

# Variabili, tipi di dati e operatori

---

## Cos'è una variabile?

---

Una variabile è tipo una **scatola con un'etichetta**. Ci metti dentro un valore, le dai un nome, e poi la usi quando ti serve. Facile, no?


In Python creare una variabile è semplicissimo: le dai un nome, metti `=`, e ci piazzì il valore. Fine. Niente dichiarazioni tipo `Declare Integer x` come in Flowgorithm — Python è più rilassato!

```
nome = "Mario"
eta = 16
altezza = 1.75

print(nome)
print(eta)
print(altezza)
```

!!! tip "Niente dichiarazioni!"

In Flowgorithm dovevi fare ``Declare Integer x`` prima di usare una variab



---

## I tipi di dati fondamentali

---

Python ha 4 tipi di dati base. Pensa a loro come le 4 "categorie" in cui Python organizza le informazioni:

Tipo	Nome Python	Esempio	Tradotto
Intero	<code>int</code>	<code>42</code> , <code>-7</code> , <code>0</code>	Numeri senza virgola
Decimale	<code>float</code>	<code>3.14</code> , <code>-0.5</code>	Numeri con il punto!
Stringa	<code>str</code>	<code>"ciao"</code> , <code>'ok'</code>	Testo tra virgolette
Booleano	<code>bool</code>	<code>True</code> , <code>False</code>	Vero o Falso, tipo sì/no

```
# La funzione type() ti dice che tipo è una variabile
# Utile quando non capisci perché il codice non funziona!
intero = 42
decimale = 3.14
testo = "ciao"
booleano = True

print(type(intero))
print(type(decimale))
print(type(testo))
print(type(booleano))
```

!!! warning "Il punto, non la virgola!"

In Python (e in tutti i linguaggi) i decimali usano il **\*\*punto\*\***, non la

## Regole per i nomi delle variabili

Non puoi dare un nome a caso alle variabili (anche se a volte vorresti chiamarle "roba123"). Ecco le regole:

- **Possono contenere:** lettere, numeri e underscore `_`
- **Devono iniziare** con una lettera o underscore (mai con un numero!)
- **Sono case-sensitive:** `nome` e `Nome` sono variabili DIVERSE (sì, Python è pignolo)
- **Non possono** essere parole riservate ( `if` , `for` , `print` ... Python se le tiene per sé)

Nome	Valido?	Perché
eta	Sì	Solo lettere, tutto ok
nome_utente	Sì	Lettere e underscore, perfetto
x1	Sì	Lettera seguita da numero
1nome	No	Inizia con un numero, vietato!
nome utente	No	Lo spazio non è ammesso
for	No	Parola riservata, Python si arrabbia

!!! tip "Convenzione snake\_case 🐍"

In Python si usa lo **snake\_case**: parole minuscole separate da underscore

## Operatori aritmetici

Ok, qui siamo nel territorio della matematica, ma niente panico: Python sa fare i conti meglio di te (nessuna offesa 😊).

Operatore	Significato	Esempio	Risultato
+	Addizione	5 + 3	8
-	Sottrazione	5 - 3	2
*	Moltiplicazione	5 * 3	15
/	Divisione	5 / 2	2.5
//	Divisione intera	5 // 2	2
%	Resto (modulo)	5 % 2	1
**	Potenza	5 ** 2	25

```
a = 17
b = 5

print("Addizione:", a + b)
print("Sottrazione:", a - b)
print("Moltiplicazione:", a * b)
print("Divisione:", a / b)
print("Divisione intera:", a // b)
print("Resto:", a % b)
print("Potenza:", a ** b)
```

!!! tip "Il resto `%` è più utile di quanto pensi!"

L'operatore ``%`` (modulo) restituisce il **resto** della divisione. Sembr

## Precedenza degli operatori

Come in matematica, le operazioni hanno un **ordine di esecuzione**. Se te lo sei dimenticato, tranquillo, è uguale a quello che hai imparato a scuola:

1. `**` (potenza) — si esegue per prima
2. `*`, `/`, `//`, `%` (moltiplicazione & co.)
3. `+`, `-` (addizione e sottrazione)

E se vuoi cambiare l'ordine? Usa le **parentesi** `()`, esattamente come in matematica!

```
# Senza parentesi: prima * poi +
risultato1 = 2 + 3 * 4
print("2 + 3 * 4 =", risultato1)

# Con parentesi: prima + poi *
risultato2 = (2 + 3) * 4
print("(2 + 3) * 4 =", risultato2)
```

## Conversione dei tipi (casting)

A volte devi **convertire** un valore da un tipo all'altro. È come tradurre da una lingua all'altra, ma per i dati:

Funzione	Converte in	Esempio
<code>int()</code>	Intero	<code>int("42")</code> → 42
<code>float()</code>	Decimale	<code>float("3.14")</code> → 3.14
<code>str()</code>	Stringa	<code>str(42)</code> → "42"
<code>bool()</code>	Booleano	<code>bool(1)</code> → True

```
# Conversione stringa → numero
testo = "100"
numero = int(testo)
print(numero + 50) # Ora funziona!

# Conversione numero → stringa
eta = 16
messaggio = "Ho " + str(eta) + " anni"
print(messaggio)
```

### La trappola dell'input (te l'avevo detto!)

Ricordi l'avvertimento della sezione precedente? Ecco, questo è il momento in cui capisci perché era importante:

```
# CORRETTO: converto in int prima di calcolare
numero = int(input("Scrivi un numero: "))
doppio = numero * 2
print("Il doppio è:", doppio)
```

Se non metti `int()` intorno a `input()`, Python pensa che sia testo e invece di moltiplicare... **ripete la stringa!** Scrivi 5, ti aspetti 10, e ottieni "55". Benvenuto nel club "ho perso mezz'ora per un errore stupido" 😂

## Operatori di assegnazione composti

Scrivere `x = x + 3` funziona, ma è un po' lungo. Python ti offre delle **scorciatoie** perché anche lui è pigro:

Operatore	Equivale a	In pratica
<code>+=</code>	<code>x = x + valore</code>	aggiunge
<code>-=</code>	<code>x = x - valore</code>	toglie
<code>*=</code>	<code>x = x * valore</code>	moltiplica
<code>/=</code>	<code>x = x / valore</code>	divide

```
punti = 100
print("Punti iniziali:", punti)

punti += 50 # Guadagno 50 punti
print("Dopo bonus:", punti)

punti -= 30 # Perdo 30 punti
print("Dopo penalità:", punti)

punti *= 2  # Raddoppio! Jackpot!
print("Dopo moltiplicatore:", punti)
```

## Esercizi

### Esercizio 1: Conversione temperatura

Scrivi un programma che converta una temperatura da Celsius a Fahrenheit. Formula: `F = C * 1.8 + 32`

```
# Completa il programma!  
celsius = float(input("Temperatura in Celsius: "))  
  
# Scrivi la formula qui sotto  
  
  
# Stampa il risultato
```

## Esercizio 2: Calcolo sconto

Un prodotto costa 80 euro e ha uno sconto del 15%. Calcola il prezzo scontato e mostra quanto risparmi!

```
prezzo = 80  
sconto_percentuale = 15  
  
# Calcola lo sconto e il prezzo finale  
  
  
# Stampa il risultato
```