ASSIGNMENT-2

How to access a particular element in set Difference b/w remove func and discard func What is intersection_update() What is symmetric_difference_update() What is symmetric_difference() What is dict() Use append() func by converting tuples into a list

Q1: How we can check or access one particular element from a set?

In [12]:
```
data = {"apple", "banana", "cherry",'guava','watermelon'}
for x in data: #Through this we can access all the element from the set
  print(x)
```

```
watermelon
guava
banana
apple
cherry
```

In [13]:
```
#It will check whether the element is present or not
print("banana" in data)
print("guava" in data)
```

```
True
True
```

Q2:Difference between remove function and discard function in a set.

```
remove()-It removes the given element from the set. If the element is not present in the set then it raises a KeyError.
Discard()-This function accepts an element as an argument and if that element exists in the set,
then it deletes that. Whereas, if the given element does not exist in the set, then discard() function does nothing.
So unlike remove() function it does not throw any error this is the main difference between the two.
```

In [14]:
```
print(data.remove("apple"))
```

```
None
```

In [15]:
```
print(data)
```

```
{'watermelon', 'guava', 'banana', 'cherry'}
```

In [18]:
```
print(data.remove("peach")) #it is showing key error
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_15140/3185285740.py in <module>
----> 1 print(data.remove("peach"))

KeyError: 'peach'
```

In [20]:
```
#Discard
print(data.discard('apple'))
print(data)
```

```
None
{'watermelon', 'guava', 'banana', 'cherry'}
```

In [22]:
```
print(data.discard('peach')) #we can see the difference between remove and discard from  the output
```

```
None
```

Q3:What is intersection_update()?

Python intersection_update() method is used to update a set with common elements only of all the sets passed in parameter of intersection_update() method.

In [24]:
```
set1 = {"java", "python", "c/cpp", "html"}
set2 = {"php", "html", "java", "R"}
set3 = {"java", "python", "ml", "dl"}
set4 = {"python", "java", "swift", "R"}
```

In [25]:
```
# perform intersection_update operation on set1
set1.intersection_update(set2, set3, set4)
# display the result set
print("After intersection_update, set1:", set1)
```

```
After intersection_update, set1: {'java'}
```

Q4:What is symmetric_difference_update()?

The symmetric difference of two sets is the set of elements which are in either of the sets but not in both of them. symmetric_difference() method returns a new set which contains symmetric difference of two sets. The symmetric_difference_update() method updates the set calling symmetric_difference_update()with the symmetric difference of sets.

In [26]:
```
A = {1,2,3,4,5,6,7}
B = {2,3,4,8,9}

# result is always none.
result = A.symmetric_difference_update(B)

print('A = ', A)
print('B = ', B)
print('result = ', result)
```

```
A =  {1, 5, 6, 7, 8, 9}
B =  {2, 3, 4, 8, 9}
result =  None
```

Q5:What is symmetric_difference()?

Python Set symmetric_difference() Method is used to get the elements present in either of the two sets, but not common to both the sets. If there are a set_A and set_B, then the symmetric difference between them will be equal to the union of set_A and set_B without the intersection between the two.

In [27]:
```
set_A = {1, 2, 3, 4, 5}
set_B = {6, 7, 3, 9, 4}
print(set_A.symmetric_difference(set_B))
```

```
{1, 2, 5, 6, 7, 9}
```

Q6: What is dict()?

Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered*, changeable and do not allow duplicates.

In [29]:
```
thisdict = {
"Name": "Jaislin",
"ID": "1234",
"Age": 21
}
print(thisdict)
```

```
{'Name': 'Jaislin', 'ID': '1234', 'Age': 21}
```

In [30]:
```
print(len(thisdict))
```

```
3
```

Q7:Use append() func by converting tuples into a list

In [36]:
```
x= ("Jaislin","Taniya", "Indus", "Pratham")
print(type(x))
print(x)
# converting tuple into a list.
y = list(x)
print(type(y))

y.append("Manjit")  # using append() on list.
```

```
<class 'tuple'>
('Jaislin', 'Taniya', 'Indus', 'Pratham')
<class 'list'>
```

In [37]:
```
x = tuple(y)  # converting list back to tuple.
print('New Tuple formed:',x)
print(type(x))
```

```
New Tuple formed: ('Jaislin', 'Taniya', 'Indus', 'Pratham', 'Manjit')
<class 'tuple'>
```

In [ ]:
```

```