Sets

Prof.Dr. Bahadır AKTUĞ
BME362 Introduction To Python

**Compiled from sources given in the references*

Sets

- Similar to lists, sets can contain elements of different data types (strings / integers etc.)
- On the other hand, the elements of sets has to be immutable
- While the elements of a sets have to be immutable, the set itself is mutable
- Sets can be expanded or shrunk upon need
- One thing which distinguishes sets from other data types is that one element could take place in a set only once
- The sets in Python are very similar to the sets in Mathematics and the traditional set operations (subset, union, intersection, difference etc.) over Python sets are available and highly practical.

Sets

Defining a new set:

Set can be defined directly set() function or {} operator.

```
>>> cities = set(["Ankara", "Adana", "Samsun"])
>>> print(cities)
{'Ankara', 'Adana', 'Samsun'}
>>> cities = set({"Ankara":"06", "Adana": "01", "Samsun":"55"})
>>> print(cities)
{'Adana', 'Ankara', 'Samsun'}
>>> cities = {"Ankara", "Adana", "Samsun"}
>>> print(cities)
{'Adana', 'Ankara', 'Samsun'}
>>> x = set("EEE105")
>>> print(x)
{'E', '0', '5', '1'}
```

Set Functions

add():

The elements of sets have to be immutable. Thus, the argument for the "add()" function has to be immutable.

```
>>> colors = set(["Yellow", "Blue", "Green"])
>>> colors.add('Red')
>>> print(colors)
{'Green', 'Red', 'Blue', 'Yellow'}
>>> colors.add (['Red','Orange'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Set Functions

update():

An element can be added to an existing set by using add() function. If more than one elements are to be added, the "update()" function can be used.

```
>>> colors = set(["Yellow","Blue","Green"])
>>> colors.update(["Orange","Red"])
>>> print(colors)
{'Green', 'Red', 'Blue', 'Orange', 'Yellow'}
```

Set Functions

clear():

Deletes (clears) all the elements in a set. However, it does not delete the set itself, just makes it an empty set.

```
>>> colors = set(["Yellow","Blue","Green"])
>>> colors.clear()
>>> print(colors)
set()
```

Set Functions

copy():

As with other data types, direct assignment of sets (the are all objects) could lead to disastrous results. "copy()" function, as the name implies, copies the elements of a set to another set.

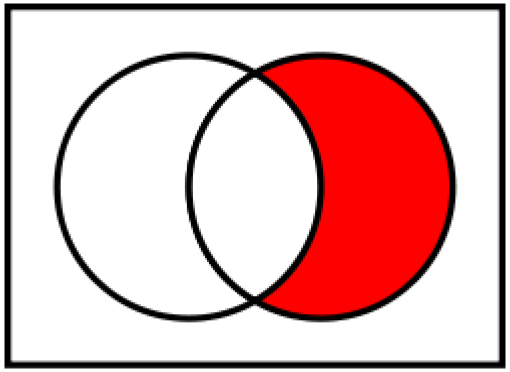
```
>>> colors = set(["Yellow", "Blue", "Green"])
>>> colors2 = colors
>>> colors2.clear()
>>> print(colors)
set()

>>> colors = set(["Yellow","Blue","Green"])
>>> colors2 = colors.copy()
>>> colors2.clear()
>>> print(colors)
{'Green', 'Blue', 'Yellow'}
```

Set Functions

difference(): (-)

Gives the difference between a set and another set (A \ B).



```
>>> x = {"a","b","c","d","e"}
>>> y = {"b","c"}
>>> z = {"c","d"}
>>> x.difference(y)
{'a', 'e', 'd'}
>>> x.difference(y).difference(z) {'a', 'e'}
```

The same operation can be done with "-" operator:

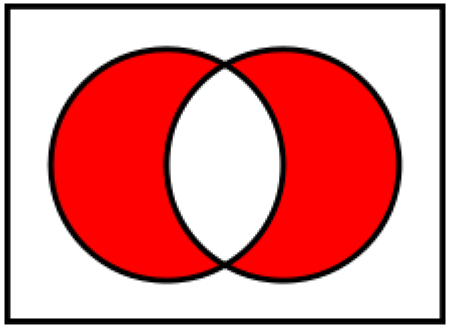
```
>>> x - y {'a', 'e', 'd'}
>>> x - y – z
{'a', 'e'}
```

Set Functions

symmetric_difference(): (^)

Corresponds to **disjunctive union** and gives the difference (non-overlapping parts) for both sets.

```
>>> x = {"a","b","c","d","e"}
>>> y = {"b","c"}
>>> z = {"c","d"}
>>> x ^ y
{'a', 'd', 'e'}
>>> y ^ z
{'b', 'd'}
>>> y - z
{'b'}
```



Set Functions

difference_update():

This is a combined operation of the difference (A \ B) and updating the original set (A) with the difference (A = A \ B)

```
>>> x = {"a","b","c","d","e"}
>>> y = {"b","c"}
>>> x.difference_update(y)
>>> print(x)
{'d', 'a', 'e'}
```

Set Functions

discard(element):

Deletes an element from a set. If the element is not in the set, it does not return an error message.

```
>>> x = {"a","b","c","d","e"}
>>> x.discard("a")
>>> x {'c', 'b', 'e', 'd'}
>>> x.discard("z")
>>> x
{'c', 'b', 'e', 'd'}
```

Set Functions

remove(element):

While similar to "discard" function, it gives an error message if the element to be removed is not in the set.

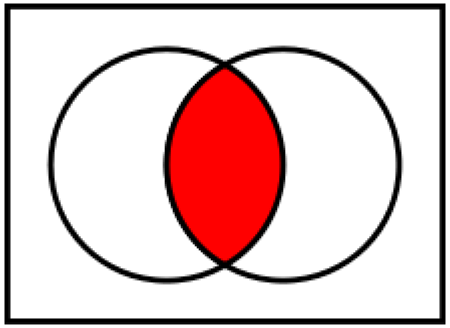
```
>>> x = {"a","b","c","d","e"}
>>> x.remove("a")
>>> x
{'b', 'c', 'd', 'e'}
>>> x.remove("z")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'z'
```

Set Functions

intersection(set): (&)

Return the intersection of a set with another set.

```
>>> x = {"a","b","c","d","e"}
>>> y = {"c","d","e","f","g"}
>>> x.intersection(y)
{'c', 'e', 'd'}
```



OR

```
>>> x & y
{'c', 'e', 'd'}
```

Set Functions

isdisjoint()

Returns a boolean expression about whether a set is disjoint with another set.

```
>>> x = {"a","b","c","d","e"}
>>> y = {"c","d","e","f","g"}
>>> x.isdisjoint(y)
False

>>> x = {"a","b","c","d","e"}
>>> y = {"h","j","k","f","g"}
>>> x.isdisjoint(y)
True
```

Set Functions

issubset() (<=)

Returns a boolean expression about whether a set is a subset of another set.

```
>>> x = {"c","d"}
>>> y = {"a","b","c","d","e"}
>>> x.issubset(y)
True
```

```
>>> x = {"c","d"}
>>> y = {"a","b","c","d","e"}
>>> x <= y
True
```

Set Functions

issuperset() (>=)

Returns a boolean expression about whether a set is a superset of another set.

```
>>> x = {"c","d"}
>>> y = {"a","b","c","d","e"}
>>> y.issuperset(x)
True
```

```
>>> x = {"c","d"}
>>> y = {"a","b","c","d","e"}
>>> y >= x
True
```

Set Functions

pop()

This function returns a random element of the set and deletes that element from the set.

```
>>> x = {"a","b","c","d","e"}
>>> x.pop()
'b'
>>> print(x)
{'c', 'a', 'd', 'e'}
>>> x.pop()
'c'
>>> print(x)
{'a', 'd', 'e'}
```

Set Functions

Multiple Sets:

Union and intersection functions can operate on more than one sets.

```
>>> x = {"a","b","c","d","e"}
>>> y = {"h","j","c","d","k"}
>>> z = {"i","b","e","d","n"}
>>> set.intersection(x,y,z)
{'d'}
>>> set.union(x,y,z)
{'j', 'd', 'i', 'b', 'c', 'a', 'k', 'n', 'h', 'e'}
```

Set Functions

Checking if element of a set:

```
>>> x = {"a","b","c","d","e"}
>>> 'c' in x
True
>>> 'p' in x
False
```

Set Functions

Loops over set elements:

```
>>> x = {"a","b","c","d","e"}
>>> for harf in x:
...   print(harf)
...
b
c
a
d
e
```

Set Functions

Frozen Sets:

While "Frozen sets" have the same properties as the normal sets, as the name implies, they cannot be modified once defined. Such immutable sets have further usage. For instance, they can keys for dictionaries and they can elements can other sets.

```
>>> x = frozenset(["a","b","c","d","e"])
>>> y = set([x,3,4,5])
>>> y
{3, 4, frozenset({'b', 'c', 'a', 'd', 'e'}), 5}
>>> x.add('f')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute 'add'
```

References

1 *Wentworth, P., Elkner, J., Downey, A.B., Meyers, C. (2014). How to Think Like a Computer Scientist: Learning with Python (3nd edition).*
2 *Pilgrim, M. (2014). Dive into Python 3 by. Free online version: DiveIntoPython3.org ISBN: 978-1430224150.*
3 *Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.*
4 *Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.*
5 *Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/.*
6 *Millman, K.J., Aivazis, M. (2011). Python for Scientists and Engineers, Computing in Science & Engineering, 13, 9-12.*
7 *John D. Hunter (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95.*
8 *Travis E. Oliphant (2007). Python for Scientific Computing, Computing in Science & Engineering, 9, 10-20.*
9 *Goodrich, M.T., Tamassia, R., Goldwasser, M.H. (2013). Data Structures and Algorithms in Python, Wiley.*
10 *http://www.diveintopython.net/*
11 *https://docs.python.org/3/tutorial/*
12 *http://www.python-course.eu*
13 *https://developers.google.com/edu/python/*
14 *http://learnpythonthehardway.org/book/*