Liste (list)

Liste (list)

- Usato per rappresentare una **sequenza mutabile** di oggetti, in genere omogenei.
- Le liste vengono definite elencando tra **parentesi quadre.**
- A differenza di tuple e stringhe che sono immutabili, le liste possono **essere mutate**.

Liste (list)

```
lista_lettere = ['r', 'g', 'b']
lista_numeri = [6,8,3,2]
lista_numeri[1] = 9
print(lista_numeri) # [6,9,3,2]

lista_di_liste = [[7,14,21],[8,16,24]]
print(lista_di_liste[1][0]) # 8
```

Liste (list)

```
lista_numeri = [6,8,3,2]
for i in lista_numeri:
    print(i, end=" ")     # 6 8 3 2
```

Liste – funzioni specifiche

1. *append()*
2. *remove()*
3. *sort()*
4. *reverse()*
5. *count()*
6. *insert()*
7. *index()*
8. *pop()*

Liste – Alcune funzioni chiave

1. *len()*
2. *max()*
3. *min()*

Liste – insert()

```
lista = [5, 4, 3, 9, 7]
print(lista)     # [5, 4, 3, 9, 7]
# inserisce in posizione 0 il numero 6
lista.insert(0, 6)
print(lista)     # [6, 5, 4, 3, 9, 7]
```

Esercizio

Avendo la lista [4,2,6,5], visualizzare i vari valori in ordine inverso:

5 6 2 4

append()

append()

```
# Aggiunge il valore indicato
# come ultimo elemento
lista = [5,2,4]
lista.append(9)
print(lista) # 5,2,4,9
```

remove()

remove()

```
# Rimuove la prima occorrenza
# del valore indicato
lista = [5,4,2,4]
lista.remove(4)
print(lista) # 5,2,4
```

sort()

sort()

```
# Ordina i valori della lista
lista = [5,4,2,4]
lista.sort()
print(lista) # 2,4,4,5
```

reverse()

reverse()

```
# Inverte l'ordine della lista
lista = [5,4,2,4]
lista.reverse()
print(lista) # 4,2,4,5
```

count()

count()

```
# Conta quante volte un dato valore
# è presente all'interno della lista
lista = [5,4,2,4]
print(lista.count(4)) # 2
print(lista.count(5)) # 1
```

insert()

insert()

```
lista = [8,3,5,6]
lista.insert(2,11) # indice, valore
print(lista) # [8, 3, 11, 5, 6]
```

index()

index()

```
lista = [8,3,5,6]
print(lista.index(5)) # 2
print(lista.index(3)) # 1
```

pop()

pop()

```
lista = [8,3,5,6]
lista.pop(1)
print(lista) # [8, 5, 6]
```

len()

len()

```
lista = [8,3,5,6]
# len: abbreviazione di length
# restituisce la lunghezza della lista
print(len(lista)) # 4
```

max() e min()

max() e min()

```
lista = [8,3,5,6]
print(max(lista), min(lista)) # 8 3
```

Lista di liste

```
valori = [["x", "o", "o"],["o","x","o"],["o","o","x"]]
valori[1][2] = "w"
for i in valori:
    for j in i:
        print(j, end=" ")
    print()
```

Argomento extra.. *random*

```
import random
# Restituisce un valore compreso tra 0 e 1
print(random.random())
# Restituisce un valore compreso tra 5 e 10
print(random.randint(5,10))
```

Liste – esercizi

Controlla se una dato è presente in una lista:

```
lista = [55, 3, 89, 21]
num = 3
if num in lista:
    print(num, " è in ", lista)
else:
    print(num, " non in ", lista)
```

Liste

Itera su parte della lista

```
lista = [5, 4, 9, 6, 10, 7, 3]
for i in lista[3:]:
    print(i, end=" ") # 6 10 7 3
for i in lista[:4]:
    print(i, end=" ") # 5 4 9 6
for i in lista[2:4]:
    print(i, end=" ") # 9 6
```

Ordinamento Liste

Ordinamento di un vettore

- Esistono diversi algoritmi per ordinare gli elementi di un vettore. Tra questi troviamo:
- ***Selection*** *sort*
- ***Bubble*** *sort*
- ***Insertion*** *sort*

Selection sort

Selection sort

Dichiarare una lista con i valori:

[6, 8, 3, 9, 5, 7, 4]

Poi spostare il valore più basso mettendolo ad indice 0. Otterremo così il seguente risultato:

[3, 6, 8, 9, 5, 7, 4]

Selection sort

Usando 2 cicli for annidati, ottenere il seguente output:

0 1 2 3 4 5 6

1 2 3 4 5 6

2 3 4 5 6

3 4 5 6

4 5 6

5 6

6

Selection sort

Avendo il vettore [6, 8, 3, 9, 5, 7, 4], scrivere un programma che stampi in output:

6 8 3 9 5 7 4

8 3 9 5 7 4

3 9 5 7 4

9 5 7 4

5 7 4

7 4

4

Selection sort

Dichiarare una lista con i valori [6, 8, 3, 9, 5, 7, 4].

Trovare il valore più basso tra tutti i numeri della lista, poi solo tra gli ultimi 6 numeri, poi solo tra gli ultimi 5 numeri e via dicendo fino a considerare solo l'ultimo numero.

Scrivere in output i numeri trovati. Il risultato dovrebbe essere:

3 3 3 4 4 4 4

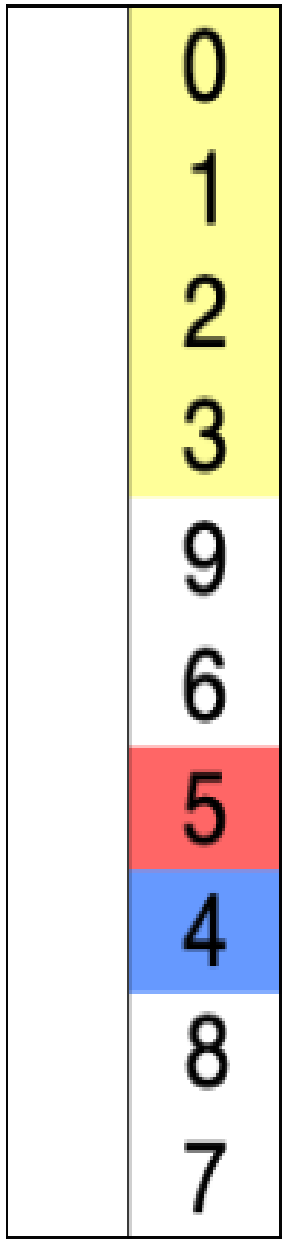
Selection sort

Opera dividendo la sequenza di input in due parti: la sottosequenza di elementi già *ordinati* (che occupa le prime posizioni dell'array) e la sottosequenza di elementi *non ordinati* (che occupa il resto dell'array).

Selection sort

Inizialmente, la sottosequenza ordinata è vuota, mentre quella non ordinata rappresenta l'intero input. L'algoritmo seleziona di volta in volta il numero minore nella sottosequenza non ordinata e lo sposta in quella ordinata.

https://upload.wikimedia.org/wikipedia/commons/9/94/Selection-Sort-Animation.gif

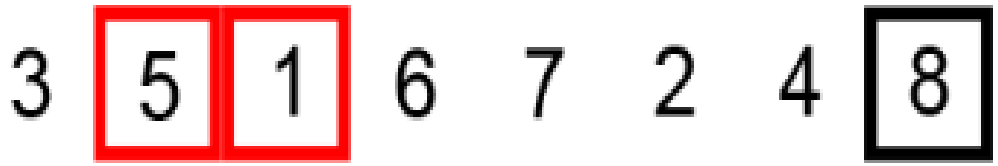


Bubble sort

Bubble sort

Il **bubble sort** è un semplice algoritmo di ordinamento non molto efficiente. Difatti, si utilizza a scopi didattici grazie alla sua semplicità e per introdurre gli sviluppatori agli algoritmi ed alla complessità computazionale.

https://upload.wikimedia.org/wikipedia/commons/0/06/Bubble-sort.gif



Bubble sort

Avendo la lista [6, 8, 3, 9, 5, 7, 4], scrivere in output a coppie i vari valori della lista:

6 8

8 3

3 9

9 5

5 7

7 4

Bubble sort

Avendo la lista [0, 1, 2, 3] scambiare di posizione gli elementi ad indice 0 con indice 2, poi quelli ad indice 0 con indice 3. Questo dovrà essere svolto usando un’unica variabile temporanea. Una volta fatti gli scambi, la lista avrà il seguente ordine:

[3, 1, 0, 2]

Bubble sort

Avendo la lista [6, 8, 3, 9, 5, 7, 4], controllare a coppie se il valore a sinistra è più grande di quello a destra. In caso positivo invertire i due valori. Nel nostro caso la prima coppia [6, 8] rimarrà invariata; poi [8, 3] verranno invertiti, diventando [3, 8]; poi [8, 9] rimarrà invariata; [9, 5] verranno invertiti, e via dicendo. Il risultato finale sarà:

[6, 3, 8, 5, 7, 4, 9]

Bubble sort

Avendo la lista [6, 8, 3, 9, 5, 7, 4], scrivere in output a coppie i vari valori della lista. Nella prima riga considerare tutti i valori, nella seconda riga solo i primi 6 valori, nella terza riga i primi 5 valori e via dicendo:

6 8, 8 3, 3 9, 9 5, 5 7, 7 4,

6 8, 8 3, 3 9, 9 5, 5 7,

6 8, 8 3, 3 9, 9 5,

6 8, 8 3, 3 9,

6 8, 8 3,

6 8,

Bubble sort

Ordinare la lista [6, 8, 3, 9, 5, 7, 4] utilizzando l’algoritmo bubble sort.

Insertion sort

Insertion sort

Molto efficiente per l’ordinamento di array di dimensione molto piccola o per array parzialmente ordinati.

L’idea di ordinamento è simile al modo in cui un giocatore di bridge ordina le carte nella propria mano.

Insertion sort

Si inizia con la mano vuota e le carte capovolte sul tavolo, poi si prende una carta alla volta dal tavolo e la si inserisce nella giusta posizione. Per trovare la giusta posizione per una carta, la si confronta con le altre carte nella mano, da destra verso sinistra.

https://upload.wikimedia.org/wikipedia/commons/0/0f/Insertion-sort-example-300px.gif

Insertion sort

Utilizzando 2 cicli for annidati, ottenere il seguente output:

0

1 0

2 1 0

3 2 1 0

4 3 2 1 0

5 4 3 2 1 0