

Stringhe: lunghezza

Ottenere la lunghezza di una stringa

versione: 2023-11-28

Stringhe: prefissi e suffissi

Verificare se una stringa è prefisso o suffisso di un'altra

versione: 2023-11-28

Stringhe: conteggio occorrenze

Contare quante occorrenze di una sottostringa ci sono in una stringa

versione: 2023-11-28

Stringhe: trovare la posizione

Trovare la posizione di una sottostringa in una stringa data

versione: 2023-11-28

Stringhe: maiuscolo/minuscolo

Convertire una stringa in maiuscolo, e viceversa

versione: 2023-11-28

Stringhe: spazi

Rimuovere gli spazi bianchi all'inizio e alla fine

versione: 2023-11-28

Stringhe: concatenazione

Data una lista di stringhe, unirle tutte per ottenerne una sola stringa, usando una stringa specifica come 'collante'

versione: 2023-11-28

Stringhe: trova e sostituisci

Sostituisci il pattern dato nella stringa con una nuova stringa

versione: 2023-11-28

Stringhe: slicing

Accedere alla sottostringa che va da `start` a `end`. `end` è inclusa o esclusa?

versione: 2023-11-28

Stringhe: conteggio occorrenze

```
s = "banana"  
s.count("na")
```

Stringhe: prefissi e suffissi

```
s = "prestigiatore"  
  
# prefisso  
s.startswith("pres")  
  
# suffisso  
s.endswith("ore")
```

Stringhe: lunghezza

```
len(string)
```

Stringhe: spazi

```
s = "    banana    "  
  
# rimuove spazi  
# a sinistra  
s.lstrip()  
  
# rimuove spazi  
# a destra  
s.rstrip()  
  
# rimuove spazi  
# a inizio/fine  
s.strip()
```

Stringhe: maiuscolo/minuscolo

```
s = "banana"  
  
# Maiuscolo -> "BANANA"  
s.upper()  
  
# Minuscolo -> "banana"  
s.lower()
```

Stringhe: trovare la posizione

```
s = "banana"  
  
# trova la prima  
# occorrenza  
s.find("na")  
  
# trova la seconda  
# occorrenza  
s.find("na", start=3)
```

Stringhe: slicing

```
s = "banana"  
start = 1  
end = 4  
  
s[start:end]
```

Stringhe: trova e sostituisci

```
s = "banana"  
pattern = "na"  
replacement = "NEWSTRING"  
  
s.replace(pattern,  
           replacement)  
# il risultato e'  
# "baNEWSTRINGna"
```

Stringhe: concatenazione

```
strings = [  
    "first",  
    "second",  
    "third"  
]  
  
# Produce una  
# sola stringa  
" | ".join(strings)
```

File: leggere un file

Aprire un file in lettura e leggerne tutto il contenuto

versione: 2023-11-28

File: scrivere un file (1)

Scrivere un file, cancellando il contenuto precedente

versione: 2023-11-28

File: scrivere un file (2)

Scrivere un file, aggiungendo al contenuto precedente

versione: 2023-11-28

Liste: ordinare (1)

Ordinare una lista in ordine crescente

versione: 2023-11-28

Liste: ordinare (2)

Ordinare una lista in ordine decrescente

versione: 2023-11-28

Liste: cercare (1)

Cercare un elemento che è in una lista

versione: 2023-11-28

Liste: cercare (2)

Cercare un elemento che non è in una lista. Cosa succede? Viene ritornato -1 o riceviamo un errore? Dobbiamo controllare prima?

versione: 2023-11-28

Liste: iterazione (1)

Iterare su tutti gli elementi di una lista, per indice, usando un ciclo for

versione: 2023-11-28

Liste: iterazione (2)

Iterare su tutti gli elementi di una lista, per indice, usando un ciclo while

versione: 2023-11-28

File: scrivere un file (2)

```
fname = "myfile.txt"
content = "Ciao"

fh = open(fname, "a")
print(content, file=fh)
fh.close()
```

File: scrivere un file (1)

```
fname = "myfile.txt"
content = "Ciao"

fh = open(fname, "w")
print(content, file=fh)
fh.close()
```

File: leggere un file

```
fname = "myfile.txt"

fh = open(fname, "r")
content = fh.read()
fh.close()
```

Liste: cercare (1)

```
my_list = [
    "dog",
    "cat",
    "tiger"
]

my_list.index("cat")
```

Liste: ordinare (2)

```
my_list = [
    3, 5, 1, 3
]

my_list.sort(
    reverse=True
)
```

Liste: ordinare (1)

```
my_list = [
    3, 5, 1, 3
]

my_list.sort()
```

Liste: iterazione (2)

```
my_list = [
    "dog",
    "cat",
    "tiger"
]

i = 0
while i < len(my_list):
    print(my_list[i])
    i += 1
```

Liste: iterazione (1)

```
my_list = [
    "dog",
    "cat",
    "tiger"
]

n = len(my_list)
for i in range(n):
    print(my_list[i])
```

Liste: cercare (2)

```
my_list = [
    "dog",
    "cat",
    "tiger"
]

target = "lion"
if target in my_list:
    my_list.index(target)
else:
    print("not found")
```

Liste: accesso (1)

Accedere all'ultimo elemento di una lista

versione: 2023-11-28

Liste: accesso (2)

Accedere all'elemento all'indice i

versione: 2023-11-28

Liste: concatenazione

Concatenare due liste assieme, in modo che tutti gli elementi di una vengano prima di tutti gli elementi dell'altra

versione: 2023-11-28

Liste: aggiungere

Aggiungere un elemento in fondo a una lista

versione: 2023-11-28

Liste: inserire

Aggiungere a una lista un elemento in una posizione i data

versione: 2023-11-28

Liste: rimozione (1)

Cancellare l'ultimo elemento di una lista

versione: 2023-11-28

Liste: rimozione (2)

Cancellare da una lista l'elemento in una posizione i data

versione: 2023-11-28

Liste: rimozione (3)

Cancellare da una lista un elemento in base al suo valore

versione: 2023-11-28

Liste: scambio elementi

Scambiare i due elementi in posizione i e j .

versione: 2023-11-28

Liste: concatenazione

```
list1 = [  
    "dog",  
    "cat",  
    "tiger"  
]  
list2 = [  
    "elephant",  
    "lion",  
    "giraffe"  
]  
  
list1 + list2
```

Liste: accesso (2)

```
my_list = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
my_list[i]
```

Liste: accesso (1)

```
my_list = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
my_list[-1]  
# oppure  
my_list[len(my_list) - 1]
```

Liste: rimozione (1)

```
animals = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
animals.pop()
```

Liste: inserire

```
animals = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
animals.insert(1, "lion")
```

Liste: aggiungere

```
animals = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
animals.append("lion")
```

Liste: scambio elementi

```
animals = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
tmp = animals[i]  
animals[i] = animals[j]  
animals[j] = tmp
```

Liste: rimozione (3)

```
animals = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
animals.remove("cat")
```

Liste: rimozione (2)

```
animals = [  
    "dog",  
    "cat",  
    "tiger"  
]  
  
animals.pop(i)
```

Insiemi

Creare un insieme vuoto

versione: 2023-11-28

Insiemi

Aggiungere un elemento a un insieme. Un insieme può contenere tipi di dato diversi?

versione: 2023-11-28

Insiemi

Eliminare un elemento da un insieme

versione: 2023-11-28

Insiemi

Quanti elementi ci sono in un insieme?

versione: 2023-11-28

Insiemi

Controllare se un insieme è sottoinsieme di un altro

versione: 2023-11-28

Insiemi

Unione di due insiemi

versione: 2023-11-28

Insiemi

Intersezione di due insiemi

versione: 2023-11-28

Insiemi

Differenza di due insiemi. È un'operazione commutativa?

versione: 2023-11-28

Insiemi

```
myset = set()
myset.add("Elemento")
myset.add(3)
```

Insiemi

```
emptyset = set()
```

Insiemi

```
s1 = {"a", "b", "c"}
s2 = {"a", "b"}

s2.issubset(s1)
```

Insiemi

```
myset = {"Pippo", "Pluto"}
len(myset)
```

Insiemi

```
myset = {"Pippo", "Pluto"}
myset.discard("Pippo")
```

Insiemi

```
s1 = {"a", "b", "c"}
s2 = {"a", "b"}

s1.difference(s2)
```

Insiemi

```
s1 = {"a", "b", "c"}
s2 = {"a", "b"}

s1.intersection(s2)
```

Insiemi

```
s1 = {"a", "b", "c"}
s2 = {"a", "b"}

s1.union(s2)
```


Insiemi

Creare una copia dell'insieme

versione: 2023-11-28

Insiemi

Controllare se un elemento è in un insieme

versione: 2023-11-28

Dizionari

Creare un dizionario vuoto

versione: 2023-11-28

Dizionari

Creare un dizionario con qualche coppia chiave/valore

versione: 2023-11-28

Dizionari

Ottenere il valore associato a una chiave

versione: 2023-11-28

Dizionari

Iterare sulle chiavi

versione: 2023-11-28

Dizionari

Iterare sui valori

versione: 2023-11-28

Dizionari

Iterare sulle coppie chiave/valore

versione: 2023-11-28

Dizionari

Controllare se il dizionario contiene una chiave

versione: 2023-11-28

Dizionari

```
emptydict = dict()
```

Insiemi

```
myset = {"Pippo", "Pluto"}  
  
if "Pippo" in myset:  
    print("Trovato!")
```

Insiemi

```
myset = {"Pippo", "Pluto"}  
  
copyset = set(myset)
```

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}  
  
for key in mydict.keys():  
    print(key)
```

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}  
  
mydict["name"]
```

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}
```

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}  
  
if "name" in mydict:  
    print("Il dizionario  
    contiene la chiave")
```

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}  
  
for (key, value) in  
    mydict.items():  
    print("key is", key,  
        "value is", value)
```

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}  
  
for value in mydict.  
    values():  
    print(value)
```

Dizionari

Aggiornare o inserire un valore associato a una chiave

versione: 2023-11-28

Dizionari

Cancellare una chiave e il suo valore. Cosa succede se la chiave non è presente?

versione: 2023-11-28

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}  
  
# Errore se la chiave non  
# presente  
mydict.pop("height")
```

Dizionari

```
mydict = {  
    "name": "Pippo",  
    "height": 190  
}  
  
mydict["favourite food"]  
    = "cake"
```