

## OPERAZIONI COMUNI

**print(x, x, x, ..., sep=' ', end='\n')**

*sep* è il carattere separatore tra i valori (default spazio), *end* il carattere finale (default a capo).

**input(s):** restituisce una stringa con le info inserite da tastiera (senza '\n'). *s* è il messaggio iniziale.

**range(i, j, k):** crea una sequenza di interi che parte da *i* (compreso, default 0), arriva fino a *j* (escluso, obbligatorio), con passo *k* (default 1).

### PER TUTTI I CONTENITORI:

**len(cont):** restituisce il numero di elementi.

**x in cont:** restituisce True se l'elemento *x* è presente in *cont*, False altrimenti.

**sum(cont):** restituisce la somma dei valori degli elementi.

**max(cont) / min(cont):** restituisce l'elemento maggiore/minore.

**cont.clear():** elimina tutti gli elementi.

**sorted(cont):** restituisce una nuova lista contenente gli elementi di *cont* ordinati. Per le opzioni avanzate vedi `list.sort()`.

### PER TUTTE LE SEQUENZE:

**seq.count(x):** restituisce quante volte *x* è presente in *seq*.

**seq[i]:** restituisce l'*i*-esimo elemento ( $0 \leq i < \text{len}(\text{seq})$ ), altrimenti *IndexError*. Se *i* < 0, parte dal fondo.

**seq[i:j]:** restituisce una sottosequenza con gli elementi consecutivi di *seq*, dalla posizione *i* (compresa, default=0) fino alla posizione *j* (esclusa, default=len(seq)).

**seq[i:j:k]:** usa *k* come "passo" per selezionare gli elementi. Se *k* < 0 e *i* > *j* va all'indietro.

## STRINGHE

**int(s):** converte *s* in intero. Eccezione: *ValueError*.

**float(s):** converte *s* in float. Eccezione: *ValueError*.

**str(x):** converte *x* in stringa.

**ord(s) :** restituisce codice Unicode (intero) di *s*[0].

**chr(i):** restituisce carattere corrispondente a codice Unicode *i*. Eccezione: *ValueError*.

**s+s1:** crea e restituisce una nuova stringa concatenando due stringhe.

**s.lower() / s.upper():** restituisce la versione minuscola/maiuscola di *s*.

**s.replace(s1, s2) / s.replace(s1, s2, n):** restituisce una nuova versione di *s* in cui ogni occorrenza di *s1* è sostituita da *s2*. Se è presente *n*, sostituisce al massimo *n* occorrenze.

**s.lstrip() / s.lstrip(s1):** restituisce una nuova versione di *s* in cui i caratteri di spaziatura (spazi, tab, newline) sono eliminati dall'inizio di *s*. Se è presente *s1*, vengono eliminati i caratteri presenti in essa invece dei caratteri di spaziatura.

**s.rstrip() / s.rstrip(s1):** Come *lstrip*, ma i caratteri vengono eliminati dalla fine di *s*.

**s.strip() / s.strip(s1):** Come *lstrip*, ma i caratteri vengono eliminati tanto all'inizio quanto alla fine.

**s1 in s:** restituisce True se *s* contiene *s1* come sottostringa, altrimenti False.

**s.count(s1):** restituisce il numero di occorrenze non sovrapposte di *s1* in *s*.

**s.startswith(s1) / s.endswith(s1):** restituisce True se *s* inizia/termina con *s1*, altrimenti False.

**s.find(s1) / s.find(s1, i, j):** restituisce il primo indice di *s* in cui inizia un'occorrenza di *s1*, oppure -1 se non c'è. Se presenti *i* e *j*, ricerca in *s[i:j]*.

**s.index(s1):** come *find*, ma se non presente solleva *ValueError*.

**s.isalnum():** restituisce True se *s* contiene sole lettere o cifre e ha almeno un carattere, altrimenti False.

**s.isalpha():** restituisce True se *s* contiene sole lettere e ha almeno un carattere, altrimenti False.

**s.isdigit():** restituisce True se *s* contiene sole cifre e ha almeno un carattere, altrimenti False.

**s.islower() / s.isupper():** restituisce True se *s* contiene sole lettere minuscole/maiuscole e ha almeno un carattere, altrimenti False.

**s.isspace():** restituisce True se *s* contiene soli caratteri di spaziatura (spazi, tab e newline) e ha almeno un carattere, altrimenti False.

### DA STRINGHE A LISTE E VICEVERSA:

**s.split(sep, maxsplit=n):** restituisce una lista di sotto-stringhe ottenute suddividendo *s* ad ogni occorrenza della stringa *sep* (separatore). Se *sep* è omissso, per default è una sequenza di caratteri di spaziatura. Se *maxsplit* è specificato, saranno fatte al massimo *n* separazioni partendo da sinistra (la lista avrà al più *n*+1 elementi).

**s.rsplit(sep, maxsplit=n):** come *split*, ma suddivide *s* partendo da destra.

**s.splitlines():** come *split*, ma usa come separatore il '\n', suddivide quindi *s* in una lista contenente le singole righe di testo presenti in *s*.

**s.join(l):** restituisce una unica stringa contenente tutti gli elementi di *l* separati dal separatore *s*.

## MATEMATICA

**abs(a), round(a), round(a, n)**

import math ↘

**math.sin(a), cos(a), tan(a), exp(a), log(a), sqrt(a).**

Possono sollevare *ValueError*

**math.isclose(a, b, rel\_tol, abs\_tol):** restituisce True se  $|a - b|$  è minore o uguale di *rel\_tol* (tolleranza relativa) o *abs\_tol* (tolleranza assoluta).

import random ↘

**random.random():** restituisce un numero casuale float nell'intervallo [0,1).

**random.randint(i, j):** restituisce un numero intero casuale tra *i* e *j* (estremi compresi).

**random.choice(seq):** restituisce un elemento qualsiasi della sequenza *seq*.

**random.shuffle(seq):** rimescola in ordine casuale gli elementi della sequenza *seq*.

## LISTE

**[]:** crea e restituisce una nuova lista vuota

**[x, ..., x]:** restituisce una nuova lista con gli elementi forniti.

**list(cont):** restituisce una nuova lista contenente tutti gli elementi del contenitore *cont*.

**l \* n:** restituisce una nuova lista replicando gli elementi di *l* per *n* volte.

**l + l1:** restituisce una nuova lista concatenando gli elementi di *l* ed *l1*.

**l == l1:** restituisce True se le due liste contengono gli stessi elementi, nello stesso ordine, altrimenti False.

**l.pop():** rimuove l'ultimo elemento e lo restituisce.

**l.pop(i):** rimuove l'elemento nella posizione *i* e lo restituisce. Gli elementi seguenti sono spostati indietro di un posto.

**l.insert(i, x):** inserisce *x* nella posizione *i* in *l*. Gli elementi da quella posizione in poi sono spostati avanti di un posto.

**l.append(x):** aggiunge *x* in coda alla lista *l*.

**l.index(x):** restituisce la posizione della prima occorrenza di *x* in *l*. L'elemento deve essere presente in lista, altrimenti solleva *ValueError*.

**l.remove(x):** rimuove l'elemento di valore *x* dalla lista e sposta indietro di un posto tutti gli elementi che lo seguono. L'elemento deve essere presente in lista, altrimenti solleva *ValueError*.

**l.extend(l1):** aggiunge tutti gli elementi della lista *l1* alla lista *l*.

**l.reverse():** rovescia l'ordine degli elementi nella lista *l*.

**l.copy() o list(l):** restituisce una nuova lista copia della lista *l*.

**l.sort(reverse=False):** ordina gli elementi della lista dal più piccolo al più grande. Se si specifica *reverse=True*, ordina in ordine inverso.

from operator import itemgetter ↘

**l.sort(key=itemgetter('k')):** ordina una lista di dizionari in base al valore del campo con chiave *k*.

**l.sort(key=itemgetter(n)):** ordina una lista di liste o di tuple in base al valore dell'elemento di indice *n*.

Nota: *reverse* e *key* si possono combinare.

## FILE

**f = open(s, modalità):** apre il file di nome *s*. *modalità*: "r" lettura, "w" scrittura. Restituisce un "oggetto file" *f*. Eccezioni: *FileNotFoundError* se il file non esiste, in generale *IOError*.

**f.close():** chiude il file *f*.

**f.readline():** restituisce una stringa con i caratteri letti dal file *f* fino a '\n' (compreso). Restituisce "" se a fine file.

**f.read(num):** restituisce una stringa con (al massimo) *num* caratteri letti dal file *f*. Senza argomenti legge l'intero file.

**f.readlines():** restituisce il contenuto dell'intero file sotto forma di lista di stringhe, una per riga.

**f.write(s):** scrive *s* nel file *f*. Nota: non aggiunge automaticamente il fine linea '\n'.

**print(...., file=f):** come print, ma scrive nel file *f* anziché su schermo.

## INSIEMI

**set():** restituisce un nuovo insieme vuoto.

**set(cont):** restituisce un nuovo insieme che contiene una copia di *cont* (senza duplicati).

**{x,x, ..., x}:** restituisce un nuovo insieme che contiene gli elementi indicati (senza duplicati).

**t.add(x):** aggiunge un nuovo elemento all'insieme *t*. Se l'elemento è già presente, non succede nulla.

**t.discard(x):** elimina l'elemento dall'insieme *t*. Se l'elemento non appartiene all'insieme, non ha effetto.

**t.remove(x):** come *discard*, ma se l'elemento non è presente solleva *KeyError*.

**t == t1:** determina se l'insieme *t* è uguale all'insieme *t1*.

**t.issubset(t1):** determina se  $t \subseteq t1$ .

**t.issuperset(t1):** determina se  $t \supseteq t1$ .

**t.isdisjoint(t1):** restituisce True se l'intersezione degli insiemi *t* e *t1* è nulla.

**t.union(t1):** restituisce un nuovo insieme pari a  $t \cup t1$ .

**t.intersection(t1):** restituisce un nuovo insieme pari a  $t \cap t1$ .

**t.difference(t1):** restituisce un nuovo insieme che contiene gli elementi che appartengono a *t* ma non a *t1*.

**t.symmetric\_difference(t1):** restituisce un nuovo insieme che contiene gli elementi presenti in uno solo degli insiemi e non in entrambi.

**t.copy()** o **set(t):** restituisce una copia dell'insieme *t*.

## DIZIONARI (k = chiave: stringa, numero, tupla)

**dict():** restituisce un nuovo dizionario vuoto.

**{}**: restituisce un nuovo dizionario vuoto.

**{k:x, ... , k:x}:** restituisce un nuovo dizionario contenente le coppie chiave/valore specificate.

**k in d:** restituisce True se la chiave *k* appartiene al dizionario *d*, altrimenti False.

**d[k] = x:** aggiunge una nuova coppia chiave/valore al dizionario *d*, se *k* non è già presente, altrimenti modifica il valore associato alla chiave *k*.

**d[k]:** restituisce il valore associato alla chiave *k*, se è presente in *d*, altrimenti solleva *KeyError*.

**d.get(k, x):** restituisce il valore associato alla chiave *k*, se è presente in *d*, altrimenti restituisce il valore di default *x*.

**d.pop(k):** elimina da *d* la chiave *k* e il valore ad essa associato; se non è presente, solleva *KeyError*. Restituisce il valore eliminato.

**d.items():** restituisce una lista di tuple (*k,x*) di tutti gli elementi di *d*.

**d.values():** restituisce una lista contenente tutti i valori presenti in *d*.

**d.keys():** restituisce una lista con le chiavi del dizionario.

**sorted(d):** restituisce una lista ordinata delle chiavi del dizionario.

**sorted(d.items()):** restituisce una lista, ordinata per chiave, di tuple (*k,x*) di tutti gli elementi di *d*.

**d.copy()** o **dict(d):** restituisce una copia del dizionario.

## LEGENDA (tipi degli argomenti/oggetti accettati)

**s, s1:** stringa                      **a,b,c, ...:** intero o float

**i, j, k, n:** intero                      **x:** qualsiasi

**l, l1:** lista;      **d:** dizionario;      **t, t1:** set

**seq:** sequenza (lista, tupla, stringa)

**cont:** contenitore (lista, tupla, stringa, set, dict)