Today Concepts

- Continue with dictionary
- set
- **order of data structures**

1. tuple
   - data structure that allows the user to store different data items at a time
   - immutable in nature(disadvantage)
2. list
   - list of students,marks list,list of items etc.
   - list allows the duplicate data.
   - can any 2 students have the same roll num/id?
3. set and
   - A set a well defined collection of objects
   - it avoids the duplication of values/data
   - it is mutable in nature but there is no index
   - set is represented by {}
   - set() is the predefined function
4. dictionary

In [1]:
```
S={}
type(S)
```
Out[1]:
```
dict
```

In [2]:
```
s=set()
type(s)
```
Out[2]:
```
set
```

In [3]:
```
A={9,4,5,6,10,23,45,1,9,8,7,10,34,56,23,12,4,5,10}
A
```
```
...
```

In [4]:
```
A[0]
```
```
...
```

In [5]:
```
#Set methods that updates the existed one
print(dir(A))
```
```
...
```

In [16]:
```
B={9,39,90,34,10,20,89,56,90,12,34,89}
B
```
Out[16]:
```
{9, 10, 12, 20, 34, 39, 56, 89, 90}
```

In [14]:
```
ord('a'),ord('X')
```
Out[14]:
```
(97, 88)
```

In [12]:
```
chr(90),chr(10)
```
Out[12]:
```
('Z', '\n')
```

In [17]:
```
A,B
```
Out[17]:
```
({1, 4, 5, 6, 7, 8, 9, 10, 12, 23, 34, 45, 56},
 {9, 10, 12, 20, 34, 39, 56, 89, 90})
```

In [18]:
```
A.intersection(B) # common elements from A&B
```
Out[18]:
```
{9, 10, 12, 34, 56}
```

In [19]:
```
A.union(B) # combination of A&B sets
```
Out[19]:
```
{1, 4, 5, 6, 7, 8, 9, 10, 12, 20, 23, 34, 39, 45, 56, 89, 90}
```

In [20]:
```
A.symmetric_difference(B) # non-similar elements from both sets
```
Out[20]:
```
{1, 4, 5, 6, 7, 8, 20, 23, 39, 45, 89, 90}
```

In [21]:
```
A-B # 9-4=5...?(deleting the values of B from A)
```
Out[21]:
```
{1, 4, 5, 6, 7, 8, 23, 45}
```

In [24]:
```
print(A)
A.difference(B) # 9-4=5
```
```
...
```

In [25]:
```
B.add(100)
```

In [26]:
```
B
```
```
...
```

In [27]:
```
B.add(50)
B
```
Out[27]:
```
{9, 10, 12, 20, 34, 39, 50, 56, 89, 90, 100}
```

In [29]:
```
B.discard(20)
```

In [30]:
```
B
```

...

In [32]:
```
B.remove(34)
B
```

...

In [33]:
```
B.pop()
B
```

...

In [34]:
```
print(dir(B))
```

...

In [35]:
```
B.update({5,6,10})
B
```

Out[35]:
```
{5, 6, 9, 10, 12, 39, 50, 56, 89, 90}
```

In [36]:
```
B.discard(1)
```

In [37]:
```
B
```

Out[37]:
```
{5, 6, 9, 10, 12, 39, 50, 56, 89, 90}
```

In [38]:
```
B.remove(1)
B
```

...

In [39]:
```
help(set.pop)
```

...

In [40]:
```
A
```

Out[40]:
```
{1, 4, 5, 6, 7, 8, 9, 10, 12, 23, 34, 45, 56}
```

In [41]:
```
A.pop()
```

Out[41]:
```
1
```

In [42]:
```
A.pop()
```
Out[42]:
```
34
```
In [43]:
```
A
```
Out[43]:
```
{4, 5, 6, 7, 8, 9, 10, 12, 23, 45, 56}
```
In [44]:
```
A.pop()
```
Out[44]:
```
4
```
In [45]:
```
A
```
Out[45]:
```
{5, 6, 7, 8, 9, 10, 12, 23, 45, 56}
```
In [46]:
```
A.pop()
```
Out[46]:
```
5
```
In [47]:
```
A.pop()
```
Out[47]:
```
6
```
In [48]:
```
A
```
Out[48]:
```
{7, 8, 9, 10, 12, 23, 45, 56}
```
In [49]:
```
A.pop()
```
Out[49]:
```
7
```
In [50]:
```
A
```
Out[50]:
```
{8, 9, 10, 12, 23, 45, 56}
```
In [52]:
```
A.isdisjoint(B) #not connected--> no common element
```
Out[52]:
```
False
```
In [53]:
```
A.issuperset(B)
```
Out[53]:
```
False
```

In [54]:
```
B.issubset(A)
```

Out[54]:
```
False
```

In [55]:
```
A.symmetric_difference_update(B) #non-common elements
A
```

...

In [59]:
```
A.intersection_update(B)
A
```

Out[59]:
```
{5, 6, 39, 50, 89, 90}
```

In [60]:
```
A
```

Out[60]:
```
{5, 6, 39, 50, 89, 90}
```

Dict

- It is a paired data structure containss key and value
- {key:value}
- dictionary methods
  - dict.keys()
    - all the keys from dictionary
  - dict.values()
    - all the values from dict

Files

-

In [61]:
```
# write a python program to prepare a dictionary of chars whose ascii valu
# string from the user/keyboard
```

In [62]:
```
dic={}
#{ch:ord(ch)}
for ch in input():
    if ord(ch)%2==0:
        dic[ch]=ord(ch)
print(dic)
```

```
I am Ruthu fROM APssDC
{' ': 32, 'R': 82, 't': 116, 'h': 104, 'f': 102, 'P': 80, 'D': 68}
```

In [63]:
```python
def create_dict(s):
    char_dict = {char: ord(char) for char in s if ord(char) % 2 == 0}
    return char_dict

s = input("Enter a string: ")
char_dict = create_dict(s)
print(char_dict)
*Output:*
Enter a string: HelloWorld
{'H': 72, 'l': 108, 'd': 100}

```

...

In [64]:
```python
chars={c:ord(c) for c in input() if ord(c)%2==0}
print(chars)
```

```
I am Ruthu FROM ApssDC
{' ': 32, 'R': 82, 't': 116, 'h': 104, 'F': 70, 'p': 112, 'D': 68}
```

In [ ]:
```python

```