

Variabili

Nomenclatura	snake_case che inizia per _ o lettera
Assegnazione singola	nome_variabile = valore
Assegnazione multipla	var1, var2, var3 = val1, val2, val3 var1 = var2 = var3 = valore

Sintassi di base

Commento

```
# Il tuo commento qui
```

Stampare a schermo

```
print()
```

Indentazione

Le righe aventi stessa indentazione appartengono allo stesso blocco di codice.

Tipi di dati

Numerici	int, float, complex, bool	
Stringhe	str (immutabile)	
Sequenze	list (ordinata, mutabile, permette duplicati)	[]
	tuple (ordinata, immutabile, permette duplicati)	()
	range	
Insiemi	set (non ordinato, mutabile, non permette duplicati)	{ }
	frozenset (non ordinato, immutabile, non permette duplicati)	
Dizionari	dict (chiave-valore)	
Binari	bytes, bytearray, memoryview	

Valore di verità

True

Un oggetto è considerato vero a meno che la sua classe non definisca un metodo `__bool__()` che restituisce False o un metodo `__len__()` che restituisce zero.

False

- > Costanti None e False
- > Zero di qualsiasi tipo numerico (0, 0.0 ecc.)
- > Sequenze o collezioni vuote ("", (), [] ecc.)



Operatori di confronto

<	Strettamente minore	<code>1 < 2</code>	<code># True</code>
<=	Minore o uguale	<code>1 <= 1</code>	<code># True</code>
>	Strettamente maggiore	<code>4 > 2</code>	<code># False</code>
>=	Maggiore o uguale	<code>3 >= 2</code>	<code># True</code>
==	Uguale	<code>5 == 7</code>	<code># False</code>
!=	Diverso	<code>1 != 2</code>	<code># True</code>
is	Identità dell'oggetto	<code>a = [1,2]</code> <code>b = a</code> <code>print(a is b) # True</code>	
is not	Identità dell'oggetto negata	<code>a = [1,2]</code> <code>b = a</code> <code>print(a is not b) # False</code>	

Possono essere concatenati arbitrariamente:

`X < Y <= Z` equivale a `X < Y AND Y <= Z`

Operatori numerici

<code>x + y</code>	Addizione	<code>3 + 4 = 7</code>
<code>x - y</code>	Sottrazione	<code>5 - 2 = 3</code>
<code>x * y</code>	Prodotto	<code>7 * 1 = 7</code>
<code>x / y</code>	Divisione	<code>8 / 4 = 2</code>
<code>x // y</code>	Divisione arrotondata per difetto	<code>3 // 2 = 1</code> <code>-1 // 2 = -1</code>
<code>x % y</code>	Modulo (ritorna il resto della divisione)	<code>7 % 3 = 1</code>
<code>-x</code>	Negazione	<code>-5</code>
<code>abs(x)</code>	Valore assoluto	<code>abs(-5.5) = 5.5</code>
<code>int(x)</code>	Conversione in int	<code>int("12 ") = 12</code> <code>int(7.2) = 7</code>
<code>float(x)</code>	Conversione in float	<code>float(7) = 7.0</code>
<code>complex(re, im)</code>	Conversione in numero complesso	<code>complex("3 + 5j")</code>
<code>divmod(x, y)</code>	Ritorna una tupla contenente quoziente intero e modulo dei parametri	<code>divmod(5, 2) = (2, 1)</code>



By GtM
cheatography.com/gtm/

Not published yet.
Last updated 10th August, 2025.
Page 2 of 11.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Operatori numerici (cont)

pow(x, y)	Potenza	<code>pow(2, 3) = 8</code>
x ** y		<code>2 ** 3 = 8</code>

Quando gli operandi sono di tipi numerici differenti, il tipo più piccolo viene convertito in quello più grande (int < float < complex).

Operatori booleani

x or y	True se almeno uno tra x e y è True, altrimenti False.	Non valuta il secondo argomento se il primo è True.
x and y	True se entrambi x e y sono True, altrimenti False.	Non valuta il secondo argomento se il primo è False.
not x	True se x è False, altrimenti False.	Ha priorità inferiore rispetto agli operatori non booleani *
* not a == b -> not (a == b)		

Tipi Numerici

int	Numeri interi a precisione illimitata. Costruttore -> <code>int()</code>
float	Numeri in virgola mobile, si utilizza il punto tra parte intera e parte decimale. Costruttore -> <code>float()</code>
complex	Numeri complessi, composti da una parte reale (n.real) ed una parte immaginaria (n.imag). Costruttore -> <code>complex()</code>
bool	Valori di verità (True e False).

Stringhe

Assegnazione	<pre>stringa = "Hello" stringa = 'Hello'</pre>	
Stringa multiriga	<pre>"""Hello, it's me"""</pre>	
Accedere ai caratteri	<pre>stringa[0]</pre>	# primo carattere
	<pre>stringa[-1]</pre>	# ultimo carattere
Lunghezza	<pre>len("He llo ")</pre>	# 5



By GtM
cheatography.com/gtm/

Not published yet.
Last updated 10th August, 2025.
Page 3 of 11.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Stringhe (cont)

Sottostringa	<code>stringa[:n]</code>	# da inizio a <code>_n</code>
	<code>stringa[n:]</code>	# da <code>°n</code> a fine
	<code>stringa[n:m]</code>	# da <code>°n</code> a <code>_m</code>
Concatenazione	<code>" ab" + " cd"</code>	# <code>" abc d"</code>
Ripetizione	<code>" " Hel lo"\ * 2</code>	# <code>"He llo Hel lo"</code>
Formattazione	<code>f"Ho {10} mele"</code>	# <code>"Ho 10 mele"</code>
Caratteri di escape	<code>\n</code>	# new line
	<code>\t</code>	# tab
	<code>\'</code>	# <code>'</code>

`°x` -> incluso / `_x` -> escluso

Metodi delle stringhe

<code>capitalize()</code>	<code>" heL lo".c ap ita lize()</code>	# <code>" Hel lo"</code>
<code>casefold()</code>	<code>" HeL lo".c as efold()</code>	# <code>" hel lo"</code>
<code>center()</code>	<code>" Hel lo".c en ter (7, " -")</code>	# <code>" -He llo -"</code>
<code>count()</code>	<code>" Hel lo".c ou nt(" l")</code>	# 2
<code>endswith()</code>	<code>"Hello".endswith("lo")</code>	# True
	<code>"Hello".endswith("ab")</code>	# False
<code>expandtabs()</code>	<code>" 1\t 2".e xpa ndt abs(1)</code>	# <code>"1 2"</code>
<code>find()</code>	<code>" Hel lo".f in d("l ")</code>	# 2
<code>format()</code>	<code>" Hel {}".f or mat ("lo ")</code>	# <code>" Hel lo"</code>
<code>format_map()</code>	come <i>format()</i>	Accetta dizionari come parametri
<code>index()</code>	come <i>find()</i>	ValueError se non trova la sottostringa
<code>isalnum()</code>	<code>"ABC123".isalnum()</code>	# True
	<code>"ab!12_".isalnum()</code>	# False
<code>isalpha()</code>	<code>"Abc".isalpha()</code>	# True
	<code>"123".isalpha()</code>	# False
<code>isascii()</code>	<code>"Abc123".isascii()</code>	# True
	<code>"Ab ⚡ 23".i sa scii()</code>	# False
<code>isidentifier()</code>	<code>"hello".isidentifier()</code>	# True
	<code>"123ab".isidentifier()</code>	# False



By GtM
cheatography.com/gtm/

Not published yet.
Last updated 10th August, 2025.
Page 4 of 11.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Metodi delle stringhe (cont)

islower()	"hello1".islower() "Hello1".islower()	# True # False
isprintable()	" Abc 1!".i sp rin tab le(){n l}} " A\t 1!".i sp rin ta ble()	# True # False
isspace()	"\t\n ".isspace() "a b".i ssp ace()	# True # False
join()	" _".j oin (["a " ,"b"])	# " a_b "
ljust()	" Abc ".l j ust (5, " -")	# " Abc --"
lower()	" HEL LO".l ower()	# " hel lo"
lstrip()	" -He llo -".l str ip(" -")	# " Hel lo- "
maketrans()	map1 = str.maketrans({"a":1,"b":2})	
translate()	map2 = str.maketrans("ab","12") map3 = str.maketrans("ab","12","c") "abc".translate(map1) "abc".translate(map2) "abc".translate(map3)	# "12c" # "12c" # " 12"
partition()	" a-b -c".p ar tit ion()	# ("a", " -", "b -c")
replace()	" abc ".re pla ce(" b","0 ")	# " a0c "
rfind()	" Hel lo".r fi nd(" l")	# 3
rindex()	come <i>rfind()</i>	ValueError se non trova la sottostringa
rjust()	" Abc ".r j ust (5, " -")	# " --A bc"
rpartition()	" a-b -c".r pa rti tion()	# ("a- b", "- ", "c")
rsplit()	" a-b -c".r sp lit ("-",1)	# ["a- b", "c "]
rstrip()	" -He llo -".r str ip(" -")	# " -He llo "
split()	" a-b -c".s pl it(" -",1)	# ["a", " b-c "]
splitlines()	" a\n b".s pli tli nes()	# ["a", " b"]
startswith()	"abc".startswith("a") "abc".startswith("c")	# True # False
strip()	" -He llo -".s tri p("- ")	# " Hel lo"
swapcase()	" XxX ".sw apc ase()	# " xXx "



Metodi delle stringhe (cont)

title()	" HELLO world".t itle()	# " Hello World"
upper()	" hel lo".u pper()	# " HEL LO"
zfill()	"1".zfill(3)	# "001"
	"-1".zfill(3)	# " -00 1"
isdigit()	# Numeri standard "123"	
isnumeric()	isdigit()	# True
isdecimal()	isnumeric()	# True
	isdecimal()	# True
	# Esponenti "123"	
	isdigit()	# True
	isnumeric()	# True
	isdecimal()	# False
	# Ideogrammi numerici	
	isdigit()	# False
	isnumeric()	# True
	isdecimal()	# False
	# Numeri decimali "12.34"	
	isdigit()	# False
	isnumeric()	# False
	isdecimal()	# False
	# Numeri negativi "-123"	
	isdigit()	# False
	isnumeric()	# False
	isdecimal()	# False
	# Stringa vuota ""	
	isdigit()	# False
	isnumeric()	# False
	isdecimal()	# False

* più aggressivo rispetto a lower()



By **GtM**
cheatography.com/gtm/

Not published yet.
 Last updated 10th August, 2025.
 Page 6 of 11.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Liste

Assegnazione	lista = ["a", 2, True, [1, 2], ...]	
Accedere agli elementi	lista[0]	# primo elemento
	lista[-1]	# ultimo elemento
Lunghezza	len(["a ", "b"])	# 2
Concatenazione	[1, 2] + [3, 4]	# [1, 2, 3, 4]
Ripetizione	[1, 2] * 2	# [1, 2, 1, 2]
Sottolista	lista[:n]	# da inizio a _n
	lista[n:]	# da °n a fine
	lista[n:m]	# da °n a _m

°x -> incluso / _x -> escluso

Metodi delle liste

append()	[1, 2].app end(3)	# [1, 2, 3]
clear()	[1, 2].clear()	# []
copy()	l1 = [1, 2]	
	l2 = l1.copy()	# l1 e l2 sono due liste distinte
	l3 = l1	# l1 e l3 puntano alla stessa lista
count()	[1, 2, 1].cou nt(1)	# 2
extend()	[1, 2].ext end (["a ", " b"])	# [1, 2, " a", " b"]
index()	[1, 2, 1].ind ex(1)	# 0
insert()	[1, 2].ins ert (1,3)	# [1, 3, 2]
pop()	[1, 2].pop(0)	# [2] , ritorna 1
remove()	[1, 2, 1].rem ove(1)	# [2, 1]
reverse()	[1, 2, 3].rev erse()	# [3, 2, 1]
sort()	[2, 1, 3].sort()	# [1, 2, 3]
	[2, 1, 3].sor t(r eve rse =True)	# [3, 2, 1]

Tuple

Assegnazione	tupla = ("a", 2, True, ...)	
	stringa = (1,)	# singolo elemento
Accedere agli elementi	tupla[0]	# primo elemento
	tupla[-1]	# ultimo elemento
Lunghezza	len((1,2))	# 2



By GtM
cheatography.com/gtm/

Not published yet.
 Last updated 10th August, 2025.
 Page 7 of 11.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Tuple (cont)

Sottotupla	<code>tupla[:n]</code>	# da inizio a <code>_n</code>
	<code>tupla[n:]</code>	# da <code>°n</code> a fine
	<code>tupla[n:m]</code>	# da <code>°n</code> a <code>_m</code>
Concatenazione	<code>(1, 2) + (3, 4)</code>	# <code>(1, 2, 3, 4)</code>
Ripetizione	<code>(1, 2) * 2</code>	# <code>(1, 2, 1, 2)</code>

`°x` -> incluso / `_x` -> escluso

Metodi delle tuple

count()	<code>(1, 2, 1).count(1)</code>	# 2
index()	<code>(1, 2, 3).index(3)</code>	# 2

Range

range(start, stop, step)

<i>start</i>	Opzionale	Numero iniziale (compreso, default = 0)
<i>stop</i>	Obbligatorio	Numero finale (escluso)
<i>step</i>	Opzionale	Passo dell'incremento (default = 1)

IF

Sintassi	<pre> if condiz_ione1: blocco_1 elif condiz_ione2: blocco_2 ... else: blocco_else </pre>
Singola istruzione	<code>if condizione: istruzione</code>
Operatore ternario	<code>se_true if condizione else se_false</code>
pass*	<pre> if condizione: pass </pre>

* i blocchi non possono essere vuoti



MATCH

Sintassi

```
match espressione:
    case valore1:
        blocco_1
    case valore2:
        blocco_2
    ...
    case _:
        blocco_de_fault
```

Combinazione di valori

```
case valore1 | valore2 | ... :
```

Case ... if

```
match len(stringa):
    case 4 if stringa[0] == "a":
        blocco_1
    ...
```

Case e sequenze

```
match sequenza:
    case [x, y]:
        print( f"Lista con due elementi: {x}, {y}")
    case [x, y, z]:
        print( f"Lista con tre elementi: {x}, {y}, {z}")
    case _:
        print( " Formato sconosciuto")
```

Case e dizionari

```
match dizionario:
    case {"nome": nome, "eta ": eta}:
        print( f"Nome: {nome}, Età: {eta}")
    case {"nome": nome}:
        print( f"Nome: {nome}")
    case _:
        print( " Formato sconosciuto")
```

WHILE

Sintassi

```
while condizione:
    blocco_codice
else:
    blocco_else
```

break

Interrompe il loop



By GtM
cheatography.com/gtm/

Not published yet.
Last updated 10th August, 2025.
Page 9 of 11.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

WHILE (cont)

continue	Salta l'iterazione corrente del loop
-----------------	--------------------------------------

FOR

Sintassi	<pre>for elemento in sequenza: blocco_codice else: blocco_else</pre>
-----------------	---

For each	<pre>for carattere in stringa: print(carattere) else: print(" Ciclo termin ato ")</pre>
-----------------	---

Range	<pre>for i in range(5): print(stringa[i]) else: print(" Ciclo termin ato ")</pre>
--------------	---

break	Interrompe il loop
--------------	--------------------

continue	Salta l'iterazione corrente del loop
-----------------	--------------------------------------

pass*	<pre>for elemento in sequenza: pass</pre>
--------------	---

* i blocchi non possono essere vuoti

Funzioni

Sintassi	def
-----------------	-----

Funzioni

Sintassi	<pre>def nome_funzione(parametri): blocco_codice</pre>	return opzionale
-----------------	--	------------------

Args (n parametri)	<pre>def contaStringhe(*stringhe): return len(stringhe) contaStringhe("a") contaStringhe("a", " b")</pre>	<pre># 1 # 2</pre>
---------------------------	--	--------------------



By GtM
cheatography.com/gtm/

Not published yet.
Last updated 10th August, 2025.
Page 10 of 11.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Funzioni (cont)

Keyword	def ordina Ele men ti(primo, secondo): print(f"Or dine: {primo} e {secondo}") ordinaElementi(1, 2) ordinaElementi(2, 1) ordinaElementi(secondo = 2, primo = 1)	# " Ordine: 1 e 2" # " Ordine: 2 e 1" # " Ordine: 1 e 2"
----------------	--	--

Kwargs (n chiave-valore)	def concatenaCV(**kwargs): concat Chiavi = "" for arg in kwargs: concat Chiavi += arg concat Valori = "" for val in kwargs.values(): concat Valori += val print(f"{c onc atC hiavi} e {concatValori}") concatenaCV(a = 1, b = 2, c = 3)	# "abc e 123"
---------------------------------	--	---------------

Default	def stampa Naz ion e(n azione = "Italia"): print(nazione) stampaNazione() stampaNazione("Germania")	# "Italia" # " Ger man ia"
----------------	--	-------------------------------

pass*	def funzione(): pass	
--------------	-------------------------	--

Solo parametri posizionali	def funzio ne(par ametri, /): blocco _codice	
-----------------------------------	--	--

Solo parametri con chiavi	def funzione(*, parametri): blocco _codice	
----------------------------------	---	--

Parametri misti	def funzio ne(pos1, pos2, /, *, chiave1, chiave2): blocco _codice	
------------------------	---	--

* i blocchi non possono essere vuoti

