Programmazione con Python

Fondamenti del linguaggio

Corso: Informatica

Docente: Leonardo Essam Dei Rossi

Licenze e crediti

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

Versione: 1.0.0 (A)
Ultima modifica: 05/05/2026 09:14

Sezione #1

Introduzione, link utili e installazione di Python

Caratteristiche generali

- Python è un linguaggio che permette di creare programmi in modo facile e veloce:
  - Possiede una sintassi più semplice rispetto ad altri linguaggi come C e C++.

- Nasce nel 1991 ed è un **linguaggio interpretato**.

Definizione: *linguaggio interpretato*
Un linguaggio interpretato è un linguaggio di programmazione in cui le istruzioni vengono eseguite direttamente da un programma chiamato «interprete», senza una fase di compilazione separata in linguaggio macchina.

Link utili

- https://www.python.org/
  - Sito web ufficiale
- https://docs.python.org/3/
  - Documentazione ufficiale (inglese / italiano / …)
- https://realpython.com/
  - Molti tutorial a diversi livelli di approfondimento

L’IDE di Visual Studio Code

- Visual Studio Code (link) è un software **IDE** (*Integrated Development Environment*):
  - Si occupa di aiutare i programmatori a scrivere codice;
  - Se avete fatto i compiti di informatica, dovreste saperlo usare! 😊

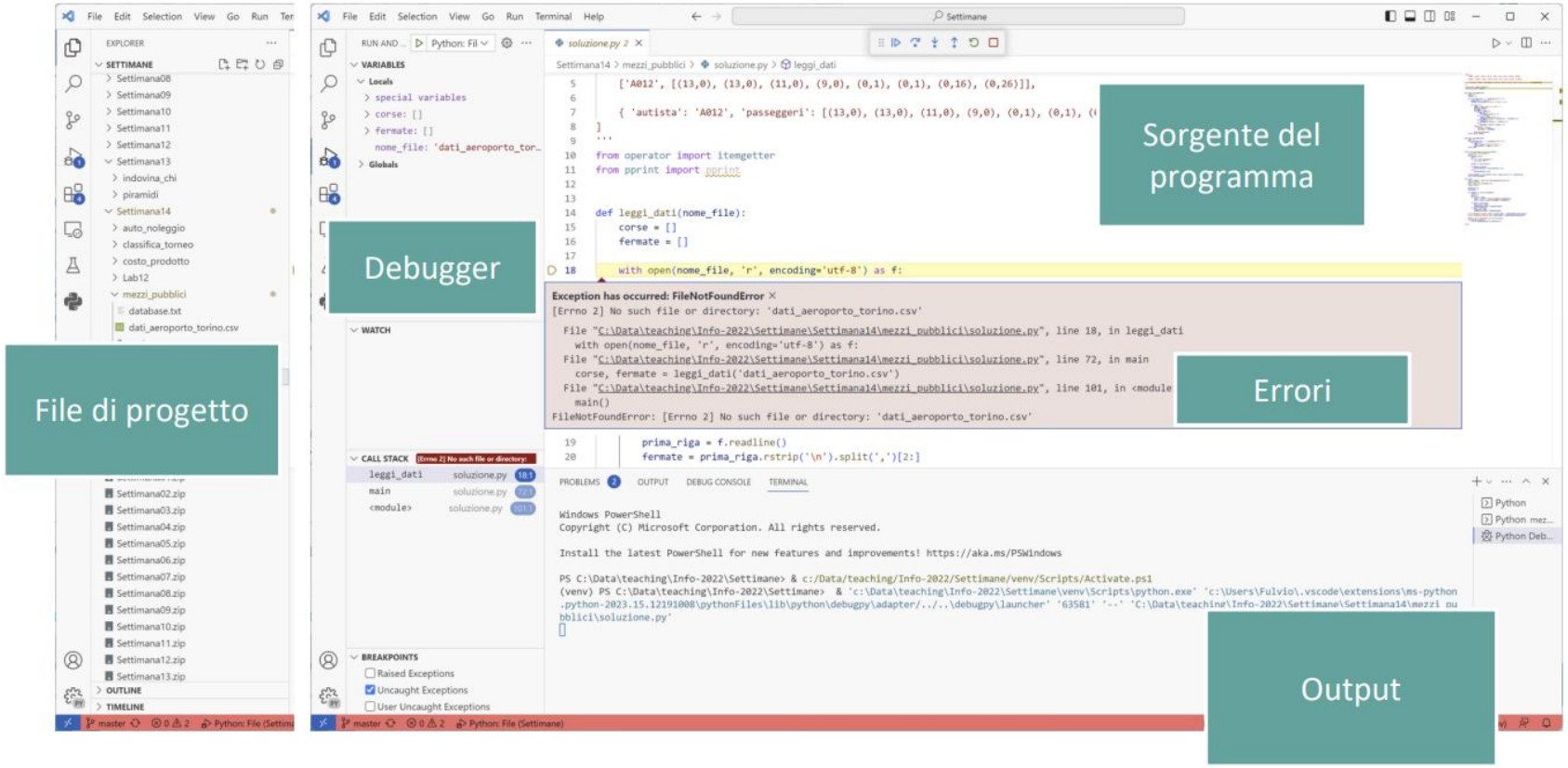
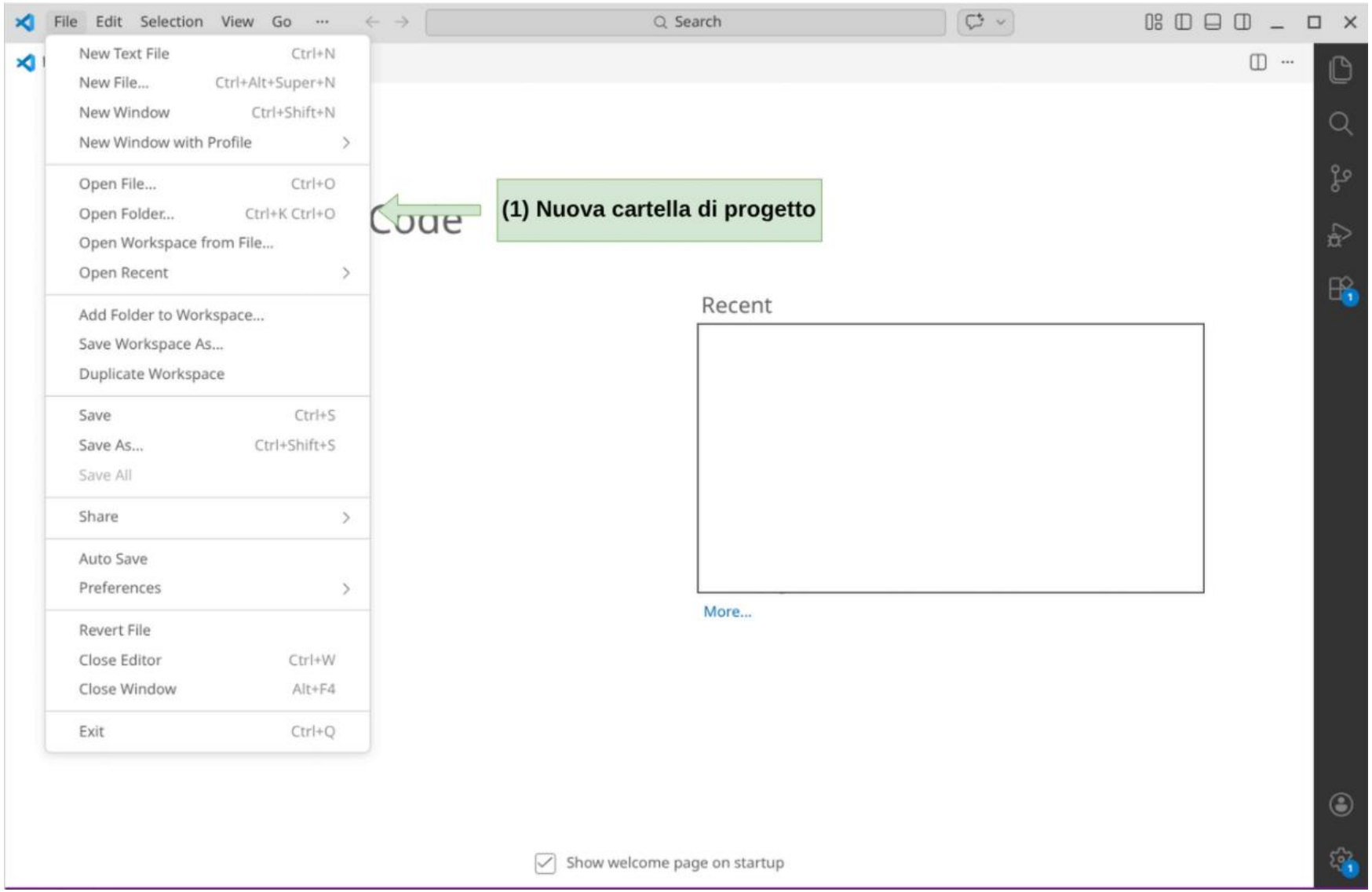
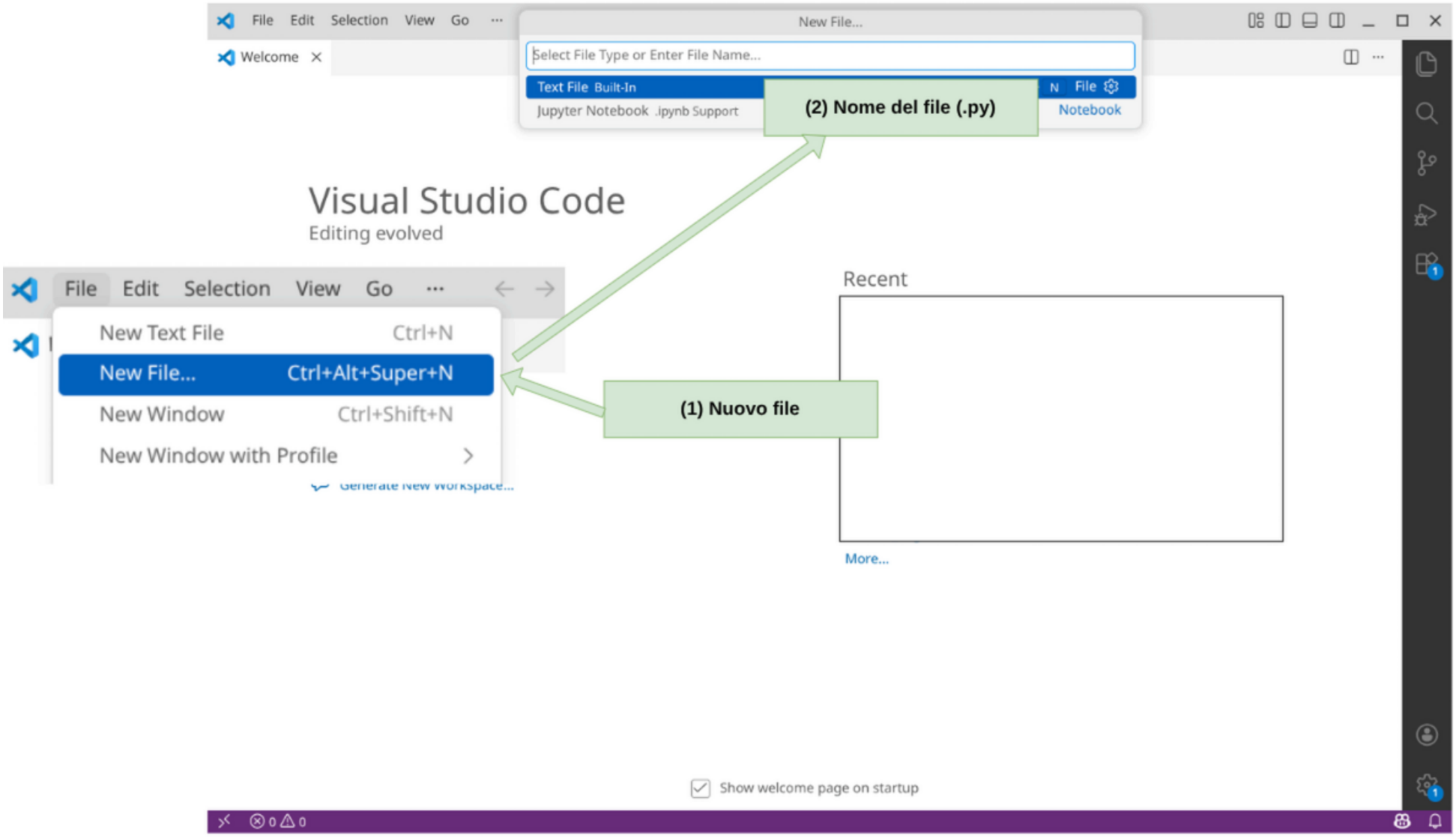


Immagine del prof. Fulvio Corno (Politecnico di Torino)

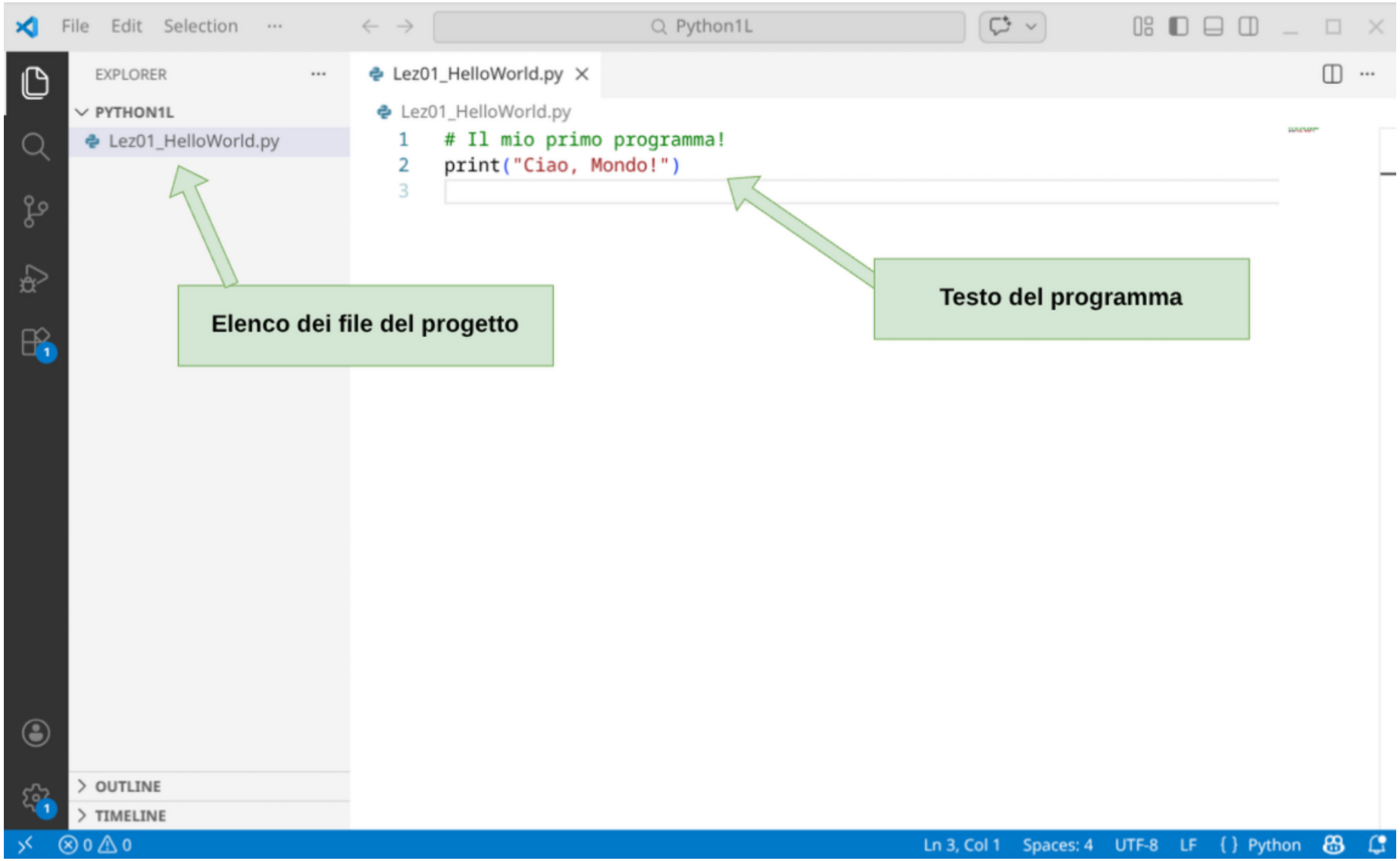
Creare un nuovo progetto



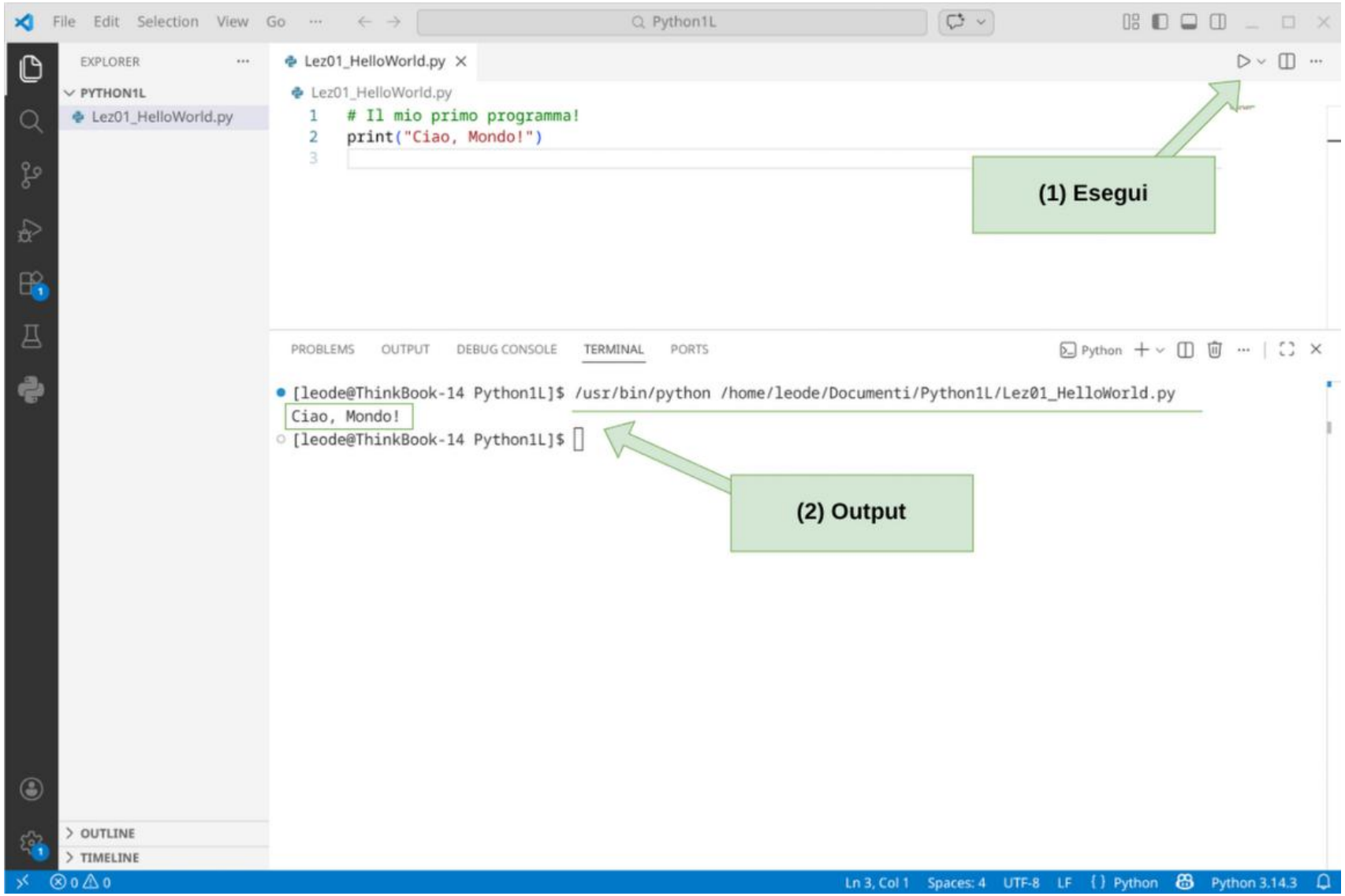
Creare un nuovo progetto



Creare un nuovo progetto



Creare un nuovo progetto



Creare un nuovo progetto

- Provate voi ad eseguire il programma «Ciao, Mondo!»:

Esempio: *programma «Ciao, Mondo!»*

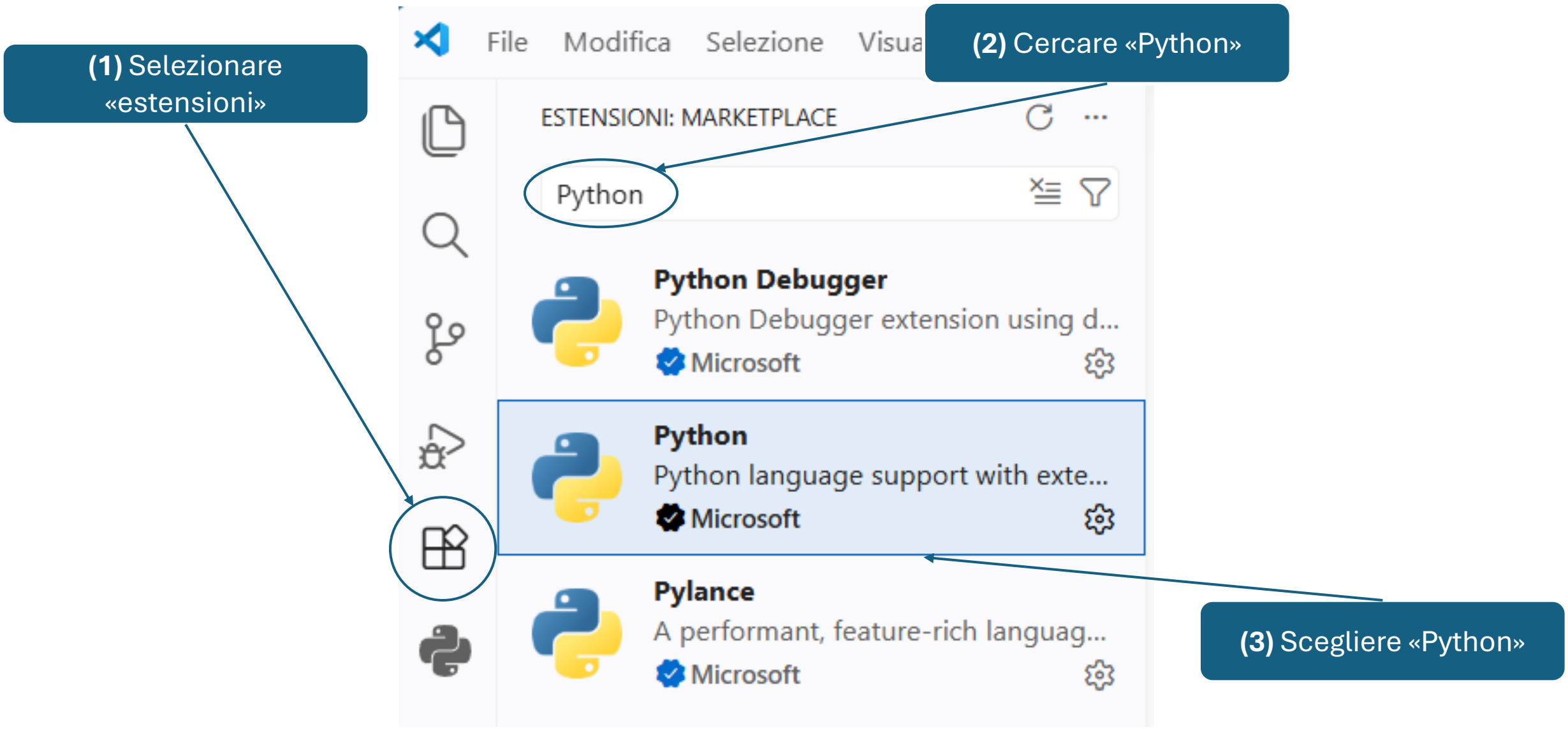
```
# Il mio primo programma!
print("Ciao, Mondo!")
```

Q: *Manca qualcosa?*

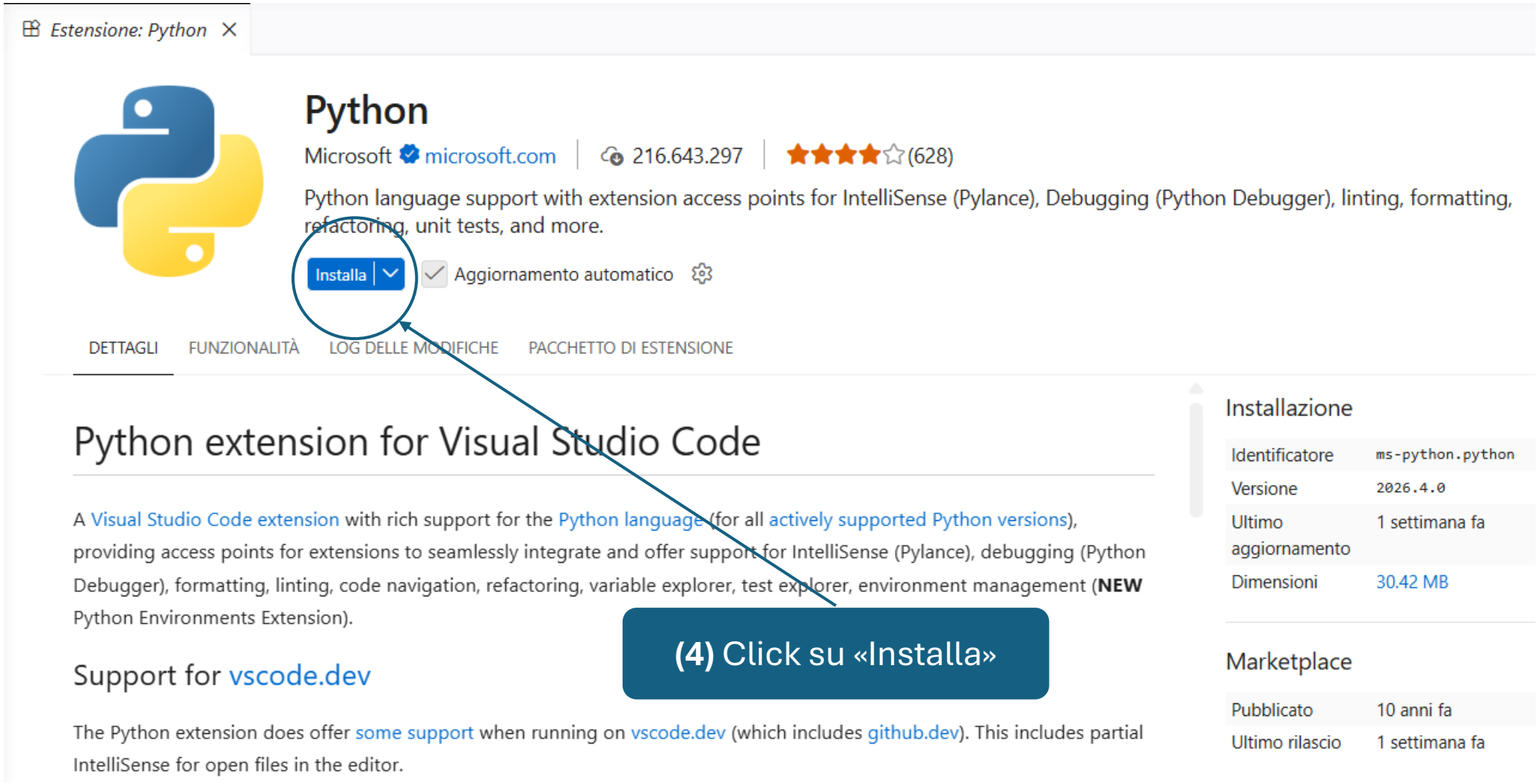
L’IDE di Visual Studio Code

- VS Code è un IDE di tipo generico;
- Ciò significa che supporta un grande numero di linguaggi di programmazione, ma dobbiamo essere noi a selezionare per quali linguaggi installare il supporto:
  - Questo si ottiene utilizzando le estensioni disponibili in VS Code.

Installare un’estensione



Installare un’estensione



Sezione #2

Variabili, valori, tipi ed espressioni

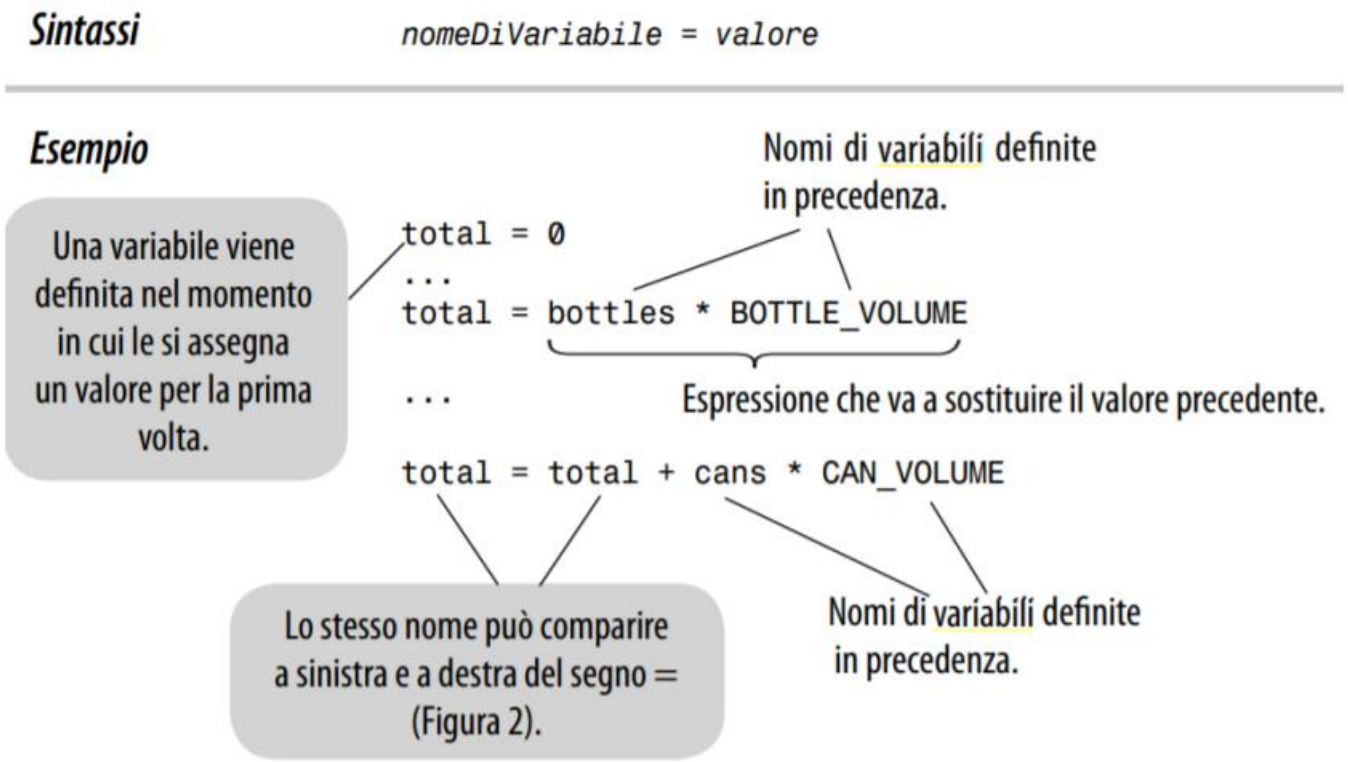
Le variabili

- 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"

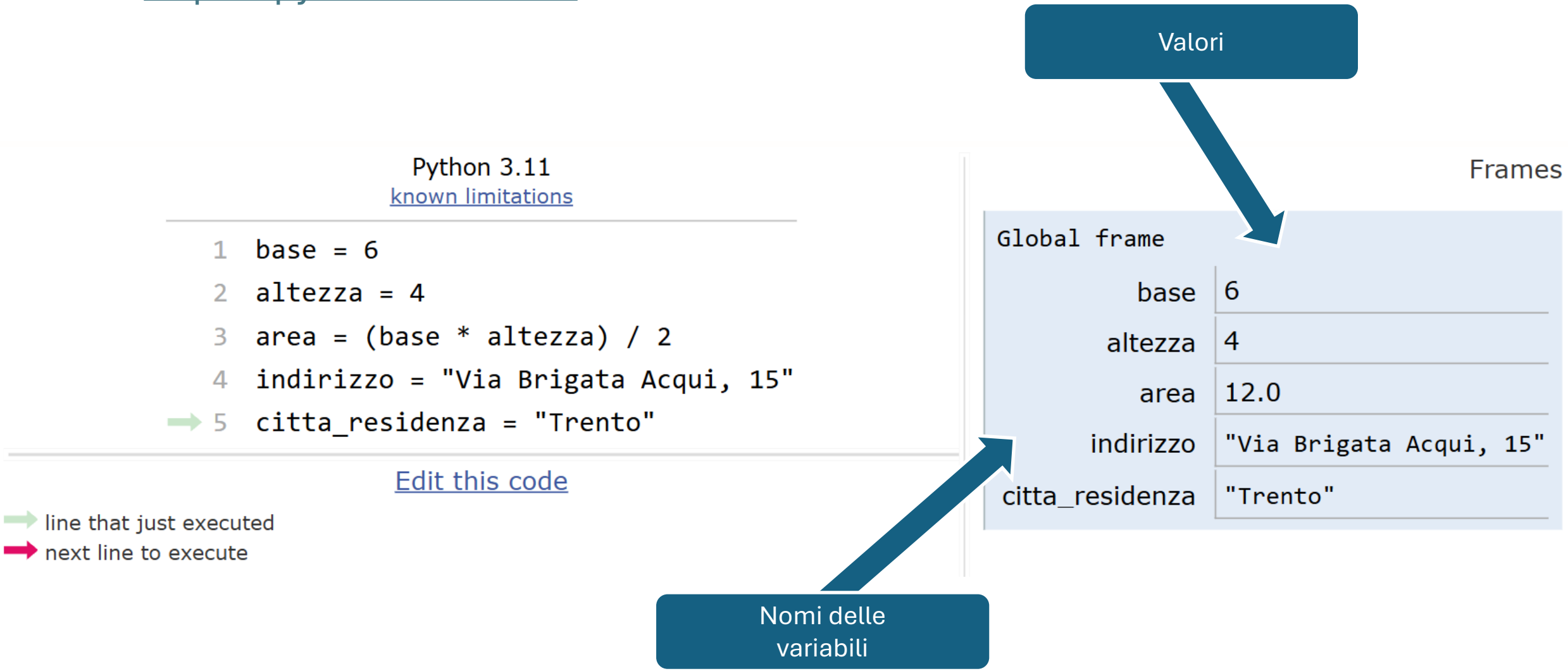
Definizione delle variabili

- 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.~~

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:
  - `miaVariabile = ToyotaYaris2022` `# Il tipo di dato "ToyotaYaris2022" non esiste!`

Combinazione di tipi di dati

- In Python valgono alcune regole:
  - Numero + numero = numero $(5 + 5 = 10)$
  - Stringa + stringa = stringa ("a" + "b" = "ab")

- Numero + stringa =

```
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
```

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.

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!*

I commenti

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

- *Damian Conway (2005)*

Sezione #3

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.

Sezione #4

Le liste e operazioni con le liste

Le liste

- Una lista è una collezione ordinata (indicizzata);
- I suoi elementi:
  - Non sono necessariamente tutti dello stesso tipo di dato;
  - Possono essere modificati ~~(a differenza delle tuple)~~;

Creazione di una lista

- Una lista è definita da una sequenza di elementi racchiusa tra [e]:

```
>>> myLista = [1, 2, "Ciao", 3.14]
>>> print(myLista)
[1, 2, 'Ciao', 3.14]
```

- Un elemento di una lista può essere a sua volta una lista di qualunque tipo:

```
>>> myLista2 = [1, 2, [3, 4], "Ciao", ["Leonardo", "Dei Rossi"], (9, 8, 7)]
>>> print(myLista2)
[1, 2, [3, 4], 'Ciao', ['Leonardo', 'Dei Rossi'], (9, 8, 7)]
```

Accedere a un elemento di una lista

- Per accedere al singolo elemento di una lista se ne indica l'indice tra [e]:

```
>>> myLista1 = ["Ciao", "come", "stai", "?"]
>>> print(myLista1[1])
come
```

- Python utilizza lo zero (0) come primo indice!

Modificare un elemento di una lista

- Una lista è una struttura dati mutabile:
  - È possibile modificarne i valori anche dopo la dichiarazione.

```
>>> myLista1 = ["Ciao", "come", "stai", "?"]
>>> print(myLista1[1])
come
>>> myLista1[1] = "Pino"
>>> print(myLista1[1])
Pino
```

Aggiungere elementi a una lista

- Sia la lista: myLista = [1, 2, 3, 4, 5]
- Usando il metodo .append(element), element viene aggiunto in fondo alla lista:
  - myLista.append(6) => [1, 2, 3, 4, 5, 6]
- Usando il metodo .insert(i, element), element viene aggiunto nella i-esima posizione:
  - myLista.insert(1, 6) => [1, 6, 2, 3, 4, 5]

Rimuovere elementi da una lista

- Sia la lista: myLista = [1, 2, 3, 4, 5]

- Usando il metodo .pop(), viene estratto (restituito) e rimosso l'ultimo elemento:
  - myLista.pop() => [1, 2, 3, 4] # .pop() restituira' 5

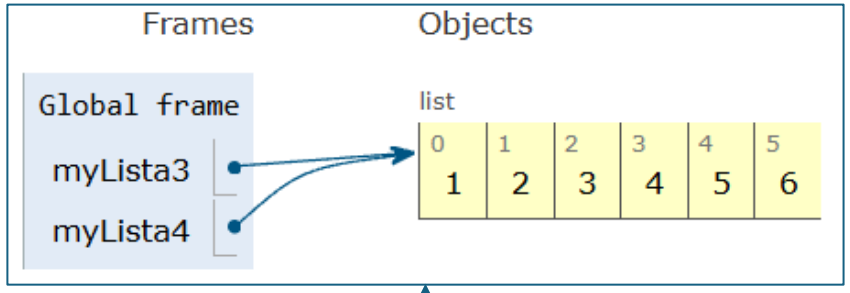
- Usando il metodo .del(i), viene estratto e rimosso l' i-esimo elemento:
  - myLista.del(1) => [1, 3, 4, 5] # .del(1) restituira' 2

Unione di liste

- In Python, usando l’operatore "+" tra due liste, si ottiene una nuova lista con gli elementi della prima e della seconda:

```
>>> myLista1 = [1, 2, 3]
>>> myLista2 = [4, 5, 6]
>>> myLista3 = myLista1 + myLista2
>>> print(myLista3)
[1, 2, 3, 4, 5, 6]
```

Copia di liste



- È possibile assegnare una lista esistente ad una nuova variabile:

```
>>> myLista4 = myLista3
>>> print(myLista3)
[1, 2, 3, 4, 5, 6]
>>> print(myLista4)
[1, 2, 3, 4, 5, 6]
```

Quello che si ottiene non è una copia della lista, ma un **riferimento alla lista originale**!!!

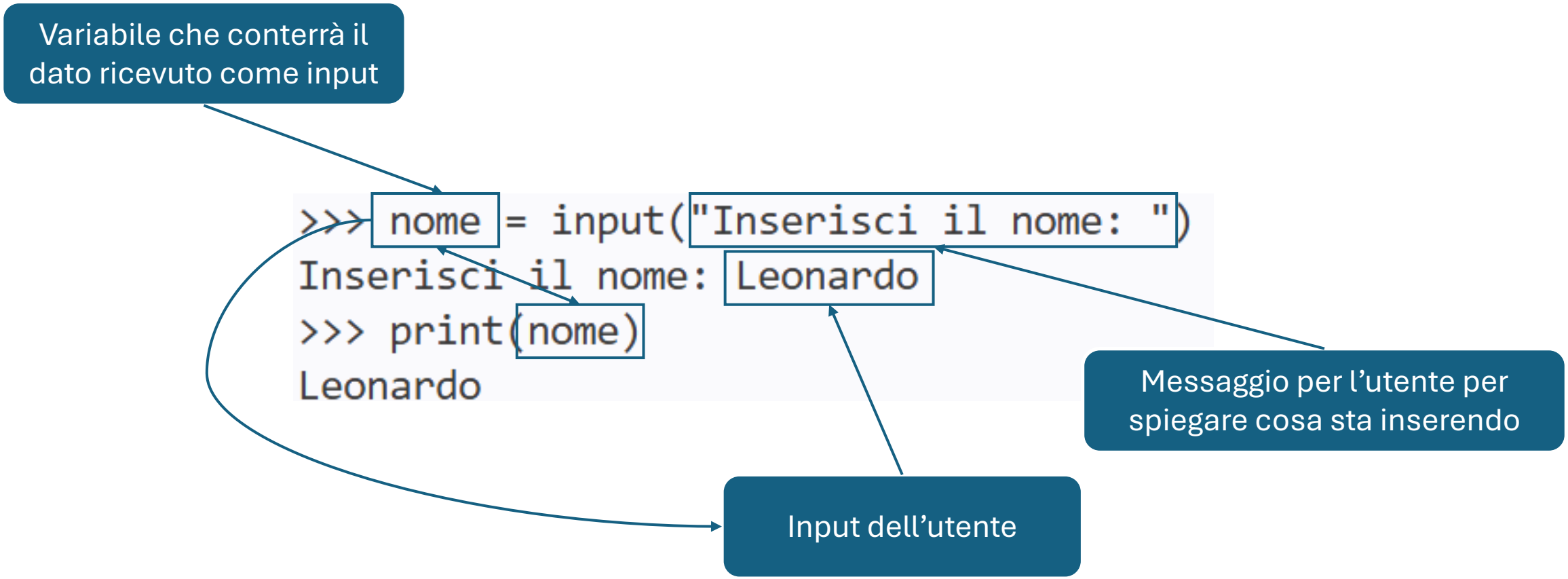
Attenzione! Modificando un elemento da myLista3, esso viene modificato anche in myLista4.

Sezione #5

Input dall’utente

Input dall’utente

- Per ricevere un valore dall’utente (input) si usa l’omonima funzione:



Esempio

Esempio: *programma che legge due numeri (v. 1)*

Si vuole realizzare un programma che legge come due numeri a e b e stampa la loro somma.

Svolgimento:

```
a = input("Inserisci a: ")          # Leggo il valore di a (1)
b = input("Inserisci b: ")          # Leggo il valore di b (2)
c = a + b                           # Faccio la somma
print(c)                            # Stampo il risultato
```

Output:

```
12                                  # ?????
```

Q: *Cos'è successo?*

Esempio

- La funzione $\mathrm{input}()$ restituisce il dato inserito dall'utente;
- Questo dato ha come tipo una stringa!

Q: *Come risolviamo?*

Esempio

Esempio: *programma che legge due numeri (v. 2)*

Si vuole realizzare un programma che legge come due numeri a e b e stampa la loro somma.

Svolgimento:

```
a = int(input("Inserisci a: "))          # Leggo il valore di a (1)
b = int(input("Inserisci b: "))          # Leggo il valore di b (2)
c = a + b                                # Faccio la somma
print(c)                                 # Stampo il risultato
```

Output:

```
3                                        # :)
```

Casting

- Python di natura è un linguaggio **debolmente tipizzato**:
  - Il tipo di dato di una variabile può cambiare a *runtime*;
  - Il tipo di dato di una variabile viene determinato automaticamente:
    - Alla dichiarazione della variabile;
    - Ad ogni successiva ri-assegnazione.

- Per «forzare» il tipo di dato (*vedasi esempio di prima*), esistono delle funzioni per fare il *cast* al tipo di dato desiderato.

Casting

Funzione	Commento
int(…)	Costruisce un numero intero a partire da una stringa di un numero intero (es. "3"), una stringa di un numero in virgola mobile (es. "3.14", rimuovendo tutte le cifre decimali). Se la stringa non è un numero succede ValueError.
float(…)	Uguale a int(…), con l'aggiunta che vengono mantenute le cifre decimali.
str(…)	Costruisce una stringa dato un qualsiasi tipo di dato(intero, decimale, …).

Sezione #6

I moduli di sistema e le librerie

I moduli standard

- Uno dei punti di forza di Python è la grande quantità di moduli predefiniti messi a disposizione di chi scrive il codice;

- Tra i moduli pre-caricati e che si possono importare ce ne sono alcuni di molto utili:
  - Modulo string;
  - Modulo math;
  - Modulo random;
  - Modulo platform;
  - *E molti altri…*

Esempio: il modulo platform

- Importiamo il modulo platform con la keyword import:

```
>>> import platform
```

- Invochiamo il metodo .system() per vedere il sistema operativo in uso:

```
>>> import platform
>>> print(platform.system())
Windows
```