TMTplus Introduction to Scientific Programming

Mahdi Farnaghi, Mahdi Khodadadzadeh & Robert Ohuru

March 2021

Exercise 5

Sets & related expressions

5.1 The use of sets

Ex 5.1

Set creation

```
movies_set1 = set(["The␣Addams␣Family", "Ghostbusters",
"Jurassic␣Park","Pulp␣Fiction", "Home␣Alone", "The␣Matrix"])

movies_set2 = {"The␣Addams␣Family", "Ghostbusters", "Jurassic␣Park",
"Pulp␣Fiction", "Home␣Alone", "The␣Matrix"}
```

Ex 5.2

Create a set by converting a list using `set(mylist)`

```
movies_list = ["The␣Addams␣Family", "Ghostbusters", "Jurassic␣Park",
"Pulp␣Fiction", "Home␣Alone", "The␣Matrix"]

movies_set3 = set(movies_list)

# check for difference between sets created using different methods
movies_set2 == movies_set3
```

There is no difference between sets created using different methods.

```
# using a set method, add the movie Star Wars to it
movies_set3.add("Star␣Wars")

# using another set method, remove the movie Ghostbusters from it
movies_set3.remove("Ghostbusters")

# determine the size of the thus obtained set
len(movies_set3)

# verifies whether the movies Star Wars & Ghostbusters are in the set
"Star␣Wars" in movies_set3
```

```
"Ghostbusters" in movies_set3

# create a shallow copy of this set.
# next delete the original set
# look into what has happened to the copy.
movies_set4 = movies_set3.copy()
movies_set3.clear()
movies_set4
```

Ex 5.3 Set a contains all the unordered elements or letters of the string `'The Addams Family'`

If the above code is ran several times, the elements appear in different order each time because *sets are unordered data structures*.

Modify the previous script so as to have it create a set with only one element, with the full movie name

```
set_a = set(["The␣Addams␣Family"])
print(a)
```

Ex 5.4 Print the list `[12,24,35,24,88,120,155,88,120,155]` with unique values by converting the list to set with unique elements

```
list_1 = [12,24,35,24,88,120,155,88,120,155]
list_2 = list(set(list_1))
print(list_2)
```

Ex 5.5 With the two given sets `a={1,3,6,78,35,55}` and `b={12,24,35,24,88,120,155}`, fill the following table about various *set operations*.

operation	code	result
difference	`a - b`	`{1, 3, 6, 78, 55}`
intersection	`a & b`	`{35}`
union	`a | b`	`{1, 3, 35, 6, 88, 12, 78, 55, 24, 155, 120}`
symmetric difference	`a ^ b`	`{1, 3, 6, 12, 78, 24, 88, 155, 55, 120}`
subset	`a <= b`	`False`
superset	`a >= b`	`False`

Ex 5.6 Using the same sets as above, fill the following table by making use of *set methods*.

operation	code	result
difference	`a.difference(b)`	`{1, 3, 6, 78, 55}`
intersection	`a.intersection(b)`	`{35}`
union	`a.union(b)`	`{1, 3, 35, 6, 88, 12, 78, 55, 24, 155, 120}`
symmetric difference	`a.symmetric_difference(b)`	`{1, 3, 6, 12, 78, 24, 88, 155, 55, 120}`
subset	`a.issubset(b)`	`False`
superset	`a.issuperset(b)`	`False`

Ex 5.7

Complete the script to display each member on a line by itself

```
s = {"The␣Addams␣Family", "Ghostbusters", "Jurassic␣Park",
"Pulp␣Fiction", "Home␣Alone", "The␣Matrix"}

# using print
print(*s, sep="\n")

# iterate over set
for member in s:
  print(member)
```

Members in set are printed in random order because *sets are unordered data structures.*

Ex 5.8

Using the set above, it is not possible to access elements of a set using expressions like `s[1]` or `s[1:]` because *sets do not support indexing*

Using again the set from the previous exercise, write a script that prints only one arbitrary element of the set.

```
# convert set to list
list_s = list(s)
# there is no need for random indexing as
# everytime the set is converted to a different ordered list
# import random
# n=random.randint(0,len(s))
print(list_s[0])
```

Ex 5.9

The script returning an error because `s` is an immutable set. Edited script

```
s = set({"The␣Addams␣Family", "Ghostbusters", "Jurassic␣Park",
            "Pulp␣Fiction", "Home␣Alone", "The␣Matrix"})
s.remove("The␣Matrix")
print(s)
```