Module 2:

Sets and Dictionaries

By Ninad Gaikwad

Reference - “Core Python Programming”
Dr. R. Nageshwara Rao
Dreamtech Press

Sets

- A set is a collection which is unordered, unchangeable, unindexed and do not allow duplicate values
- Create a Set: thisset = {"apple", "banana", "cherry"}
- A set can contain different data types
- The set() Constructor: thisset = set(("apple", "banana", "cherry"))
- Membership: print("banana" in thisset)
- Once a set is created, you cannot change its items, but you can add new items.
- The object in the update() method can be any iterable object (tuples, lists, dictionaries etc.)
- Eg.

```
a = set( [ a for a in "apple" ] )

b= { b for b in "banana" }

u = a.union(b)

print(u)
```

Set Methods with description

Method	**Description**
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)

Set Methods with description

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
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

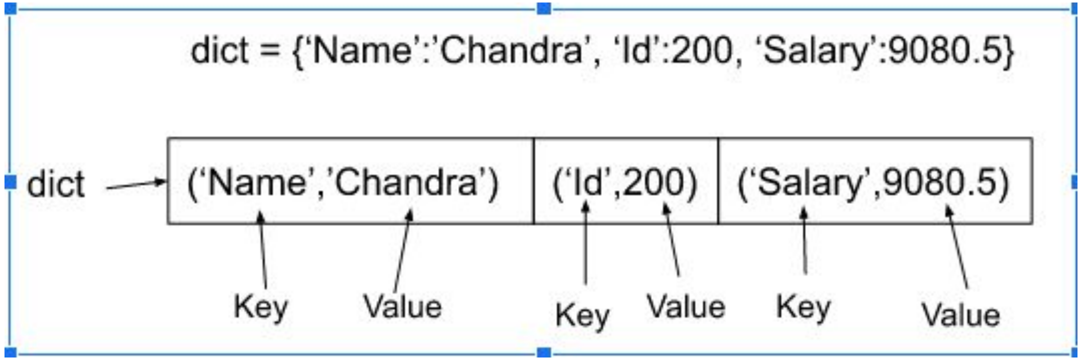
Experiment 5:

Q. Write a menu-driven program to demonstrate the use of set in python

a) Accept two strings from the user
b) Display Common Letters in Two Input Strings (Set Intersection)
c) Displays letters which are in the First String but not in the Second (Set Difference)
d) Displays set of all Letters of Both the Strings (Set Union)
e) Displays letters which are in the Two Strings but not in Both (Symmetric Difference)

Dictionaries

- Dictionaries represent a group of unordered elements arranged in the form of key-value pairs.



11.1 Operations on Dictionaries

- Membership in dictionary
    - 'Id' in dict
- Keys should be unique, duplicate entries are overwritten by latest value

Dictionaries

11.2 Dictionary methods

- d.clear(), d.copy, d.fromkeys(s [,v]), d.get(k [,v]), d.items(), d.keys(), d.values(), d.update(x), d.pop(k [,v]), d.setdefault(k [,v])
- dict.keys() - displays all keys in dictionary
- dict.values() - displays all values in dictionary
- dict.items() - displays key value pairs as tuples
- dict.update(k, v) - store a key value pair in dictionary
- dict.get(key, -1) - will return the value if key is found in the dictionary, else will return -1

11.3 Using for loop with dictionary

-
  ```
  for k in dict:
          print(k) # Prints all keys
  ```

-
  ```
  for k in dict:
          print(dict[k]) # print all values in dictionary
  ```

-
  ```
  for k, v in dict.items():
          print('key={}, val={}'.format(k, v)) # prints the keys and values
  ```

Dictionaries

11.4 Sorting dictionary elements using lambda

- Elements of dictionary can either be sorted based on keys or values
- Eg

```
colors = {10:‘Red’, 35:‘Green’, 15:’Blue’, 25:’White’}
sorted(colors.items(), key = lambda t: t[0])
# will sort based on the keys or t[0] element
sorted(colors.items(), key = lambda t: t[1])
# will sort based on the value or t[1] element
```

11.5 Convert to dictionary

1. List to dictionary
   - Convert the list to a zip class object and then zip into a dictionary
   - Eg

   ```
   countries = [“USA”, “India”, “Germany”, “France”]
   cities = [“washington”, “New Delhi”, “Berlin”, “Paris”]
   z= zip(countries, cities) # Convert to zip object
   d=dict(z) # convert to dictionary
   ```
2. Strings to dictionary
   - Extract the individual key value pairs and pass them in dictionary
   - d is a list containing all the key value pairs
   - d1={} # empty dictionary

   ```
   for k, v in d.items():
           d1[k] = int(v)
   ```

Dictionaries

11.6 Passing dictionaries to functions

- Accessing values of dictionary in function
  - Eg def fun(dict1):

```
                    for i, j in dict1.items():
                              print(i,"--", j)
```

- Passing dictionary to a function
  - Eg fun(d)

11.7 Ordered Dictionaries

- The elements are stored in the same order as they are entered into the dictionary
- Eg

```
from collections import OrderedDict
d= OrderedDict()
d[10] = 'A'
d[11] = 'B'
d[12] = 'C'
d[13] = 'D'

# Will print all elements in the order in which they are inserted
for i, j in d.items():
          print(i, j)
```

Experiment 6:

Q. Write a menu-driven program to demonstrate the use of a dictionary in python

a. Create a key/value pair dictionary
b. Update/concatenate and delete the item of the existing dictionary
c. Find a key and print its value
d. Map two list into a dictionary