

3-Strutture dati

November 14, 2020

1 Strutture dati

Benvenuti alla nuova lezione sulle strutture dati di Python, partiamo dalla definizione di strutture di dati, per strutture dati si intende un modo di organizzare, gestire e immagazzinare dati in maniera efficiente e, anche se non sempre, modificabile. In python esistono diversi tipi di strutture dati che sono: - Liste - Tuple - Sets - Dictionaries

1.1 Liste

Una lista è una collezione di valori **modificabile** definita secondo la classe `list` ed in grado di contenere una qualsiasi collezione di dati il cui ordine è definito da un **indice** che parte da **0**, per chi avesse già esperienza sono gli analoghi degli array 1D in altri linguaggi, in python le parentesi `[]` definiscono una lista o la parola chiave `list(elementi)`, vediamo alcuni esempi:

```
[1]: numbers = [1, 2, 3] #questa è una lista che contiene solo numeri
letters = ['a', 'b', 'c'] #questa è una lista di lettere
mix = [1, 'ciao', 4] #questa è una lista con diversi tipi di valori, sia
    →numeri, che stringhe
#stampiamo i valori delle liste
print('la lista numbers contiene: ', numbers)
print('la lista letters contiene: ', letters)
print('la lista mix contiene: ', mix)
#per sapere il tipo della lista
print('Il tipo lista è definito come: ', type(numbers))
```

```
la lista numbers contiene: [1, 2, 3]
la lista letters contiene: ['a', 'b', 'c']
la lista mix contiene: [1, 'ciao', 4]
Il tipo lista è definito come: <class 'list'>
```

Niente impedisce la creazione di liste contenenti altre liste, questo è un concetto che verrà utilizzato in futuro per alcune trattazioni, queste liste sono dette **nested** ed è possibile definirle nel seguente modo:

```
[2]: list_of_list = [1, 2, ['a', 'b', 45], 'fine'] #lista di liste
#stampiamo il valore e il tipo
print('list_of_list contiene: ', list_of_list)
print('Il tipo di list_of_list è: ', type(list_of_list))
```

```
list_of_list contiene: [1, 2, ['a', 'b', 45], 'fine']  
Il tipo di list_of_list è: <class 'list'>
```

Per sapere quanti elementi sono presenti all'interno della lista è possibile usare il comando `len`.

```
[3]: print('la lista numbers ha', len(numbers), 'elementi')
```

la lista numbers ha 3 elementi

Per sapere se un elemento è contenuto nella lista possiamo usare `in`.

```
[4]: 3 in numbers
```

```
[4]: True
```

1.1.1 Operazioni sulle liste

Le liste posseggono una serie di funzioni proprie attraverso cui sono possibili modificare o accedere agli elementi contenuti in essa, vediamo alcuni `####` Accedere a singoli o multipli elementi `###` Usando l'indice è possibile selezionare singoli o multipli elementi all'interno della lista.

Accedere al singolo elemento attraverso il comando `lista[indice]`:

```
[5]: #ricorda che l'indice della lista parte da 0!  
print('Il primo elemento della lista numbers è: ', numbers[0])  
print('Il secondo elemento della lista numbers è: ', numbers[1])  
print('Il terzo elemento della lista numbers è: ', numbers[2])
```

Il primo elemento della lista numbers è: 1

Il secondo elemento della lista numbers è: 2

Il terzo elemento della lista numbers è: 3

Accedere a molteplici elementi attraverso il comando `lista[inizio:fine]`:

```
[6]: print('dal primo al secondo elemento della lista numbers: ', numbers[0:2])  
      #Altro modo sarebbe potuto essere  
print('sempre dal primo al secondo elemento di numbers: ', numbers[:2])  
print('dal secondo al terzo elemento della lista numbers: ', numbers[1:3])  
print('Tutti gli elementi della lista numbers: ', numbers[:])
```

dal primo al secondo elemento della lista numbers: [1, 2]

sempre dal primo al secondo elemento di numbers: [1, 2]

dal secondo al terzo elemento della lista numbers: [2, 3]

Tutti gli elementi della lista numbers: [1, 2, 3]

Notate bene che l'indice finale non accede all'elemento dell'indice usato, ma a quello precedente, questa è una convenzione usata in alcuni linguaggi, tra cui python.

Funzioni delle liste Le liste contengono delle funzioni a cui è possibile accedere attraverso `.funzione()` questo concetto appartiene alla programmazione ad oggetti che verrà spiegato in

seguito, per ora a noi è sufficiente capire cosa fanno e come usarle. ##### Appendere un nuovo elemento alla fine #####

```
[7]: numbers.append(4)
print('Lista numbers con un nuovo elemento: ', numbers)
```

Lista numbers con un nuovo elemento: [1, 2, 3, 4]

Estendere una lista

```
[8]: numbers.extend([5, 6, 7])
print('Lista numbers estesa: ', numbers)
```

Lista numbers estesa: [1, 2, 3, 4, 5, 6, 7]

Rimuovere un valore

```
[9]: numbers.remove(5) #dobbiamo passare il valore, non l'indice!
print('Lista numbers con il valore 5 rimosso: ', numbers)
```

Lista numbers con il valore 5 rimosso: [1, 2, 3, 4, 6, 7]

Ordinare una lista

```
[10]: #ordinare in ordine decrescente la lista
numbers.reverse() #oppure .sort(reverse = True)
print('Lista numbers in ordine decrescente: ', numbers)
#ordinare una lista in ordine crescente
numbers.sort()
print('Lista numbers in ordine crescente: ', numbers)
```

Lista numbers in ordine decrescente: [7, 6, 4, 3, 2, 1]

Lista numbers in ordine crescente: [1, 2, 3, 4, 6, 7]

Svuotare una lista

```
[11]: numbers.clear() #analogo a del numbers[:]
print('Lista numbers svuotata: ', numbers)
```

Lista numbers svuotata: []

Sono presenti altre funzionalità in liste, per conoscerle, consultate la [documentazione](#).

1.2 Tuple

Una tupla è simile a una lista con delle differenze, **una tupla non può essere modificata nei suoi elementi**, inoltre le tuple sono caratterizzate da parentesi () e di un separatore , a differenza delle liste, possibile definirla anche con tuple().

```
[12]: t = (123, 'ciao', 47) #tupla
print('la tupla t contiene: ', t)
print('la tupla t è del tipo: ', type(t))
```

la tupla t contiene: (123, 'ciao', 47)
la tupla t è del tipo: <class 'tuple'>

```
[13]: #non può essere modificata  
t[0] = 5
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-13-418e8d116b23> in <module>  
      1 #non può essere modificata  
----> 2 t[0] = 5  
  
TypeError: 'tuple' object does not support item assignment
```

```
[14]: #ma può essere usata come estensione o per ottenere elementi  
x = t[0]  
u = (t, 65, 'paolo')  
print('x è: ', x)  
print('u è: ', u, ' di tipo: ', type(u))
```

x è: 123
u è: ((123, 'ciao', 47), 65, 'paolo') di tipo: <class 'tuple'>

La [documentazione](#) fornisce maggiori dettagli.

1.3 Set

Un set è uguale alla tupla con la differenza che **non ci possono essere duplicati e non sono ordinati**, i set sono definiti con {} o set().

```
[15]: s = {1345, 'arco', 'fiume'}  
print('il set s contiene: ', s)  
print('s è del tipo: ', type(s))
```

il set s contiene: {1345, 'arco', 'fiume'}
s è del tipo: <class 'set'>

```
[16]: s[0] = 5 #immutabile
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-16-03b2ccee89a0> in <module>  
----> 1 s[0] = 5 #immutabile  
  
TypeError: 'set' object does not support item assignment
```

Set supporta delle operazioni a differenza di tuple, consultate pure la [documentazione](#) per ulteriori info.

1.4 Dictionaries

I dictionaries sono un'altra struttura di dati simili alle liste, ma caratterizzate da una **key** associata ad una **value** nella forma **key:value** il cui **indice è definito dalla key**, sono inoltre caratterizzate dalle parentesi {} o definite con dict().

```
[17]: #dizionario
capitali = {'piemonte' : 'Torino', 'lombardia' : 'Milano', 'Liguria': 'Genova'}
print('capitali contiene: ', capitali)
print('Tipo di capitali è: ', type(capitali))
```

```
capitali contiene: {'piemonte': 'Torino', 'lombardia': 'Milano', 'Liguria':
'Genova'}
Tipo di capitali è: <class 'dict'>
```

```
[18]: print(capitali['piemonte'])
capitali['piemonte'] = 'Roma' # mutabile
print(capitali['piemonte'])
```

```
Torino
Roma
```

```
[19]: #estendibile con una chiave e un valore associato
capitali['Lazio'] = 'Roma'
print(capitali)
```

```
{'piemonte': 'Roma', 'lombardia': 'Milano', 'Liguria': 'Genova', 'Lazio':
'Roma'}
```

```
[20]: #ordinati secondo le CHIAVI
print(sorted(capitali))
```

```
['Lazio', 'Liguria', 'lombardia', 'piemonte']
```

Per avere maggiori informazioni potete consultare la [documentazione](#).

COMPLIMENTI AVETE FINITO LA LEZIONE SULLE STRUTTURE DI DATI IN PYTHON!