In [1]:
```
!pip install nbconvert[webpdf]
```

```
Requirement already satisfied: nbconvert[webpdf] in c:\users\jayanth\anaconda3\envs\dask
env\lib\site-packages (6.5.4)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\jayanth\anaconda3\envs\dask
env\lib\site-packages (from nbconvert[webpdf]) (0.8.4)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\jayanth\anaconda3\envs\dask
env\lib\site-packages (from nbconvert[webpdf]) (5.3.0)
Requirement already satisfied: lxml in c:\users\jayanth\anaconda3\envs\daskenv\lib\site-
packages (from nbconvert[webpdf]) (4.9.1)
Requirement already satisfied: pygments>=2.4.1 in c:\users\jayanth\anaconda3\envs\dasken
v\lib\site-packages (from nbconvert[webpdf]) (2.11.2)
Requirement already satisfied: beautifulsoup4 in c:\users\jayanth\anaconda3\envs\daskenv
\lib\site-packages (from nbconvert[webpdf]) (4.11.1)
Requirement already satisfied: nbformat>=5.1 in c:\users\jayanth\anaconda3\envs\daskenv
\lib\site-packages (from nbconvert[webpdf]) (5.7.0)
Requirement already satisfied: defusedxml in c:\users\jayanth\anaconda3\envs\daskenv\lib
\site-packages (from nbconvert[webpdf]) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in c:\users\jayanth\anaconda3\envs\daskenv\li
b\site-packages (from nbconvert[webpdf]) (3.1.2)
Requirement already satisfied: packaging in c:\users\jayanth\anaconda3\envs\daskenv\lib
\site-packages (from nbconvert[webpdf]) (23.0)
Requirement already satisfied: tinycss2 in c:\users\jayanth\anaconda3\envs\daskenv\lib\s
ite-packages (from nbconvert[webpdf]) (1.2.1)
Requirement already satisfied: traitlets>=5.0 in c:\users\jayanth\anaconda3\envs\daskenv
\lib\site-packages (from nbconvert[webpdf]) (5.7.1)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\jayanth\anaconda3\envs\dasken
v\lib\site-packages (from nbconvert[webpdf]) (0.5.13)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\jayanth\anaconda3\envs\dasken
v\lib\site-packages (from nbconvert[webpdf]) (2.1.1)
Requirement already satisfied: jupyterlab-pygments in c:\users\jayanth\anaconda3\envs\da
skenv\lib\site-packages (from nbconvert[webpdf]) (0.1.2)
Requirement already satisfied: bleach in c:\users\jayanth\anaconda3\envs\daskenv\lib\sit
e-packages (from nbconvert[webpdf]) (4.1.0)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\jayanth\anaconda3\envs\das
kenv\lib\site-packages (from nbconvert[webpdf]) (0.4)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\jayanth\anaconda3\envs\d
askenv\lib\site-packages (from nbconvert[webpdf]) (1.5.0)
Collecting pyppeteer<1.1,>=1
  Downloading pyppeteer-1.0.2-py3-none-any.whl (83 kB)
     ---------------------------------------- 0.0/83.4 kB ? eta -:--:--
     ------------------ ------------------- 41.0/83.4 kB 991.0 kB/s eta 0:00:01
     -------------------------------------- 83.4/83.4 kB 937.7 kB/s eta 0:00:00
Requirement already satisfied: platformdirs>=2.5 in c:\users\jayanth\anaconda3\envs\dask
env\lib\site-packages (from jupyter-core>=4.7->nbconvert[webpdf]) (2.5.2)
Requirement already satisfied: pywin32>=300 in c:\users\jayanth\anaconda3\envs\daskenv\l
ib\site-packages (from jupyter-core>=4.7->nbconvert[webpdf]) (305.1)
Requirement already satisfied: nest-asyncio in c:\users\jayanth\anaconda3\envs\daskenv\l
ib\site-packages (from nbclient>=0.5.0->nbconvert[webpdf]) (1.5.6)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\jayanth\anaconda3\envs
\daskenv\lib\site-packages (from nbclient>=0.5.0->nbconvert[webpdf]) (7.4.9)
Requirement already satisfied: jsonschema>=2.6 in c:\users\jayanth\anaconda3\envs\dasken
v\lib\site-packages (from nbformat>=5.1->nbconvert[webpdf]) (4.17.3)
Requirement already satisfied: fastjsonschema in c:\users\jayanth\anaconda3\envs\daskenv
\lib\site-packages (from nbformat>=5.1->nbconvert[webpdf]) (2.16.2)
Collecting pyee<9.0.0,>=8.1.0
  Downloading pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\jayanth\anaconda3\envs\da
skenv\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (4.65.0)
Collecting websockets<11.0,>=10.0
  Downloading websockets-10.4-cp310-cp310-win_amd64.whl (101 kB)
     ---------------------------------------- 0.0/101.4 kB ? eta -:--:--
     -------------------------------------- 101.4/101.4 kB 2.9 MB/s eta 0:00:00
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\jayanth\anaconda3\envs
```

```
\daskenv\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (1.4.4)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\jayanth\anaconda3\env
s\daskenv\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (6.0.0)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\jayanth\anaconda3\envs
\daskenv\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (1.26.15)
Requirement already satisfied: certifi>=2021 in c:\users\jayanth\anaconda3\envs\daskenv
\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (2022.12.7)
Requirement already satisfied: soupsieve>1.2 in c:\users\jayanth\anaconda3\envs\daskenv
\lib\site-packages (from beautifulsoup4->nbconvert[webpdf]) (2.3.2.post1)
Requirement already satisfied: six>=1.9.0 in c:\users\jayanth\anaconda3\envs\daskenv\lib
\site-packages (from bleach->nbconvert[webpdf]) (1.16.0)
Requirement already satisfied: webencodings in c:\users\jayanth\anaconda3\envs\daskenv\l
ib\site-packages (from bleach->nbconvert[webpdf]) (0.5.1)
Requirement already satisfied: zipp>=0.5 in c:\users\jayanth\anaconda3\envs\daskenv\lib
\site-packages (from importlib-metadata>=1.4->pyppeteer<1.1,>=1->nbconvert[webpdf]) (3.1
1.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\jayanth\anaconda3\envs\daskenv
\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert[webpdf]) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users
\jayanth\anaconda3\envs\daskenv\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1->
nbconvert[webpdf]) (0.18.0)
Requirement already satisfied: tornado>=6.2 in c:\users\jayanth\anaconda3\envs\daskenv\l
ib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert[webpdf]) (6.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\jayanth\anaconda3\envs
\daskenv\lib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert[webpd
f]) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in c:\users\jayanth\anaconda3\envs\daskenv\li
b\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert[webpdf]) (23.2.
0)
Requirement already satisfied: colorama in c:\users\jayanth\anaconda3\envs\daskenv\lib\s
ite-packages (from tqdm<5.0.0,>=4.42.1->pyppeteer<1.1,>=1->nbconvert[webpdf]) (0.4.6)
Installing collected packages: pyee, websockets, pyppeteer
Successfully installed pyee-8.2.2 pyppeteer-1.0.2 websockets-10.4
```

Python-Lesson 03

In this notebook, we shall cover the following concepts

- Bitwise operators
- Membership operators
- Identity Operators
- Conditional Statements
- Python Data Structures
  - List
  - Tuple
  - Sets
  - Dictionary
  - Collections in Python

- Bitwise Operators
- Membership Operators
- Identity Operators

- Bitwise Operators

- Bitwise AND (&) Operator

In [6]:
```
print(bin(3))
print(bin(5))  # 011 & 101 --> 1
```
```
0b11
0b101
```

In [7]:
```
3&5
```
Out[7]:
```
1
```

- Bitwise OR (|) Operator

In [ ]:
```
# 0011 | 0101 -> 0111
```

In [9]:
```
print(3|5)
```
```
7
```

- Bitwise XOR (^) Operator

In [11]:
```
# 1011 ^ 1100 -> 0111 (11^12 -> 7)
print(11^12)
```
```
7
```

- Bitwise << left shift operator - The left shift operator shifts the left bits operand towards left.
- 1100 << 2 means that the two zeros are appended at the right side. 110000 is the output

In [14]:
```
### input is 1100 --> 12 << 2 output is 110000 (48) is the output
12 << 2
```
Out[14]:
```
48
```

- Bitwise >> right shift operator - The right shift operator shifts the right side bits. The rigt side bit are removed.
- 110011 >> 2 means that the last two bits are removed from the right side. 1100 is the output.

In [21]:
```
# 51 -> in binary is 110011 and when last two are removed we get 1100 (12) as the output
51 >> 2
```
Out[21]:
```
12
```

- Bitwise one's complement operator (~) x -> -(x+1) . This gives the complement of the binary number
- ~10 (1010) -> -11

In [22]:
```
~10
```
Out[22]:
```
-11
```

In [24]:
```
~-10
```
Out[24]:
```
9
```

In [26]:
```
#### Exercise - check if a number is even using bitwise & operator
x = 11
x&1 == 1  # odd number
```
Out[26]:
```
True
```
In [28]:
```
y = 12
y&1 == 0 # even number
```
Out[28]:
```
True
```
- Membership Operators - in and not in

In [35]:
```
### We can check the if the values are available or not in the given string for operator
## a='hello' in b='hello can i know the time please ?' check a in b
a = 'hello'
b = 'hello can i know the time please ?'
a in b
```
Out[35]:
```
True
```
In [33]:
```
### We can check the if the values are available or not in the given string for operator
## a='bell' in b='hello can i know the time please ?' check a not in b
a = 'bell'
b = 'hello can i know the time please ?'
a not in b
```
Out[33]:
```
True
```
In [34]:
```
### We can check the if the values are available or not in the given string for operator
## a='bell' in b='hello can i know the time please ?' check a  in b
a = 'bell'
b = 'hello can i know the time please ?'
a in b
```
Out[34]:
```
False
```
- Identity Operators - is and is not

In [36]:
```
### is or is not operators can be used to know if certain conditions are met are not
a = 21
b = 33
c = b
print(c is b)
```
Out[36]:
```
True
```
In [37]:
```
print(c is not b)
```
```
False
```
Python Conditional Statements

if expr: </br> statement </br> elif expr: </br> statement </br> elif expr: </br> statement </br> elif expr: </br> statement </br> : </br> : </br> : </br> else:
statement

In [58]:
```
## Simple condition for two numbers
```

```
x = 10
y = 20
if y > x:
    print('Y is greater')
else:
    print('X is greater')
```

```
Y is greater
```

In [43]:

```
### Enter a name to Capture input from the command line and write conditions for it.
name = input('Enter name ')
if name == 'Jayant' :
    print('Please start')
elif name == 'Vamshi':
    print('Please turn around')
elif name == 'Mayank':
    print('Please be seated')
else:
    print('Not in the name')
```

```
Enter name Hari
Not in the name
```

In [45]:

```
### Writing a if else block of code to capture the statements such that
## we print statements only for age group of 13
age = input('Enter age of participant')
age = int(age)
if age >= 13:
    print('You can start coding')
else:
    print('Please learn basics of scartch')
```

```
Enter age of participant15
You can start coding
```

Hacker Rank Questions

### Hacker Rank Question Given an integer, perform the following conditional actions: If n is odd, print Weird If n is even and in the inclusive range of 2 to 5, print Not Weird If n is even and in the inclusive range of to 6 to 20, print Weird If n is even and greater than 20 , print Not Weird n >= 1 and n <= 100

In [57]:

```
## Summary of above n --> weird ( odd , even and 6 to 20) Not weird (even and 2to 5 and
n = input('Enter a number 1 to 100 ')
n = int(n)
if n % 2 != 0 and n <= 20:
    print('Weird')
elif n%2 == 0 and 2<=n <= 5:
    print('Not weird')
elif n%2 == 0 and 6<=n<=20:
    print('Weird')
else:
    print('Not weird')
```

```
Enter a number 1 to 100 4
Not weird
```

In [ ]:

```

```

Loops in Python

- while loop in python

In [35]:

```
### while loops in python
x = 0
```

```
while x < 7:
    print(x)
    x += 1
```

```
0
1
2
3
4
5
6
```

In [36]:
```
### while loops in python
x = 5
while x > 0:
    print(x)
    x -= 1
```

```
5
4
3
2
1
```

- for loop in python

- Range Object - A Range

If we want to generate numbers from 0 to n - then we can use range()

- Generate numbers using range object
- range()- A range object generates a sequence of numbers starting from 0 to n

In [41]:
```
#for loops in python using range object - Starting from 0 and ending until 3
for i in range(3):
    print(i)
```

```
0
1
2
```

In [42]:
```
# for loops in python using range object - Starting from a number 2 and ending until 10.
for i in range(2,10):
    print(i)
```

```
2
3
4
5
6
7
8
9
```

In [43]:
```
# for loops in python using range object - Staring from a number and ending at a number
for i in range(2,14,2):
    print(i)
```

```
2
4
6
8
```

```
10
12
```

Python Data Structures

- List
- Tuple
- Sets
- Dictioary
- Collections in Python

- List - List of comma separated values (items) between square brackets </br>
  ```
  - Lists can be heterogenous - can take different values. </br>
  - Lists can be accessed based on indices. </br>
  ```

</br> | 0 | 1 | 2 | 3 | 4 | </br> | 1 | 2 | 3 | 4 | 5 | </br> | -5 | -4 | -3 | -2 | -1 | </br>

In [10]:
```python
### Creating an empty list - []
numval = []
```

In [1]:
```python
numbers = [1,2,3,4,5]
numbers
```

Out[1]:
```
[1, 2, 3, 4, 5]
```

In [2]:
```python
#### finding the numbers with zeroth index - numbers[0]
numbers[0]
```

Out[2]:
```
1
```

In [4]:
```python
#### Finding the numbers from the last - numbers[-1]
numbers[-1]
```

Out[4]:
```
5
```

In [5]:
```python
#### finding the number from first (0) to last- numbers[0:]
numbers[0:]
```

Out[5]:
```
[1, 2, 3, 4, 5]
```

In [6]:
```python
#### finding the numbers from first index (1) to (4) index - numbers[1:4]
numbers[1:4]
```

Out[6]:
```
[2, 3, 4]
```

In [7]:
```python
#### finding the numbers from second index (2) to (5) index - numbers[2:5]
numbers[2:5]
```

Out[7]:
```
[3, 4, 5]
```

In [8]:
```python
#### finding the numbers from last index -3 to to the zeroth index -numbers[-3:]
numbers[-3:]
```

Out[8]:
```
[3, 4, 5]
```

- Print all numbers

```
In [9]:   ### listing all the numbers from the list - numbers[:]
          numbers[:]

Out[9]:   [1, 2, 3, 4, 5]
```

- print length of numbers

```
In [11]:  ### length of the list can be found using - len(numbers)
          len(numbers)

Out[11]:  5
```

- Assign value to particular index

```
In [14]:  #### We can reassign value to a index in a list - assign a new number to index 0
          numbers[0] = 11

In [15]:  numbers

Out[15]:  [11, 2, 3, 4, 5]
```

- append() elements to a list

```
In [16]:  #### List values can be appended at the end in a list - numbers.append() - numbers.appen

In [17]:  #### List of number after the append - numbers

Out[17]:  [11, 2, 3, 4, 5, 6]
```

- To check the type of the list

```
In [18]:  #### Check type of list
          type(numbers)

Out[18]:  list
```

- Elements of list is heterogenous

```
In [19]:  #### Different types of values an be appended to a list
          listValues = ['Apple','Banana',23,24,25,26]
          listValues

Out[19]:  ['Apple', 'Banana', 23, 24, 25, 26]
```

- Append elements to a list using append()

```
In [20]:  #### List can also be appended with 'Mango' at the list
          listValues.append('Mango')

In [21]:  #### list value can also be appended with [33,45,56]
          listValues.append([33,45,56])
```

```
In [22]: listValues
```

```
Out[22]: ['Apple', 'Banana', 23, 24, 25, 26, 'Mango', [33, 45, 56]]
```

- Create list using the timing values - ['1:30','2:30','3:30','4:30','5:30','6:30','7:30']

```
In [30]: ### timings values are like this - timings = ['1:30','2:30','3:30','4:30','5:30','6:30',
         timings = ['1:30','2:30','3:30','4:30','5:30','6:30','7:30']
         timings[0:3]
```

```
Out[30]: ['1:30', '2:30', '3:30']
```

```
In [31]: ## Slice the list value from 2:5
         timings[2:5]
```

```
Out[31]: ['3:30', '4:30', '5:30']
```

```
In [32]: ## Slice the list value from 4:
         timings[4:]
```

```
Out[32]: ['5:30', '6:30', '7:30']
```

- reverse() a list

```
In [33]: ### To find the reverse of the list - list[::-1]
         timings[::-1]
```

```
Out[33]: ['7:30', '6:30', '5:30', '4:30', '3:30', '2:30', '1:30']
```

```
In [46]: ### Alternatively to reverse the list- use the timings.reverse()
         timings.reverse()
         timings
```

```
Out[46]: ['1:30', '2:30', '3:30', '4:30', '5:30', '6:30', '7:30']
```

- index of a value in the list

```
In [47]: #### To get the index of the given element - timings.index('2:30')
         timings.index('2:30')
```

```
Out[47]: 1
```

- insert an element at a particular index

```
In [49]: ### Insert the value 8 at the index 2 - timings.insert(2,8)
```

```
In [50]: 
```

```
Out[50]: ['1:30', '2:30', 8, '3:30', '4:30', '5:30', '6:30', '7:30']
```

- remove a value from the list

```
In [51]: ### To remove an element from the list we can do this - timings
         timings.remove(8)
```

```
timings
```

Out[51]:
```
['1:30', '2:30', '3:30', '4:30', '5:30', '6:30', '7:30']
```

- pop an element from the list

In [52]:
```
### to remove the top most element from the list - timings - timings.pop()
timings.pop()
```

Out[52]:
```
'7:30'
```

- sort() an element from the list

In [56]:
```
### using the sort() list can be sorted - timings.sort()
timings.sort()
```

In [57]:
```
timings
```

Out[57]:
```
['1:30', '2:30', '3:30', '4:30', '5:30', '6:30']
```

- extend an element from the list

In [58]:
```
## Extend the list by adding the elements - Elements shall be appended at the end of the
## extend - ['7:30','8:30','9:30']
timings.extend(['7:30','8:30','9:30'])
```

In [59]:
```
timings
```

Out[59]:
```
['1:30', '2:30', '3:30', '4:30', '5:30', '6:30', '7:30', '8:30', '9:30']
```

- Iterate a list

In [61]:
```
for timing in timings:
    print("The time now is {}".format(timing))
```

```
The time now is 1:30
The time now is 2:30
The time now is 3:30
The time now is 4:30
The time now is 5:30
The time now is 6:30
The time now is 7:30
The time now is 8:30
The time now is 9:30
```

- sum of elements in a list

In [3]:
```
### Lets build a list using 10,11,12,13,14,15,16 - The sum of all the elements in the li
### in two ways  - sum(intList) or using for loops

print('Sum of the list of elements',sum(intList))
print(intList)
```

```
Sum of the list of elements 91
[10, 11, 12, 13, 14, 15, 16]
```

- sum of the elements using for loops

```
In [4]:  intList = 10,11,12,13,14,15,16
         slist = 0
         for i in intList:
             slist += i
         print(slist)

         91
```

- for comprehensions

For comprehension is a simple way of returning the value based on certain input

```
In [62]:  ## Based on the range object lets find the square of each number
          y = [x**2 for x in range(5)]

In [63]:  y

Out[63]:  [0, 1, 4, 9, 16]

In [65]:  ## We can also include condition within the for comrephension
          y = [x**2 for x in range(10) if x%2 == 0 ]
          print(y)

          [0, 4, 16, 36, 64]

In [66]:  ### return if only list contains letters starting with s
          names = ['sunday','super','semi-conductor','non-conductor','mango','apple']
          y = [x for x in names if x.startswith('s')]

In [67]:  y

Out[67]:  ['sunday', 'super', 'semi-conductor']
```

Tuples

- Tuples are immutable objects that can be represented in ()
- Tuples can be accessed using indices
- Tuples are created by passing vlues to ()

```
In [76]:  tup1 = (25,'Mango','Rajesh',40,25,24,25)
          tup1[1]

Out[76]:  'Mango'

In [77]:  tup1[3]

Out[77]:  40

In [78]:  ### Tuples can be iterated over using for loop
          for x in tup1:
              print(x)

          25
          Mango
          Rajesh
          40
          25
```

```
24
25
```

In [75]:
```
## Assigning a value to tuple shows error - tuple object does not support assignment
tup1[0] = 35
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[75], line 2
      1 ## Assigning a value to tuple shows error - tuple object does not support assign
ment
----> 2 tup1[0] = 35

TypeError: 'tuple' object does not support item assignment
```

In [79]:
```
## Count function gives the count of the element in the tuple
tup1.count(25)
```

Out[79]:
```
3
```

In [68]:
```
### Exercise - There are two list [2,3,4] and [2,5,6] -Get a combination (x,y)
## x is the element from first list,
## y is the element from second list - Get combinations (x,y) such that x is not equal t

x = [2,3,4]
y = [2,5,6]
output = [(a,b) for a in x for b in y if x!=y]
print(output)
```

```
[(2, 2), (2, 5), (2, 6), (3, 2), (3, 5), (3, 6), (4, 2), (4, 5), (4, 6)]
```

- **Sets in Python**
  - Sets contain unique elements
  - Sets are unordered sequence of elements
  - They are represented using the {} (curly braces)

In [80]:
```
elements = {'Mahesh','Mahesh','Suraj','Ravi','Kiran','Surya'}
elements
```

Out[80]:
```
{'Kiran', 'Mahesh', 'Ravi', 'Suraj', 'Surya'}
```

In [6]:
```
#### Sets can be constructed using the following set keyword and passing a list to it
eles = set(['Mahesh','Mahesh','Suraj','Ravi','Kiran','Surya'])
eles
```

Out[6]:
```
{'Kiran', 'Mahesh', 'Ravi', 'Suraj', 'Surya'}
```

- Iteration over sets in python

In [83]:
```
elements
```

Out[83]:
```
{'Kiran', 'Mahesh', 'Ravi', 'Suraj', 'Surya'}
```

In [84]:
```
for x in elements:
    print(x)
```

```
Suraj
Surya
Ravi
```

```
Mahesh
Kiran
```

In [86]:
```
## The set elements are given below - We can represent the data as els = {'Mahesh','Mahe
els = {'Mahesh','Mahesh','Suraj','Ravi','Kiran','Surya'}
len(els)
```

Out[86]:
```
5
```

- Adding elements to set

In [87]:
```
### Add the following element to the set elements 'Mitra' using add function
els.add('Mitra')
```

In [88]:
```
els
```

Out[88]:
```
{'Kiran', 'Mahesh', 'Mitra', 'Ravi', 'Suraj', 'Surya'}
```

In [91]:
```
## Lets create one more set els2 = els2 = {'kiran','Mahesh', 'Mitra'}
els2 = {'kiran','Mahesh', 'Mitra'}
```

- seta.difference(setb) - Difference gives the elements that are in seta

In [93]:
```
els.difference(els2)
```

Out[93]:
```
{'Kiran', 'Ravi', 'Suraj', 'Surya'}
```

- difference_update() - Removes the elements from the other set

In [98]:
```
els.difference_update(els2)
```

In [99]:
```
els
```

Out[99]:
```
{'Kiran', 'Ravi', 'Suraj', 'Surya'}
```

In [100…
```
els2
```

Out[100]:
```
{'Mahesh', 'Mitra', 'kiran'}
```

In [101…
```
## Create the set and try to perform union on els and els2
els = {'Mahesh','Mahesh','Suraj','Ravi','Kiran','Surya'}
len(els)
```

Out[101]:
```
5
```

- union combines both the sets with unique elements

In [103…
```
els.union(els2)
```

Out[103]:
```
{'Kiran', 'Mahesh', 'Mitra', 'Ravi', 'Suraj', 'Surya', 'kiran'}
```

- intersection provides the common elements between the sets

In [105…
```
els.intersection(els2)
```

```
Out[105]: {'Mahesh'}
```

```
In [106…: print(els)
          print(els2)
```

```
{'Suraj', 'Surya', 'Ravi', 'Mahesh', 'Kiran'}
{'Mitra', 'Mahesh', 'kiran'}
```

- Symmetric difference gives the elements that are not common as one single set.

```
In [108…: ## els.symmetric_difference(els2) gives the common elements between both the sets.
          els.symmetric_difference(els2)
```

```
Out[108]: {'Kiran', 'Mitra', 'Ravi', 'Suraj', 'Surya', 'kiran'}
```

- Comprehension for set - Similar to for comprehension for Set
  - return type is Set when set comprehension is perfromed

```
In [8]: listVal = {'Mahesh','Mahesh','Suraj','Ravi','Kiran','Surya'}
        uniEle = {x for x in listVal}
        uniEle
```

```
Out[8]: {'Kiran', 'Mahesh', 'Ravi', 'Suraj', 'Surya'}
```

- **Dictionary in Python** - Dictionaries in Python are basically key,value pairs.
  - In a dictionary, key can be any unique object
  - Keys are unique and non-duplicated
  - Values can be duplicated
  - Dictionaries can be constructed using the dict() or {:}

- Create Dictionary using the dict() keyword

```
In [13]: idName = dict()
         idName[101] = 'Suraj'
         idName[102] = 'Mahesh'
         idName[103] = 'Surya'
         idName[104] = 'Ravi'
         idName[105] = 'Kiran'
         idName[106] = 'Mitra'
         idName
```

```
Out[13]: {101: 'Suraj',
          102: 'Mahesh',
          103: 'Surya',
          104: 'Ravi',
          105: 'Kiran',
          106: 'Mitra'}
```

```
In [37]: ### Another way of creating the dictionary is the following
         idName = {101: 'Suraj',102: 'Mahesh',103: 'Surya', 104: 'Ravi', 105: 'Kiran', 106: 'Mitr
         idName
```

```
Out[37]: {101: 'Suraj',
          102: 'Mahesh',
          103: 'Surya',
          104: 'Ravi',
          105: 'Kiran',
          106: 'Mitra'}
```

- Retrieve the value of dictionary using key

In [17]:
```
### The dictionary name and the id can be passed to get the value - idName[101]
```

- Check if the key is in the dictionary

In [21]:
```
### Check if a particular key is available in the dictionary - 101 in idName
101 in idName
```

Out[21]:
```
True
```

In [24]:
```
### CHeck if a particular key is not available in the dictionary - 109 not in idName
print(109 not in idName)
print(110 in idName)
```

```
True
False
```

- iterate over the dictionary using keys()

In [34]:
```
### We can iterate over the dictionary using the keys() function on the dictionary.
for k in idName.keys():
    print(f'Key is {k}')
```

```
Key is 101
Key is 102
Key is 103
Key is 104
Key is 105
Key is 106
```

- Values for the dictionary using values()

In [36]:
```
for v in idName.values():
    print(f'value is {v}')
```

```
value is Suraj
value is Mahesh
value is Surya
value is Ravi
value is Kiran
value is Mitra
```

- iterate over a dictionary using items()

In [33]:
```
for key,value in idName.items():
    print(f'Id is {key}, Name is {value}')
```

```
Id is 101, Name is Suraj
Id is 102, Name is Mahesh
Id is 103, Name is Surya
Id is 104, Name is Ravi
Id is 105, Name is Kiran
Id is 106, Name is Mitra
```

- Iterate over a dictionary using the dictionary object

```
In [39]: ### Using the dictionary object only the dictionary is used
         for keyVal in idName:
             print(keyVal)
```

```
101
102
103
104
105
106
```

- Delete a key in the dictionary using the del keyword

```
In [40]: ## Delete the key value pair using the idName[key]
         del idName[101]
```

```
In [41]: idName
```

```
Out[41]: {102: 'Mahesh', 103: 'Surya', 104: 'Ravi', 105: 'Kiran', 106: 'Mitra'}
```

- To get a value from the dictionar using the get() function or the []

```
In [42]: idName[102]
```

```
Out[42]: 'Mahesh'
```

```
In [43]: idName.get(102)
```

```
Out[43]: 'Mahesh'
```

- If the key is not in the dictionary using the get() method we can use the default value

```
In [49]: ### Lets say we dont have the key/value pair in the dictionary it returns a key erro
         idName[109]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[49], line 2
      1 ### Lets say we dont have the key/value pair in the dictionary
----> 2 idName[109]

KeyError: 109
```

```
In [51]: # If the key is available then it returns the value
         idName.get(102,"Value is Not there")
```

```
Out[51]: 'Mahesh'
```

```
In [46]: idName.get(109,"Key/Value is Not there")
```

```
Out[46]: 'Key/Value is Not there'
```

```
In [55]: ### To get the keys in the reversed order we can use reversed on the dictionary
```

```
In [54]: for i in reversed(idName):
    print(i)
```

```
106
105
104
103
102
```

In [58]:
```
## Collecting the dictionary keys as list - list(idName)
list(idName)
```

Out[58]:
```
[102, 103, 104, 105, 106]
```

- To remove one key,value pair from the dictionary - using popitem()

In [59]:
```
## idName.popitem() will remove one key value pair from the dictionary
idName.popitem()
```

Out[59]:
```
(106, 'Mitra')
```

- To remove one key at a time from the dictionary - using pop()

In [64]:
```
### ## idName.pop(key) will pop one key value pair for the given key from the dictionary
idName.pop(104)
```

Out[64]:
```
'Ravi'
```

In [65]:
```
### If the key is not available in the dictionary, pop() shall given an error on the dic
idName.pop(105)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[65], line 2
      1 ### If the key is not available in the dictionary, pop() shall given an error on
 the dictionary
----> 2 idName.pop(105)

KeyError: 105
```

- length of the dictionary - len() function

In [67]:
```
# To get the length of the dictionary
len(idName)
```

Out[67]:
```
2
```

- Lets say we want to get the key,value pairs using the dictionary using dictionary comprehension

In [71]:
```
### Dictionary comprehension can be perfromed using the following:
idName = idName = {101: 'Suraj',102: 'Mahesh',103: 'Surya', 104: 'Ravi', 105: 'Kiran', 1

# get only the even ids

evenIdName = {k:v for (k,v) in idName.items() if k%2 ==0 }
evenIdName
```

Out[71]:
```
{102: 'Mahesh', 104: 'Ravi', 106: 'Mitra'}
```

```
In [ ]:
```