COMPSCI 1☺1

Principles of Programming

Lecture 23 – More on dictionaries, using dictionaries to manage a small file of information

Learning outcomes

- At the end of this lecture, students should be able to:
  - Delete key:value pairs from a dictionary
  - Create a list of keys, values, key:value tuples from a dictionary
  - Use dictionary objects to manage a small file of information

Recap

- Dictionaries - dictionaries are used to store key:value pairs (items)
  - a dictionary object can be created in two ways
  - items can be added to a dictionary
  - Items can be retrieved from the dictionary
  - the pairs in a dictionary can be traversed using for … in

```
def main():
  english_italian = {"yes":"si", "bye":"ciao",
                     "no":"no", "maybe":"forse",
                     "thank you":"grazie"}
  english_italian["never"] = "mai"
  print(english_italian["bye"] )
  for word in english_italian:
      print(english_italian[word])
  print(len(english_italian))
main()
```

```
ciao
mai
no
forse
ciao
si
grazie
6
```

Deleting a key:value pair from the dict object

- The **del** operator is used to delete a key:value pair from the dictionary.

```
def main():
  my_dict = {"a": 4, "b": 6, "c": 5}
  print("1.", my_dict)

  del my_dict["b"]
  print("2.", my_dict)

  del my_dict["a"]
  print("3.", my_dict)

main()
```

```
1. {'a': 4, 'b': 6, 'c': 5}
2. {'a': 4, 'c': 5}
3. {'c': 5}
```

Deleting a key:value pair from a dict object

- The **del** operator gives an error if the key of the key:value pair being deleted is not in the dictionary. Because of this, it is customary to test before deleting a key:value pair.

```
def main():
  my_dict = {"a": 4, "b": 6, "c": 5}
  print("1.", my_dict)

  if "b" in my_dict:      #Test first
     del my_dict["b"]
  print("2.", my_dict)

  del my_dict["z"]
  print("3.", my_dict)

main()
```

```
1. {'a': 4, 'b': 6, 'c': 5}
2. {'a': 4, 'c': 5}
.... Other error information
KeyError: 'z'
```

Methods which can be used with a dict object

- The keys, the values, the associations as tuples, can be obtained from a dictionary object using the methods:

  my_dict = {…}

  my_dict**.items()** – to access all the key/value pairs as tuples

  my_dict**.keys()** – to access all the keys

  my_dict**.values()** – to access all the values

- The elements in these collections can be accessed using a for … in loop.

```
def main():
  my_dict = {"a": 4, "b": 6, "c": 5}
  for letter in my_dict.keys():
     print(letter)
  for number in my_dict.values():
     print(number)
  for item in my_dict.items():
     print(item)
main()
```

```
b
c
a
6
5
4
('b', 6)
('c', 5)
('a', 4)
```

Methods which can be used with a dict object

- When a for … in loop is used with a dictionary object, Python loops through each key in the dictionary:

```
def main():
  my_dict = {"a": 4, "b": 6, "c": 5}

  for letter in my_dict.keys():
     print(letter)


  for key in my_dict:
     print(key)

main()
```

Note that both these loops do the same job.

```
b
c
a
b
c
a
```

Methods which can be used with a dict object

- Often it is useful to convert the individual keys (or values, or item tuples) of the dictionary into lists by enclosing the keys (or values, or item tuples) in list():

```
def main():
  my_dict = {"a": 4, "b": 6, "c": 5}
  items_list = list(my_dict.items())
  keys_list = list(my_dict.keys())
  values_list = list(my_dict.values())

  print("items list", items_list)
  print("keys list", keys_list)
  print("values list", values_list)

main()
```

```
items list [('a', 4), ('c', 5), ('b', 6)]
keys list ['a', 'c', 'b']
values list [4, 5, 6]
```

Note on deleting key-value pairs from dictionary objects

If you try and remove elements from a dict object while iterating through its keys using a for … in loop, you will get an error.

```
def main():
    my_dict = {"and":4,"many":2,"for":5,"very":1}
    for key in my_dict:
        del my_dict[key]

main()
```

RuntimeError: dictionary changed size during iteration

Instead, create a separate list of the dictionary keys, iterate through this list and delete any unwanted items from the dict object:

```
def main():
    my_dict = {"and":4,"many":2,"for":5,"very":1}
    print(my_dict)
    keys_list = list(my_dict.keys())
    for key in keys_list:
        del my_dict[key]
    print(my_dict)

main()
```

```
{'and': 4, 'many': 2, 'for': 5, 'very': 1}
{}
```

Using dictionaries - Our file information

- We wish to manage a small file of ratings for four films.
- The film list is:

```
film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]
```

- The text file, "Ratings.txt", stores the ratings made by seven people of the four films (0 means the person didn't rate the film, 1 means the person hated the film, 9 means they loved it):

Ratings.... — Edited

```
Mary 2 0 6 2
Joy 2 8 3 9
Adam 7 2 0 7
Li 0 2 3 8
Jo 3 2 0 8
Sam 9 2 3 8
John 0 9 4 8
```

Loading the information

- Firstly we read all the lines of text from the file into a list (removing the newline character - "\n" - from the end of each line).

Ratings.... — Edited

```
Mary 2 0 6 2
Joy 2 8 3 9
Adam 7 2 0 7
Li 0 2 3 8
Jo 3 2 0 8
Sam 9 2 3 8
John 0 9 4 8
```

```
def get_lines_from_file(filename):
    ????




def main():
    film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]

    number_of_films = len(film_list)
    filename = "Ratings.txt"

    lines_of_text = get_lines_from_file(filename)

main()
```

["Mary 2 0 6 2", "Joy 2 8 3 9", ...]

Loading the file information into dictionaries

- person_name : list of ratings dictionary, i.e., the person_name is the key and the list of ratings is the corresponding value.

Ratings.... — Edited

```
Mary 2 0 6 2
Joy 2 8 3 9
Adam 7 2 0 7
Li 0 2 3 8
Jo 3 2 0 8
Sam 9 2 3 8
John 0 9 4 8
```

→

["Mary 2 0 6 2", "Joy 2 8 3 9", ...]

↓

```
{  "Mary": [2, 0, 6, 2],
   "Joy": [2, 8, 3, 9],
    ...
}
```

Loading the file information into dictionaries

- From all the 'lines of text' list: ["Mary 2 0 6 2", "Joy 2 8 3 9", ...],
  we wish to create a dictionary: person_name : list of ratings

```
def get_people_ratings_dict(lines_of_text):
   people_ratings = {}





   return people_ratings

def main():
   film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]
   number_of_films = len(film_list)
   filename = "Ratings.txt"
   lines_of_text = get_lines_from_file(filename)
   people_ratings_dict = get_people_ratings_dict(lines_of_text)

main()
```

{"Mary": [2, 0, 6, 2], "Joy": [2, 8, 3, 9], ...}

Loading the file information into dictionaries

```
film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]
```

- person_name : list of ratings dictionary (see slides 12 and 13)
- film_title : list of ratings dictionary, i.e., the film_title is the key and the list of seven ratings (one from each person) is the corresponding value.

Ratings.... — Edited

```
Mary 2 0 6 2
Joy 2 8 3 9
Adam 7 2 0 7
Li 0 2 3 8
Jo 3 2 0 8
Sam 9 2 3 8
John 0 9 4 8
```

```
["Mary 2 0 6 2", "Joy 2 8 3 9", ...]
```

↓

```
{ "Mary": [2, 0, 6, 2],
  "Joy": [2, 8, 3, 9],
  ...
}
```

↓

```
{"Jaws" : [2, 2, 7, 0, 3, 9, 0]
"The Goonies" : [0, 8, 2, 2, 2, 2, 9]
"Aliens": [6, 3, 0, 3, 0, 3, 4]
"Commando" : [2, 9, 7, 8, 8, 8, 8]
}
```

Loading the information into dictionaries

- From the people dictionary **{"Mary": [2, 0, 6, 2], "Joy": [2, 8, 3, 9], …}**

  we wish to create another dictionary: film_title:list of ratings

```
def get_film_ratings_dict(film_list, people_ratings_dict):
   #Jaws – get the first rating from every person
   #The Goonies– get the second rating from every person, etc.
   film_index = 0
   film_ratings_dict = {}

   return film_ratings_dict

def main():
   film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]
   number_of_films = len(film_list)
   filename = "Ratings.txt"
   lines_of_text = get_lines_from_file(filename)
   people_ratings_dict = get_people_ratings_dict(lines_of_text)
   film_ratings_dict = get_film_ratings_dict(film_list,
                                              people_ratings_dict)

main()
```

{'Jaws': [2, 2, 7, 0, 3, 9, 0], 'The Goonies': [0, 8, 2, 2, 2, 2, 9], …}

The two dictionaries

- So far, from the film list:

```
film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]
```

and the ratings information in the file:

Ratings.... — Edited

```
Mary 2 0 6 2
Joy 2 8 3 9
Adam 7 2 0 7
Li 0 2 3 8
Jo 3 2 0 8
Sam 9 2 3 8
John 0 9 4 8
```

we have created two dictionaries:

people_ratings_dict

```
{
'Mary': [2, 0, 6, 2],
'John': [0, 9, 4, 8],
'Adam': [7, 2, 0, 7],
'Sam': [9, 2, 3, 8],
'Joy': [2, 8, 3, 9],
'Jo': [3, 2, 0, 8],
'Li': [0, 2, 3, 8]
}
```

film_ratings_dict

```
{
"Jaws": [2, 2, 7, 0, 3, 9, 0]
"The Goonies": [0, 8, 2, 2, 2, 2, 9]
"Aliens": [6, 3, 0, 3, 0, 3, 4]
"Commando": [2, 9, 7, 8, 8, 8, 8]
}
```

Using the dictionaries

- The user can select a person's name from the dictionary keys, see the person's ratings list as well as the average of that person's non-zero ratings.

```
def process_person_ratings_request(people_ratings_dict):
    ???

def main():
    …
    process_person_ratings_request(people_ratings_dict)

main()
```

people_ratings_dict

```
{
'Mary': [2, 0, 6, 2],
'John': [0, 9, 4, 8],
'Adam': [7, 2, 0, 7],
'Sam': [9, 2, 3, 8],
'Joy': [2, 8, 3, 9],
'Jo': [3, 2, 0, 8],
'Li': [0, 2, 3, 8]
}
```

```
John
Mary
Adam
Jo
Joy
Li
Sam
Enter name: Sam
[9, 2, 3, 8] Sam - average rating: 5.5
```

Using the dictionaries

- The user can select a person from the dictionary keys and see the person's ratings list as well as the average of their non-zero ratings.

{"Mary": [2, 0, 6, 2], "Joy": [2, 8, 3, 9], ...}

```
def process_person_ratings_request(people_ratings_dict):


def display_keys(dictionary):
    ???

def get_average_rating(list_of_numbers):
    ???

def main():
    film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]
    number_of_films = len(film_list)
    filename = "Ratings.txt"
    lines_of_text = get_lines_from_file(filename)
    people_ratings_dict = get_people_ratings_dict(lines_of_text)
    film_ratings_dict = get_film_ratings_dict(film_list, people_ratings_dict)
    print("Process People-Rating Request")
    process_person_ratings_request(people_ratings_dict)
```

Using the dictionaries

- The user can select a film from a list of titles, see the film's ratings as well as the average of all the non-zero ratings for that film.

```
film_ratings_dict
{
"Jaws": [2, 2, 7, 0, 3, 9, 0]
"The Goonies": [0, 8, 2, 2, 2, 2, 9]
"Aliens": [6, 3, 0, 3, 0, 3, 4]
"Commando": [2, 9, 7, 8, 8, 8, 8]
}
```

```
def process_film_ratings_request(film_list, film_ratings_dict):
    ???

def main():
    …
    process_film_ratings_request(film_list, film_ratings_dict)
main()
```

```
Process Film-Rating Request
1 Jaws
2 The Goonies
3 Aliens
4 Commando
Enter selection: 1
[2, 2, 7, 0, 3, 9, 0] Jaws - average rating: 4.6
```

Using the dictionaries

- The user can select a film from a list of titles, and see the film's ratings as well as the average of all the non-zero ratings for that film.

{‘Jaws': [2, 2, 7, 0, 3, 9, 0], 'The Goonies': [0, 8, 2, 2, 2, 2, 9], …}

```
def process_film_ratings_request(film_list, film_ratings_dict):



def display_numbered_list(list_of_items):
   ???

def get_average_rating(list_of_numbers):
   #see previous code

def main():
   film_list = ["Jaws", "The Goonies", "Aliens", "Commando"]
   number_of_films = len(film_list)
   filename = "Ratings.txt"
   lines_of_text = get_lines_from_file(filename)
   people_ratings_dict = get_people_ratings_dict(lines_of_text)
   film_ratings_dict = get_film_ratings_dict(film_list, people_ratings_dict)
   print("Process Movie-Rating Request")
   process_film_ratings_request(film_list, film_ratings_dict)
```

Summary

- The **del** operator is used to delete an key:value pair from the dictionary.
- The keys, the values, the associations as tuples can be obtained from a dictionary object using the methods:

  my_dict**.items()** – to access all the key/value pairs as tuples

  my_dict**.keys(**) – to access all the keys

  my_dict**.values()** – to access all the values

Often it is useful to convert the individual keys (or values, or item tuples) of the dictionary into lists by enclosing the keys (or values, or item tuples) in list()

Python features used in this lecture

```
my_dict = {"a": 4, "b": 6, "c": 5}

for letter in my_dict.keys():
   print(letter)
for number in my_dict.values():
   print(number)
for item in my_dict.items():
   print(item)

items_list = list(my_dict.items())
keys_list = list(my_dict.keys())
values_list = list(my_dict.values())

print("items list", items_list)
print("keys list", keys_list)
print("values list", values_list)

if "b" in my_dict:     #Test first
   del my_dict["b"]
```