Le6 Anaconda

December 4, 2019

1 Esercitazione 6

1.1 Dizionari

I dizionari sono dinamici e mutabili.

Un dizionario è una collezione di elementi costituiti da coppie di chiavi e valori, dove ad ogni chiave corrisponde il suo valore. <dizionario> = "<key>": <valore>, "<key>": valore, ... "<key>": <valore>

[ ]:
```
spesa = {"Mela": 10, "Pasta": 2, "Coca Cola": 1.50}
```

[ ]:
```
spesa
```

1.1.1 Accedere agli elementi

[ ]:
```
spesa = {"Mela": 10, "Pasta": 2, "Coca Cola": 1.50}
print(spesa["Pasta"]) # Accesso mediante chiave key
```

[ ]:
```
spesa = {"Mela": 10, "Pasta": 2, "Coca Cola": 1.50}
print(spesa["Pera"])
```

1.1.2 Modificare e Aggiungere elementi

Sia per aggiungere o che per modificare un dizionario ci basterà assegnare il valore che vogliamo alla chiave che vogliamo aggiungere/modificare:

[ ]:
```
spesa = {"Mela": 10, "Pasta": 2, "Coca Cola": 1.50}

spesa["Pera"] = 3.5
print("Pera aggiunta: " + spesa["Pera"])
print("Pera aggiunta: " + str(spesa["Pera"]))
print("Pera aggiunta: {}".format(spesa["Pera"]))

spesa["Pera"] = 2
print("Pera modificata: {}".format(spesa["Pera"]))
```

1.1.3 Rimuovere elementi

Per rimuovere una coppia di chiavi/valori, si utilizza la parola chiave del, seguita dalla chiave che vogliamo eliminare:

[ ]:
```
spesa = {"Mela": 10, "Pasta": 2, "Coca Cola": 1.50}
print(spesa)
del spesa["Mela"]
print(spesa)
```

1.1.4 Metodi built-in

dizionario.get(<chiave>, [<default>]) Con il metodo get possiamo ottenere il valore associato alla chiave specificata. Tuttavia, se la chiave non dovesse esistere, invece di sollevare un'eccezione, mi ritornerà come risultato un valore impostato nel parametro opzionale default. Inoltre se viene omesso il valore di default e la chiave inserita non esiste, avremmo come risultato None.

[ ]:
```
persone = {"Marco": 45, "Gigi": 28}
print(persone.get("Marco"))
print(persone.get("Antonio"))
print(persone.get("Antonio", "Non Trovato"))
```

Un altro metodo utilie è clear(), che ci permette di pulire il nostro dizionario rimuovendo tutto il suo contenuto:

[ ]:
```
persone = {"Marco": 45, "Gigi": 28}
print(persone)
persone.clear()
print(persone)
```

Con il metodo keys() otteniamo una lista di tuple di tutte le chiavi presenti nel nostro dizionario:

[ ]:
```
frutta = {"Mela": 2, "Banana": 1, "Pera": 4}
lista_keys = list(frutta.keys())
print(lista_keys)
```

In maniera analoga il funzionamento di values() per tutti i valori di un dizionario:

[ ]:
```
spesa = {"Birra": 1.60, "Salame": 2.50, "Nutella": 2}
somma = sum(spesa.values())
print(somma)
```

Infine abbiamo il metodo items() il quale ci ritorna una lista di tuple contenente sia le chiavi che i rispettivi valori del nostro dizionario:

[ ]:
```
frutta = {"Mela": 2, "Banana": 1, "Pera": 4}
lista_items = list(frutta.items())
print(lista_items)
```

1.1.5 Iterare sui dizionari

[ ]:
```
spesa = {
 "Birra": {"Quantita": 6, "Prezzo": 1.60},
 "Patatine": {"Quantita": 3, "Prezzo": 1},
 "Latte": {"Quantita": 2, "Prezzo": 1.20}
 }

tot = 0
for elemento in spesa.keys():
 tot += spesa[elemento]["Quantita"] * spesa[elemento]["Prezzo"]

print(tot)
```

Esercizio 1 Data una stringa S, contare quante volte compare ciascuna lettera. Utilizzare i dizionari

Esercizio 2 Rappresentare mediante un dizionario la rete definita nel file grafo.png. Il programma deve prevedere metodi per l'inserimento di nuovi nodi, l'inserimento di nuovi archi, la modifica di archi, la cancellazione di archi e di nodi. L'utente deve scegliere una delle operazioni previste mediante menu visibile su terminale.

[ ]:
```
def conta_caratteri(s):
    d = dict()
    for c in s:
        if c not in d:
            d[c] = 1
        else:
            d[c] += 1
    return d

conta_caratteri("pippo")
```

[ ]:
```

```