Programing in Python

Lecture 7b - Sets and Dictionary

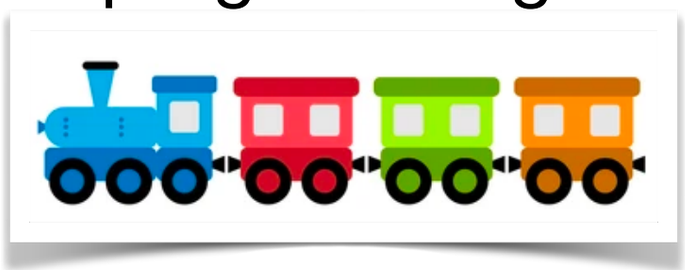
Instructor: Zhandos Yessenbayev

Outline

- Python Collections
- Python Sets
- Python Dictionaries
- Word Counts Exercise

Python Collections

- There are **four** collection data types in the Python programming language:



  - **List** is a collection which is *ordered* and *changeable* (with *duplicates*).

  - **Tuple** is a collection which is *ordered* and *unchangeable* (with *duplicates*).

  - **Set** is a collection which is *unordered*, *unchangeable* (with *no duplicates*).

  - **Dictionary** is a collection which is *ordered* and *changeable* (with *no duplicates*).

Python Sets

- **Set** is a collection which is *unordered*, *unchangeable* with *no duplicates*.
- We can initialize a set with some items or as empty **set( )**
- Items (elements) of a set can be of *any type*
- *Length* of a set can be found using **len( )** function

```
>>> set1 = {'Tom', 'Johns', 2 }
>>> len(set1)
3
>>> set2 = set()
>>> len(set2)
0
>>> set2
set()
```

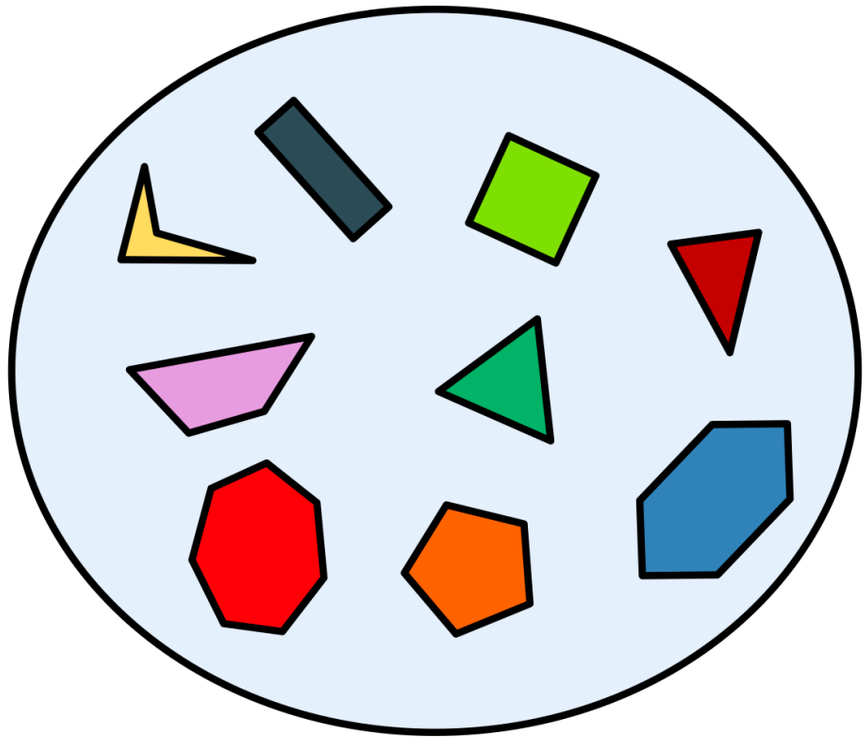
Accessing Set Items

- We cannot change items of the set (after creation and addition), but we can add and remove items

- We cannot access items by index, but
  we can loop through the set items using a **for** loop

```
>>> set1 = {1, 2, 3, 4}
>>>
>>> set1.add(5)
>>> set1.remove(1)
>>>
>>> set1.discard(1)
>>> set1.remove(1)
>>>
>>> for x in set1:
...     print(x)
...
>>> 4 in set1
```

Check membership



Set Methods

Some built-in set methods:

difference()	Returns a set containing the difference between two or more sets
intersection()	Returns a set, that is the intersection of two other sets
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not

Source: https://www.w3schools.com/python/python_sets_methods.asp

List Methods

```
>>> set1 = {0, 1, 2, 3}
>>> set2 = {-2, -1, 0, 1}
>>>
>>> set1.difference(set2)
>>>
>>> set1.intersection(set2)
>>>
>>> set3 = set1.union(set2)
>>> set4 = set1.update(set2)
>>>
>>> set1.isdisjoint(set2)
>>>
>>> set2.issubset(set1)
>>>
>>> set1.issuperset(set2)
>>>
```

Python Dictionary

- **Dictionary** is a collection which is *ordered* and *changeable* with *no duplicates*.

- Dictionaries are used to store data values in **key:value** pairs.

- *Length* of a dictionary can be found using **len( )** function

```
>>> dict1 = {'name':'Tom', 'age':25 }
>>> len(dict1)
2
>>> dict2 = dict()
>>> dict3 = {}
>>>
```

Accessing Dict Items

- We can access values by keys

```
>>> dict1 = {'name':'Tom', 'age':25 }
>>>
>>> dict1[‘name’]
>>> dict1[‘age’]
>>>
>>> dict1[‘age’] = 30
>>>
>>> dict1[‘city’] = ‘London’
>>>
>>> dict1.update({‘age’: 35})
>>>
>>> dict1.pop("name")
>>>
```

Accessing Dict Items

- We can get the lists of the *keys* with **keys( )** and *values* with **values( )** functions.

```
>>> car = {“brand": “Ford”, "year": 1964 }
>>>
>>> keys = car.keys()
>>>
>>> car[“model”] = “Mustang"
>>>
>>> vals = car.values()
>>>
>>> car[“year”] = 1970
>>>
>>> print(keys)
>>> print(vals)
```

Looping through Dict

- We can loop through dictionary with **keys**
- We also can use **tuple unpacking**

```
>>> counts = { 'tom' : 1 , 'bob' : 42, 'jan': 100}
>>>
>>> for key in counts:
...     print(key, counts[key])
...
>>>
>>> for (key, val) in enumerate(counts):
...   print(key, val)
...
>>>
```

Word Counts

- *Exercise:* **Find the counts of all the words in a file.**

    1. Create a new file *input.txt*

    2. Fill the file with some text

    3. Open the file from a program

    4. Read the file line by line

    5. Find the words in each line

    6. Populate a dictionary of the words with the counts

    7. Print the words with their counts row by row

Thanks!