Python — Cheat Sheet

Marini Mattia

November 08, 2026

1 Variabili

Cosa sono

Una **variabile** è un nome che punta a un valore: una "scatola" etichettata in cui mettiamo un dato per riutilizzarlo.

```
nome = "Ada"       # crea la variabile "nome"
eta = 36           # crea la variabile "eta"
eta = 37           # aggiorna: ora eta vale 37
```

- `nome` e `eta` sono i **nomi**;
- `"Ada"` e `36` sono i **valori**;
- `=` si legge "**metti dentro**", **non** "è uguale a";
- una variabile va **inizializzata prima di usarla** (altrimenti `NameError`).

I tipi più comuni

Python capisce il **tipo** da solo, guardando il valore scritto.

```
x  = 5             # int   → numero intero
p  = 3.14          # float → numero con la virgola
s  = "ciao"        # str   → stringa (testo tra virgolette)
v  = [6, 7, 8]     # list  → lista di valori
ok = True          # bool  → vero / falso

print(x)           # 5
print(type(x))     # <class 'int'>
```

Liste: uso base

```
voti = [6, 7, 8]
voti[0]            # 6   ← si parte da 0!
voti[2] = 9        # modifica il terzo elemento
len(voti)          # 3   quanti elementi
voti.append(10)    # aggiunge in fondo
```

> **N.B.** Il primo elemento è in posizione `0`, non `1`.

print(): la voce del programma

```
print("Ciao")      # Ciao
print(7 + 3)       # 10
print("x =", 5)    # x = 5
```

2 if / elif / else

Sintassi

```
if condizione:
    # blocco eseguito SOLO se la condizione è True
```

- `if` + **condizione** + **due punti** `:`;
- il corpo è **indentato** (1 tab);
- il blocco parte solo se la condizione è `True`.

if / elif / else

```
voto = 7
if voto >= 9:
    print("Ottimo")
elif voto >= 6:
    print("Sufficiente")
else:
    print("Insufficiente")
```

- `if` → primo controllo, **obbligatorio**;
- `elif` → controllato **solo se** i precedenti erano falsi;
- `else` → nessuna condizione, ultima alternativa (può mancare);
- di tutta la catena si esegue **un solo blocco**.

if vs elif

```
# if separati: si controllano TUTTI
if x > 0:
    print("positivo")
if x > 10:
    print("grande")   # può stampare anche questa
```

```
# if / elif: si ferma al primo vero
if x > 0:
    print("positivo")
elif x > 10:
    print("grande")   # NON controllata se x > 0 è vera
```

`elif` serve quando le alternative si **escludono a vicenda**.

Le condizioni

```
5 == 5    # True   uguale
5 != 3    # True   diverso
5 > 3     # True   maggiore
5 <= 5    # True   minore o uguale
```

Per combinarle: `and` (entrambe vere), `or` (almeno una), `not` (nega).

```
if voto >= 6 and voto <= 10:
    print("Voto valido e sufficiente")
```

> **N.B.** `=` **assegna**, `==` **confronta**: `if x = 5:` è un errore.

3 Ciclo while

Sintassi

```
while condizione:
    # corpo: ripetuto finché la condizione è vera
```

Esempio completo

```
i = 0              # 1. inizializza la variabile di controllo
while i < 3:       # 2. condizione, controllata prima di ogni giro
    print(i)       #    corpo del ciclo
    i = i + 1      # 3. aggiorna: senza questo → ciclo infinito
print("fine")
```

Output: `0 1 2 fine`.

> **N.B.** Per capire un ciclo fai la **traccia**: esegui il codice a mano, una riga alla volta, annotando il valore di `i`.

controllo	corpo	i dopo
`0 < 3` vero	stampa `0`	1
`1 < 3` vero	stampa `1`	2
`2 < 3` vero	stampa `2`	3
`3 < 3` falso	esci	3

Attenzione

- se la condizione non diventa mai falsa ⟶ **ciclo infinito**;
- la variabile di controllo va aggiornata **dentro** il ciclo.
- il corpo può non essere eseguito **nemmeno una volta** (se la condizione è subito falsa).

4 Esempi ultra comuni

Scorrere i numeri da 0 a n-1

```
n = 5
i = 0
while i < n:
    print(i)       # 0 1 2 3 4
    i = i + 1
```

La scorciatoia più usata fa la stessa cosa:

```
for i in range(n):
    print(i)       # 0 1 2 3 4
```

Numero pari o dispari

`%` è il **resto** della divisione: `n % 2` vale `0` se `n` è pari.

```
n = 8
if n % 2 == 0:
    print("pari")
else:
    print("dispari")
```

Sommare i numeri da 1 a n

```
n = 5
somma = 0
i = 1
while i <= n:
    somma = somma + i
    i = i + 1
print(somma)       # 15
```

Scorrere una lista elemento per elemento

```
voti = [6, 7, 8]
i = 0
while i < len(voti):
    print(voti[i])
    i = i + 1
```

Scorciatoia:

```
for v in voti:
    print(v)
```

Contare i pari in una lista

```
numeri = [3, 8, 5, 10]
conta = 0
i = 0
while i < len(numeri):
    if numeri[i] % 2 == 0:
        conta = conta + 1
    i = i + 1
print(conta)       # 2
```

5 Errori ultra comuni

Errore di sintassi: leggi VS Code

Quando scrivi qualcosa che Python non capisce, il programma **non parte** e compare un **SyntaxError** con la **riga** e una breve spiegazione. In VS Code il pezzo sbagliato è **sottolineato in rosso**: passa il mouse sopra e leggi il messaggio. Non tirare a indovinare: **leggi l'errore**.

Dimenticare i due punti `:`

```
if x > 0           # ✗ SyntaxError: expected ':'
    print("sì")

if x > 0:          # ✓
    print("sì")
```

Vale per `if`, `elif`, `else`, `while`.

Indentazione

```
if a < b:
print("a")        # ✗ IndentationError: manca il rientro

if a < b:
    print("a")    # ✓ (1 tab)
if a < b:
    print("a")
   a = a + 1      # ✗ stesso blocco = stessa indentazione
```

Non mischiare **tab** e **spazi**.

Confondere `=` con `==`

```
if x = 5:          # ✗ assegna, non confronta
if x == 5:         # ✓ confronta
```

Confondere `<` con `<=`

```
while i < 3:     # si ferma a 2 → 0 1 2
while i <= 3:    # include il 3 → 0 1 2 3
```

Chiediti sempre: l'estremo è **incluso** o no?

Altri classici

- **Maiuscole**: `Voto` e `voto` sono variabili diverse.
- **Nome inesistente** o variabile usata prima di crearla ⟶ `NameError`.
- **Ciclo infinito**: hai dimenticato `i = i + 1`.
- **Parentesi o virgolette non chiuse** ⟶ `SyntaxError`.
- **else con una condizione**: `else` non ne vuole (`else:`).
- **Indentazione incoerente**: Python la controlla **prima** di eseguire.