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 ($0 \le i < \text{len}(seq)$, 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 a 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 omesso, 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*. m*odalità:* "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)

COMMON OPERATIONS:

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

sep is the separator character between the values (default is space), *end* the final character (default is new line).

input(s): It returns a string with the information entered from the keyboard (without '\n'). *s* is the initial message.

range(i, j, k): It creates a sequence of integers that starts from *i* (inclusive, default 0), goes up to *j* (excluded, mandatory), with step *k* (default 1).

FOR ALL THE CONTAINERS:

len(cont): It returns the number of elements.

x in cont: It returns True if the element *x* is included in *cont*, False otherwise.

sum(cont): It returns the sum of the values of the elements.

max(cont) / min(cont): It returns the maximum/minimum element.

cont.clear(): It deletes all the elements. **sorted(cont)**: It returns a new list containing the elements of *cont* ordered. For advanced options, see list.sort().

FOR ALL THE SEQUENCES:

seq.count(x): It returns how many times *x* is contained in *seq*.

seq[i]: It returns the element *i* (0≤*i*<len(*seq*), otherwise *IndexError*). If *i*<0, it starts from the bottom.

seq[i:j]: It returns a subsequence with consecutive elements of *seq*, from position *i* (inclusive, default = 0) to position *j* (excluded, default = len (seq)).

seq[i:j:k]: It uses *k* as "step" to select elements. If *k <0* and *i*> *j* it goes backwards.

STRINGS

int(s): It converts *s* as integer. Exception: *ValueError*.

float(s): It converts *s* as float. Exception: *ValueError*.

str(x): It converts *x* as string.

ord(s) : It returns the Unicode code (integer) of *s*[0].

chr(i): It returns the corresponding Unicode character *i*. Exception: *ValueError*.

s+s1: It creates and returns a new string concatenating two strings.

s.lower() / s.upper(): It returns the lowercase / uppercase version of *s*.

s.replace(s1, s2) / s.replace(s1, s2, n): It returns a new version of *s* in which every occurrence of *s1* is replaced by *s2*. If *n* is present, it replaces at most *n* occurrences.

s.lstrip() / s.lstrip(s1): It returns a new version of *s* in which whitespace characters (spaces, tabs, newlines) are deleted from the beginning of *s*. If *s1* is present, the characters present in it are deleted instead of the whitespace characters.

s.rstrip() / s.rstrip(s1): As *lstrip*, but the characters are stripped from the end of s.

s.strip() / s.strip(s1): As *lstrip*, but the characters are eliminated both at the beginning and at the end.

s1 in s: It returns *True* if *s* contains *s1* as a substring, otherwise *False*.

s.count(s1): It returns the number of non-overlapping occurrences of *s1* in *s.*

s.startswith(s1) / s.endswith(s1): It returns *True* if s starts / ends with *s1*, otherwise *False*.

s.find(s1) / s.find(s1, i, j): It returns the first index of *s* in which an occurrence of *s1* begins, or -1 if there is none. If *i* and *j* are present, it searches in *s [i: j].*

s.index(s1): as *find*, but if it is not present it raises *ValueError*.

s.isalnum(): It returns *True* if s contains only letters or digits and has at least one character, otherwise *False*.

s.isalpha(): It returns *True* if s contains only letters and has at least one character, otherwise *False*.

s.isdigit(): It returns *True* if s contains only digits and has at least one character, otherwise *False*.

s.islower() / s.isupper(): It returns *True* if s contains only lowercase / uppercase letters and has at least one character, otherwise *False*.

s.isspace(): It returns *True* if s contains only whitespace characters (spaces, tabs, and newlines) and has at least one character, otherwise *False*.

FROM STRINGS TO LISTS AND VICE VERSA:

s.split(sep, maxsplit=n): It returns a list of sub-strings obtained by subdividing *s* at each occurrence of the string sep (separator). If *sep* is omitted, it is a sequence of whitespace characters by default. If maxsplit is specified, a maximum of n separations will be made starting from the left (the list will have at most *n + 1* elements).

s.rsplit(sep, maxsplit=n): as *split*, but it subdivides *s* starting from the right.

s.splitlines(): as *split*, but it uses as separator the '\n', then divides *s* into a list containing the single lines of text present in *s*.

s.join(l): It returns a single string containing all the elements of *l* separated by the separator *s.*

MATHEMATICS

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

import math ↘

math.sin(a), **cos(a)**, **tan(a)**, **exp(a)**, **log(a)**, **sqrt(a)**. They can raise *ValueError*

math.isclose(a, b, rel_tol, abs_tol): It returns *True* if $|a - b|$ is lower or equal to *rel_tol* (relative tolerance) or *abs_tol* (absolute tolerance).

import random ↘

random.random(): It returns a random number float in the interval [0,1).

random.randint(i, j): It returns a random integer between *i* and *j* (inclusive).

random.choice(seq): It returns any element of the sequence *seq*.

random.shuffle(seq): It shuffles the elements of the sequence *seq* in random order.

LISTS

[]: It creates and returns a new empty list

[x , ..., x]: It returns a new list with the supplied elements.

list(cont): It returns a new list containing all the elements of the *cont* container.

l * n: It returns a new list by replicating the elements of *l n* times.

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

l == l1: It returns *True* if the two lists contain the same elements, in the same order, otherwise *False*.

l.pop(): It removes the last element and returns it.

l.pop(i): It removes the element in position *i* and returns it. The following items are moved back one place.

l.insert(i, x): It inserts *x* into position *i* in *l.* Elements from that position on are moved forward one place.

l.append(x): It adds x at the end of the list l.

l.index(x): It returns the position of the first occurrence of *x* in *l*. The element must be present in the list, otherwise it raises *ValueError*.

l.remove(x): It removes the element with value *x* from the list and moves all the subsequent elements back one place. The element must be present in the list, otherwise it raises *ValueError*.

l.extend(l1): It adds all elements of list *l1* to list *l*.

l.reverse(): It reverses the order of the elements in the list *l*.

l.copy() o list(l): It returns a new list copy of list *l*.

l.sort(reverse=False): It sorts the items in the list from smallest to largest. If you specify reverse = True, it sorts in reverse order.

from operator import itemgetter ↘

l.sort(key=itemgetter('k')): It sorts a list of dictionaries based on the value of the field with key *k*.

l.sort(key=itemgetter(n)): It sorts a list of lists or tuples based on the value of the index element *n*.

Note: reverse and key can be combined.

FILES

f = open(s, mode): it opens the file named *s*. Mode*:* "r" reading, "w" writing. It returns an "object file" *f*. Exception: *FileNotFoundError* if the file does not exist, in general *IOError*.

f.close(): it closes the file *f*.

f.readline(): It returns a string with characters read from file *f* up to and including '\ n'. It returns "" if at end of file.

f.read(num): It returns a string with (at most) *num* characters read from the file *f*. Without arguments, it reads the entire file.

f.readlines(): It returns the contents of the entire file as a list of strings, one per line.

f.write(s): It writes *s* to file *f*. Note: It does not automatically add the newline '\ n'.

print(...., file=f): as print, but it writes to file *f* instead of printing to screen.

SETS

set(): It returns a new empty set.

set(cont): It returns a new set containing a copy of *cont* (without duplicates).

{x ,x, ..., x}: It returns a new set containing the indicated elements (without duplicates).

t.add(x): It adds a new element to the set *t*. If the element is already present, nothing happens.

t.discard(x): It deletes the element from the set *t*. If the element does not belong to the whole, it has no effect.

***t*.remove(x):** as *discard*, but if the element is not present, it raises *KeyError*.

t == t1: It determines if the set *t* is equal to *t1*.

t.issubset(t1): It determines if $t \subseteq t1$.

t.issuperset(t1): It determines if $t \supseteq t1$.

t.isdisjoint(t1): It returns True if the intersection of the set *t* and *t1* is null.

t.union(t1): it returns a new set equal to $t \cup t1$.

t.intersection(t1): It returns a new set equal to $t \cap t1$.

t.difference(t1): It returns a new set that contains elements that belong to *t* but not *t1*.

t.symmetric_difference(t1): It returns a new set that contains elements in only one of the sets and not both.

t.copy() o **set(t):** It returns a copy of the set *t*.

DICTIONARY (*k* = key: string, number, tuple)

dict(): It returns a new empty dictionary.

{}: It returns a new empty dictionary.

{k:x, ... , k:x}: It returns a new dictionary containing the specified key/value pairs.

k in d: It returns *True* if the key *k* belongs to the dictionary *d*, otherwise *False*.

d[k] = x: It adds a new key / value pair to dictionary d, if *k* is not already present, otherwise it modifies the value associated with key *k*.

d[k]: It returns the value associated with key *k*, if it is present in *d*, otherwise it raises *KeyError*.

d.get(k, x): It returns the value associated with the key *k*, if it is present in *d*, otherwise it returns the default value *x*.

d.pop(k): It removes from *d* the key *k* and the value associated with it; if not present, it raises *KeyError*. It returns the deleted value.

d.items(): It returns a list of tuples (k, x) of all elements of *d*.

d.values(): It returns a list containing all the values of *d*.

d.keys(): It returns a list with all the dictionary keys.

sorted(d): It returns an ordered list of the dictionary keys.

sorted(d.items()): It returns a list, sorted by key, of tuples (k, x) of all elements of *d*.

d.copy() o **dict(d):** It returns a copy of the dictionary.

LEGEND (tipi degli argomenti/oggetti accettati)

s, s1: string **a,b,c, ...:** integer or float

i, j, k, n: integer **x:** any

l, l1: list; **d:** dictionary; **t, t1:** set

seq: sequence (list, tuple, string)

cont: container (list, tuple, string, set, dict)

vers 1.0 / by AS-FC-CF