco comune perché rende poco comprensibile il programma.
CanVolume	Python è un linguaggio *case-sensitive*, quindi CanVolume != canVolume. Per convenzione (*) i nomi delle variabili iniziano con la lettera minuscola.
6pack	Il nome di una variabile non può iniziare con una cifra.
can volume	Il nome di una variabile non può contenere spazi.
class	La parola «class» è una istruzione riservata e non può essere usata come nome di variabile.
ltr/fl.oz	Non si possono usare simboli come «/» e «.» nei nomi delle variabili.

- **Q:** *che cos'è una convenzione?*

Nomi di variabili in Python

- In programmazione, è importante dare nomi descrittivi alle variabili:
  - Questo risulta particolarmente importante quando un programma è scritto da più persone.

Le costanti

- In Python una **constante** è una variabile il cui valore *non andrebbe modificato* dopo che le è stato assegnato un *valore iniziale;*
- È buona norma usare le maiuscole per nominare le costanti:
  - $\mathrm{FORZA\_GRAVITA} = 9.81$
- Python permette di modificare il valore di una costante:
  - Solo perché si può fare, non significa che si deve fare!

Le costanti

- È consuetudine usare le MAIUSCOLE per le costanti in modo da distinguerle dalle variabili:
  - È molto chiaro visivamente.

- Ad esempio:
  - `BOTTLE_VOLUME = 2` `# Constant`
  - `MAX_SIZE = 100` `# Constant`
  - `taxRate = 5` `# Variable`

I commenti

- È *cosa buona e giusta* utilizzare commenti all'inizio di ogni programma e chiarire i dettagli del codice;
- I commenti sono d'aiuto agli altri e un modo per tenere traccia del ragionamento:
  - Commentare serve ad aggiungere spiegazioni per chi legge il codice!
- Il compilatore/interprete ignora i commenti:
  - Altri programmatori li leggeranno;
  - Anche tu, un giorno.

I commenti

«Documentation is a love letter that you write to your future self.»

- *Damian Conway (2005)*

Sezione #2

Operazioni aritmetiche

Operatori aritmetici elementari

- Python supporta tutte le operazioni aritmetiche elementari:
  - Addizione (+)
  - Sottrazione (-)
  - Moltiplicazione (*)
  - Divisione (/)
  - Potenza (**)
- Usare le parentesi per scrivere le espressioni:

$$\frac{a+b}{2} \longrightarrow (\mathrm{a} + \mathrm{b}) / 2 \qquad b \times (1 + \frac{r}{100})^{n} \longrightarrow \mathrm{b} * ((1 + \mathrm{r} / 100) ** \mathrm{n})$$

Utilizzare diversi tipi numerici

- Se si usano numeri interi e a virgola mobile in un'espressione matematica, il risultato sarà un numero a virgola mobile:

```
>>> 7 + 4.0 # Porta al valore a virgola mobile
11.0
```

- 4 e 4.0 sono tipi di dato diversi per il computer;
- **Ricorda!** Se si usano stringhe con numeri interi o a virgola mobile, il risultato sarà un errore.

Divisione intera

- Quando si dividono due numeri interi con l'operatore /, si ottiene un valore a virgola mobile:
  - Esempio: $7 / 4$ dà 1.75

- Si può però anche eseguire una divisione intera usando l'operatore //:
  - L'operatore // calcola il quoziente e ignora la parte frazionaria:
    - Esempio: $7 // 4$ dà come risultato 1
  - Se gli operandi sono frazionari, effettua la divisione e poi tronca il risultato.

Calcolare il resto

- Se si è interessati al resto della divisione tra interi, va usato l'operatore % (detto modulo):
  - $\text{resto} = 7 \,\%\, 4$
- Il valore del resto sarà 3;
- Talvolta detto «modulo»;
- Usata prevalentemente tra numeri interi:
  - Per operandi frazionari, l'operazione non è particolarmente utile.

Esercizio #1: Convertire centesimi in dollari

- Si vuole realizzare un programma in Python che, dato il numero di centesimi, calcola quanti dollari corrispondono e i centesimi di resto;
  - Esempio: 123 centesimi => 1$ e 23 cent.
    100 centesimi => 1$ e 0 cent.
    99 centesimi => 0$ e 99 cent.