List Arguments Summarize

Questo PDF presenta un sunto di tutti gli argomenti che saranno coperti durante le lezioni riguardanti le liste.

- **List Structure**
    - Sintassi `[1, 2, 3, 4, 5, ...]`
    - Contenuto di ogni tipo `[1, "uno", 2, [1, 2], "tre"]`
    - Differenze con le stringhe
        - Le stringhe sono *iterables* delimitati dai **doppi apici**
        - Le liste sono *iterables* delimitati dalle **parentesi quadre**
        - Le liste possono contenere ogni tipo di *object* (str, list, int), mentre le stringhe
          no. Da notare che se io avessi una stringa come questa `"ciao1[2, 2, 3]"`, dovrei
          considerare 1 o `[2, 2, 3]` come delle sottostringhe, quindi *str object* e non *list*
          o *int*.
        - Le stringhe e le liste differiscono anche per i **built-in methods**
- **List Properties**
    - Ordinamento non **in-structure**. Posso creare una lista non ordinata come questa `[3, 1, 2]`
    - **Indexing**. Posso accedere ad un elemento della lista tramite il suo indice
    - **Extensibility**. Posso aumentare o decrementare la sua dimensione
- **Loop on lists**
    - Scansionare una lista utilizzando un **For Loop**

```
l = [1, 2, 3, 4, 5, "uno", "due", "tre", "quattro", "cinque"]
for x in l:
    print(x)
```

```
output:
1
2
3
4
5
uno
due
tre
quattro
cinque
```

    - Scansionare una lista utilizzando un **While Loop**

```
l = [1, 2, 3, 4, 5, "uno", "due", "tre", "quattro", "cinque"]
index=0
while index <= len(l) - 1:
    print(l[index])
    index = index + 1
```

```
output:
1
2
3
4
5
uno
due
tre
quattro
cinque
```

Considera la lista seguente come esempio di partenza `l = [1, 2, 3, 4]`

- **Lists Methods**
    - **Append method**. Appende (inserisce alla fine) un elemento dato in input.

```
l.append(5)
```

```
output: None
modified list: [1, 2, 3, 4, 5]
```

    - **Clear**. Rimuove tutti gli elementi da una lista.

```
l.clear()
```

```
output: None
modified list: []
```

    - **Copy**. Ritorna una copia della lista.

```
l.copy()
```

```
output: [1, 2, 3, 4]
modified list: None
```

    - **Extend**. Inserisce gli elementi di una lista data in input in quella originale.

```
l.extend([5, 6, 7, 8])
```

```
output: None
modified list: [1, 2, 3, 4, 5, 6, 7, 8]
```

    - **Count**. Ritorna il numero di occorrenza, nella lista, di un elemento dato in input.

```
l.count(3)
```

```
output: 1
modified list: None
```

- **Index**. Ritorna la posizione di un dato elemento nella lista, errore se l'elemento non vi
  appartiene.

```
l.index(5)
l.index(24)
```

```
output 1: 4
modified list 1: None

output 2: "ValueError: 24 is not in list"
modified list 2: None
```

- **Insert**. Inserisce un dato elemento nella lista, nella posizione specificata.

```
l.insert(0, "I'm at the beginning")
l.insert(len(l)//2 + 1, "I'm at the middle")
l.insert(len(l), "I'm at the end")
```

```
output 1: None
modified list 1: ["I'm at the beginning", 1, 2, 3, 4]

output 2: None
modified list 2: ["I'm at the beginning", 1, 2, "I'm at the middle", 3,
4]

output 3: None
modified list 3: ["I'm at the beginning", 1, 2, "I'm at the middle", 3,
4, "I'm at the end"]
```

- **Pop**. Rimuove e ritorna un elemento dalla lista. Senza input ritorna l'ultimo, oppure
  quello nella posizione indicata.

```
l.pop()
l.pop(2)
```

```
output 1: 4
modified list: [1, 2, 3]

output 2: 3
modified list: [1, 2]
```

- **Remove**. Rimuove un dato elemento dalla lista, solo se questo vi appartiene.

```
l.remove(4)
l.remove(4)
```

```
output 1: None
modified list: [1, 2, 3]

output 2: "ValueError: list.remove(x): x is not in list"
modified list: [1, 2, 3]
```

- **Reverse**. Fa il reverse (scrive al contrario) della lista.

```
l.reverse()
```

```
output: None
modified list: [4, 3, 2, 1]
```

- **Sort**. Ordina la lista.

```
l = [4, 2, 3, 1]
l.sort()
```

```
output: None
modified list: [1, 2, 3, 4]
```

- **From the strings to the lists**
  - Il metodo *Split* delle stringhe. Data una stringa in input, divide l'input per un separatore
    sep dato anch'esso in input e ritorna una lista con tutte le parole che erano prima o dopo il
    separatore.

```
stringa = "ciao come stai"
lista = stringa.split() # separatore è lo spazio
print(lista)
```

```
output: ["ciao", "come", "stai"]
```

  - Formattazione di stringhe tramite il metodo *Join*. Essenzialmente, data una stringa che
    contiene uno o più separatori, formatta la stringa mettendo a sinistra o a destra del
    separatore le parole che stanno all'interno della lista data in input.

```
stringa = "ciao come stai"
lista = stringa.split("o") # ["cia", " c", "me stai"]
print("/".join(lista))

print("-".join([21, 11, 2019]))
```

```
output 1: "ciao/ c/me stai"
output 2: "21-11-2019"
```