PYTHON PROGRAMMING LAB

LAB ASSIGNMENT FILE

SUBMITTED TO:

MR AKSHAT AGRAWAL
AMITY SCHOOL OF
ENGINEERING &
TECHNOLOGY

SUBMITTED BY:

HARSH - A501144824003
M.TECH(AI)
M.TECH 2024-2026

INDEX

Experiment 1

Aim: Introduction to Python and its data types

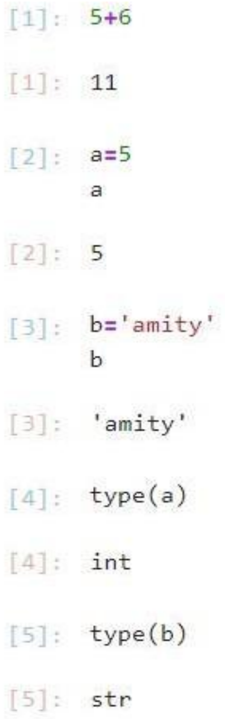
Theory:

Python- It is an interpreted, object-oriented, high-level programming language with dynamic semantics.

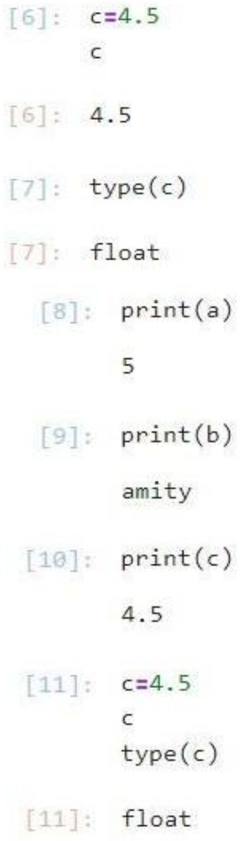
Data types:

1. Int
2. Float
3. String
4. Boolean
5. Complex
6. List
7. Tuple
8. Set
9. Dictionary
10. None type

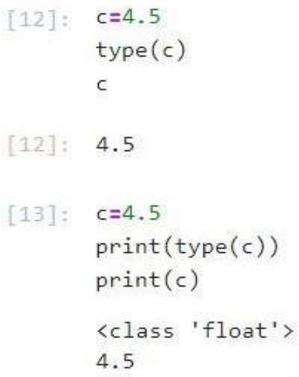
Source code/Output:



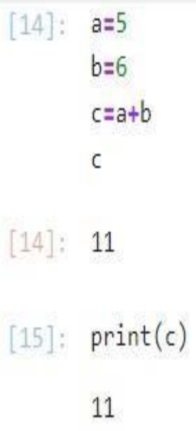
```
[1]:  5+6

[1]:  11

[2]:  a=5
      a

[2]:  5

[3]:  b='amity'
      b

[3]:  'amity'

[4]:  type(a)

[4]:  int

[5]:  type(b)

[5]:  str
```



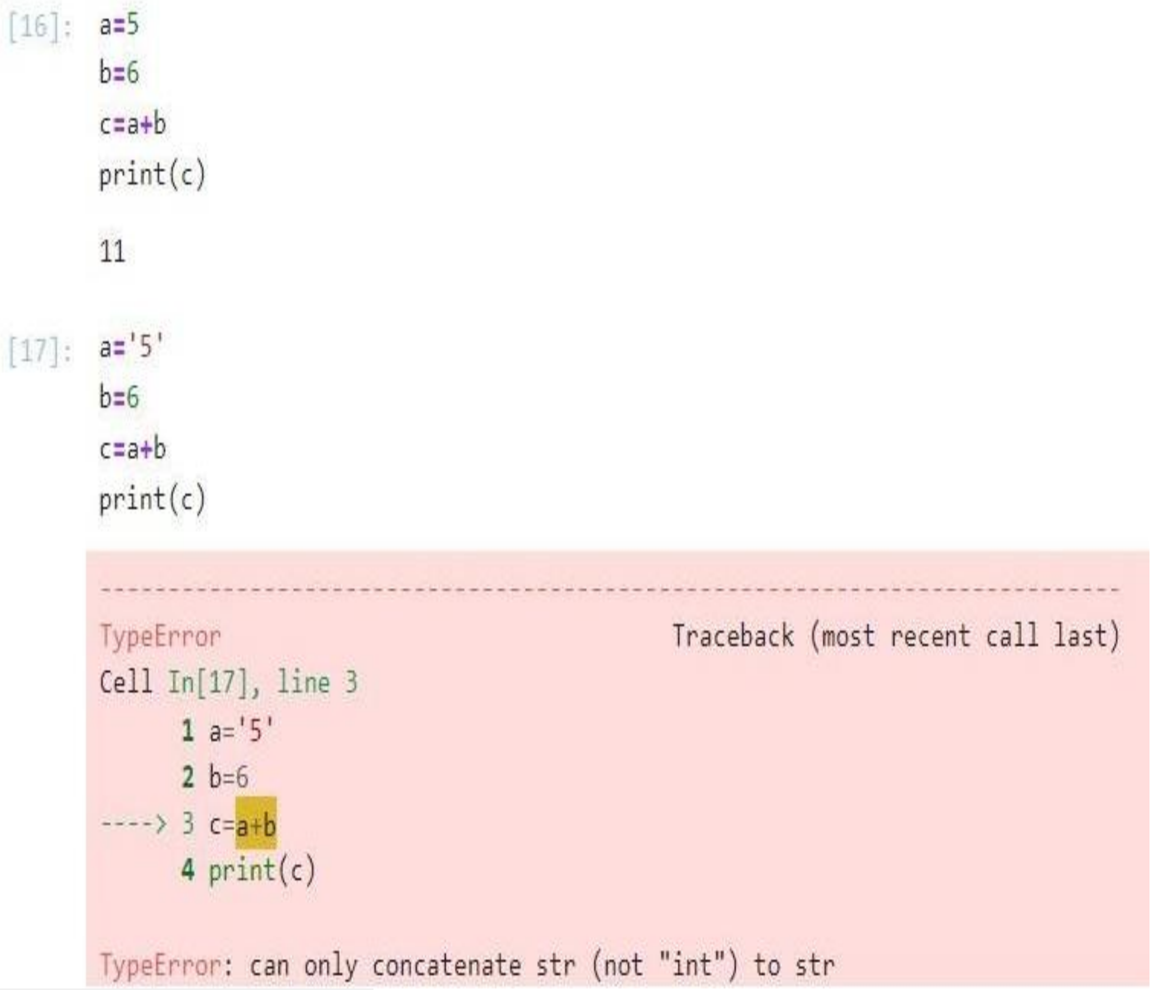
```
[6]:  c=4.5
      c

[6]:  4.5

[7]:  type(c)

[7]:  float

[8]:  print(a)

      5

[9]:  print(b)

      amity

[10]: print(c)

      4.5

[11]: c=4.5
      c
      type(c)

[11]: float
```



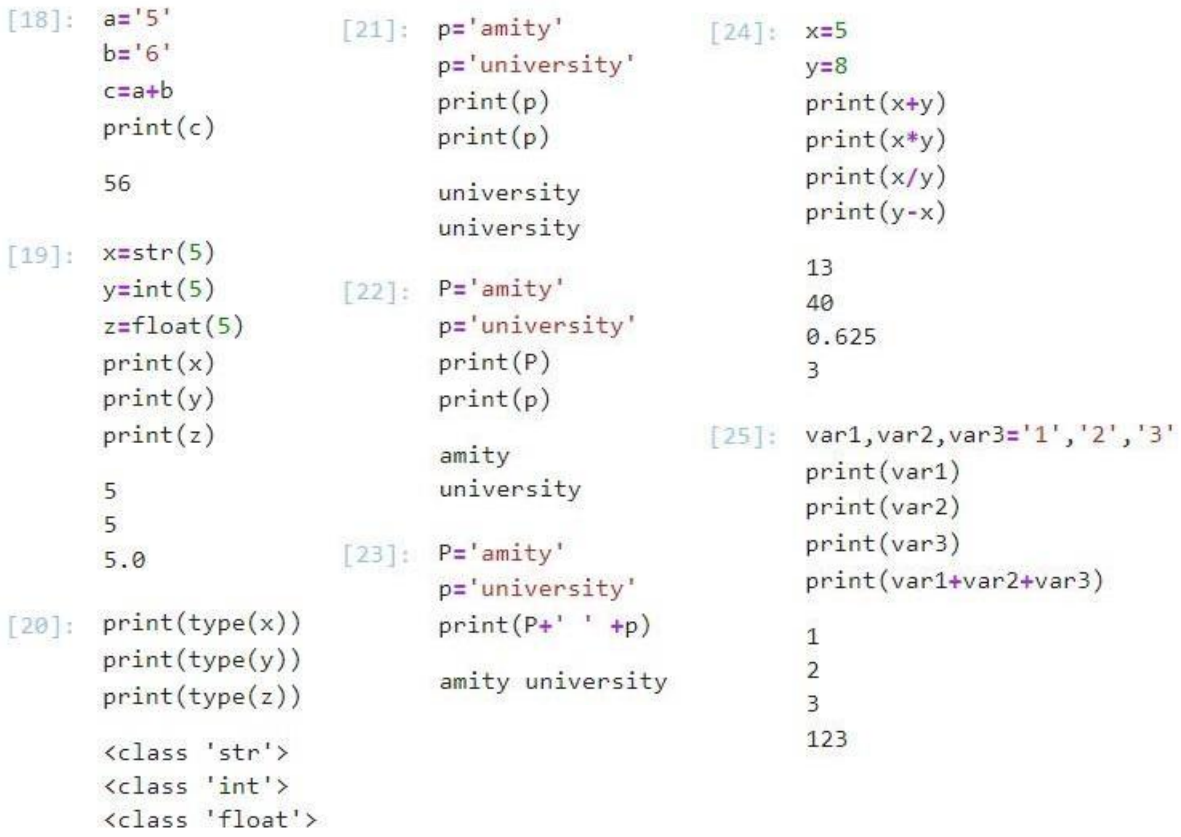
```
[12]: c=4.5
      type(c)
      c

[12]: 4.5

[13]: c=4.5
      print(type(c))
      print(c)

      <class 'float'>
      4.5
```



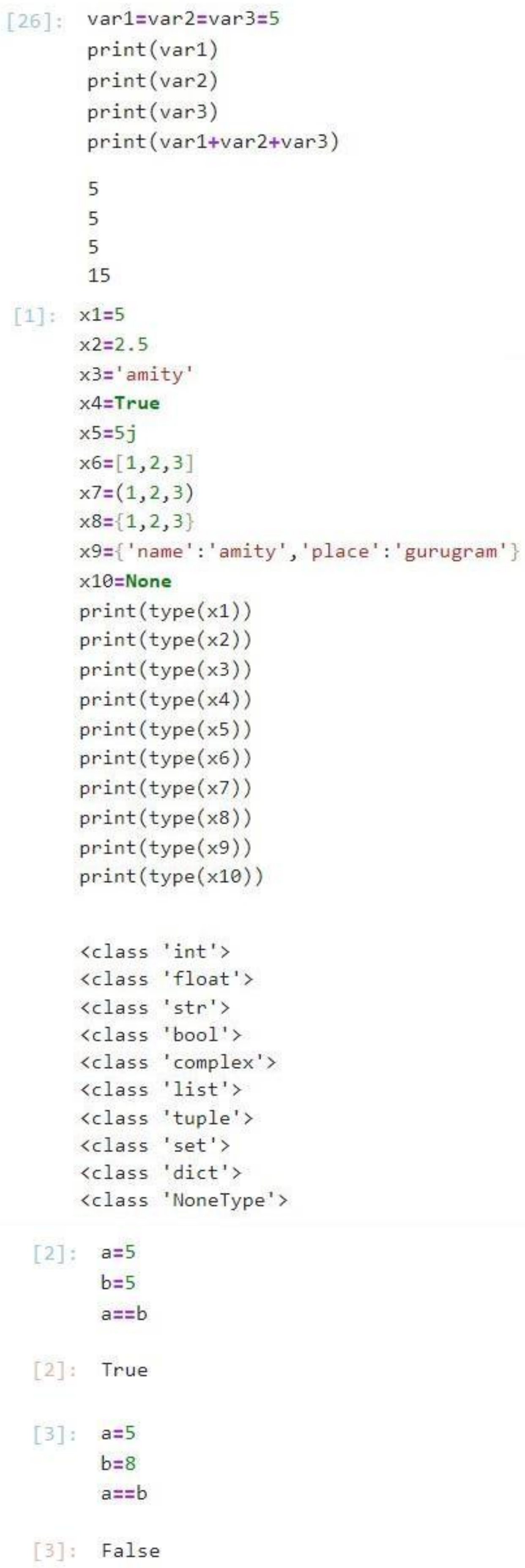
```
[14]: a=5
      b=6
      c=a+b
      c

[14]: 11

[15]: print(c)

      11
```



```
[16]: a=5
      b=6
      c=a+b
      print(c)

      11

[17]: a='5'
      b=6
      c=a+b
      print(c)

      ---------------------------------------------------------------------------
      TypeError                                 Traceback (most recent call last)
      Cell In[17], line 3
            1 a='5'
            2 b=6
      ----> 3 c=a+b
            4 print(c)

      TypeError: can only concatenate str (not "int") to str
```



```
[18]: a='5'
      b='6'
      c=a+b
      print(c)

      56

[19]: x=str(5)
      y=int(5)
      z=float(5)
      print(x)
      print(y)
      print(z)

      5
      5
      5.0

[20]: print(type(x))
      print(type(y))
      print(type(z))

      <class 'str'>
      <class 'int'>
      <class 'float'>

[21]: p='amity'
      p='university'
      print(p)
      print(p)

      university
      university

[22]: P='amity'
      p='university'
      print(P)
      print(p)

      amity
      university

[23]: P='amity'
      p='university'
      print(P+' ' +p)

      amity university

[24]: x=5
      y=8
      print(x+y)
      print(x*y)
      print(x/y)
      print(y-x)

      13
      40
      0.625
      3

[25]: var1,var2,var3='1','2','3'
      print(var1)
      print(var2)
      print(var3)
      print(var1+var2+var3)

      1
      2
      3
      123
```

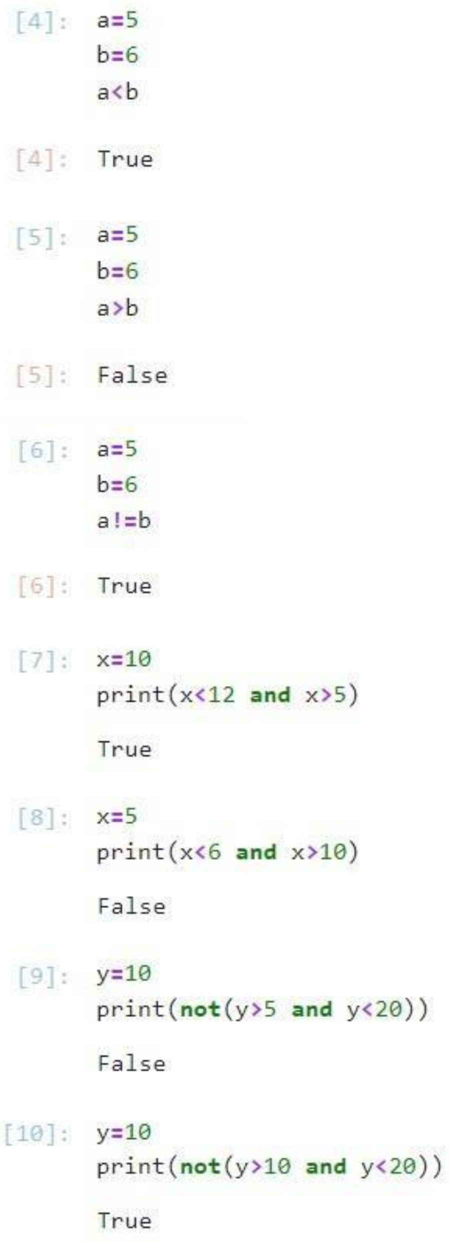


```
[26]: var1=var2=var3=5
      print(var1)
      print(var2)
      print(var3)
      print(var1+var2+var3)

      5
      5
      5
      15
```

```
[1]: x1=5
     x2=2.5
     x3='amity'
     x4=True
     x5=5j
     x6=[1,2,3]
     x7=(1,2,3)
     x8={1,2,3}
     x9={'name':'amity','place':'gurugram'}
     x10=None
     print(type(x1))
     print(type(x2))
     print(type(x3))
     print(type(x4))
     print(type(x5))
     print(type(x6))
     print(type(x7))
     print(type(x8))
     print(type(x9))
     print(type(x10))

     <class 'int'>
     <class 'float'>
     <class 'str'>
     <class 'bool'>
     <class 'complex'>
     <class 'list'>
     <class 'tuple'>
     <class 'set'>
     <class 'dict'>
     <class 'NoneType'>
```

```
[2]: a=5
     b=5
     a==b

[2]: True
```

```
[3]: a=5
     b=8
     a==b

[3]: False
```



```
[4]: a=5
     b=6
     a<b

[4]: True
```

```
[5]: a=5
     b=6
     a>b

[5]: False
```

```
[6]: a=5
     b=6
     a!=b

[6]: True
```

```
[7]: x=10
     print(x<12 and x>5)

     True
```

```
[8]: x=5
     print(x<6 and x>10)

     False
```

```
[9]: y=10
     print(not(y>5 and y<20))

     False
```

```
[10]: y=10
      print(not(y>10 and y<20))

      True
```

Experiment 2

Aim: Usage of List data type

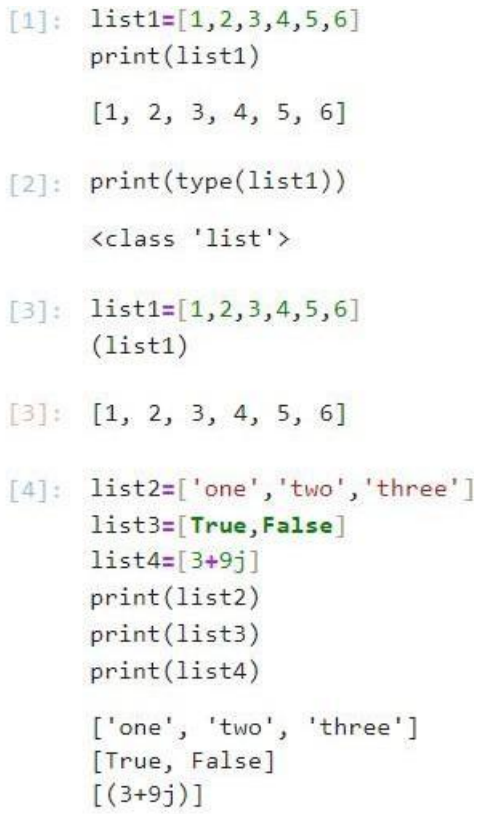
Theory:

Lists are used to store multiple items in a single variable.

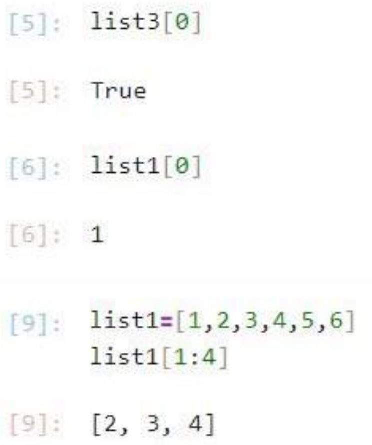
Lists are one of 4 built-in data types in Python used to store collections of data. List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1], etc.

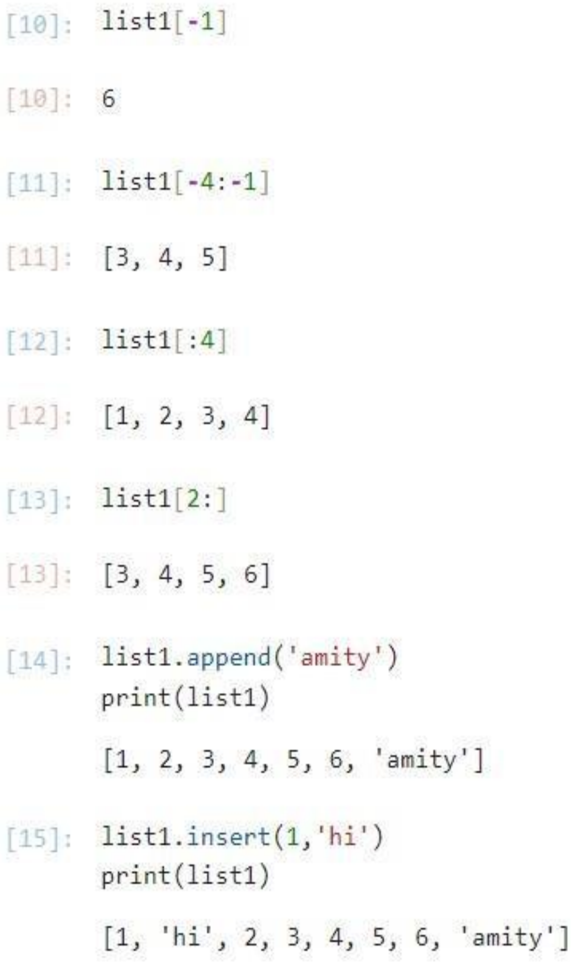
Source code/Output:



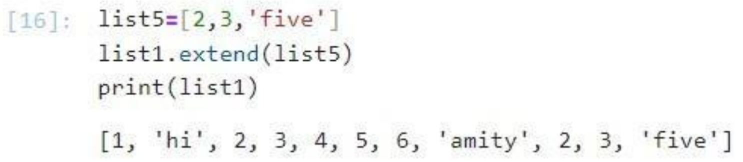
```
[1]:  list1=[1,2,3,4,5,6]
      print(list1)

      [1, 2, 3, 4, 5, 6]

[2]:  print(type(list1))

      <class 'list'>

[3]:  list1=[1,2,3,4,5,6]
      (list1)

[3]:  [1, 2, 3, 4, 5, 6]

[4]:  list2=['one','two','three']
      list3=[True,False]
      list4=[3+9j]
      print(list2)
      print(list3)
      print(list4)

      ['one', 'two', 'three']
      [True, False]
      [(3+9j)]
```



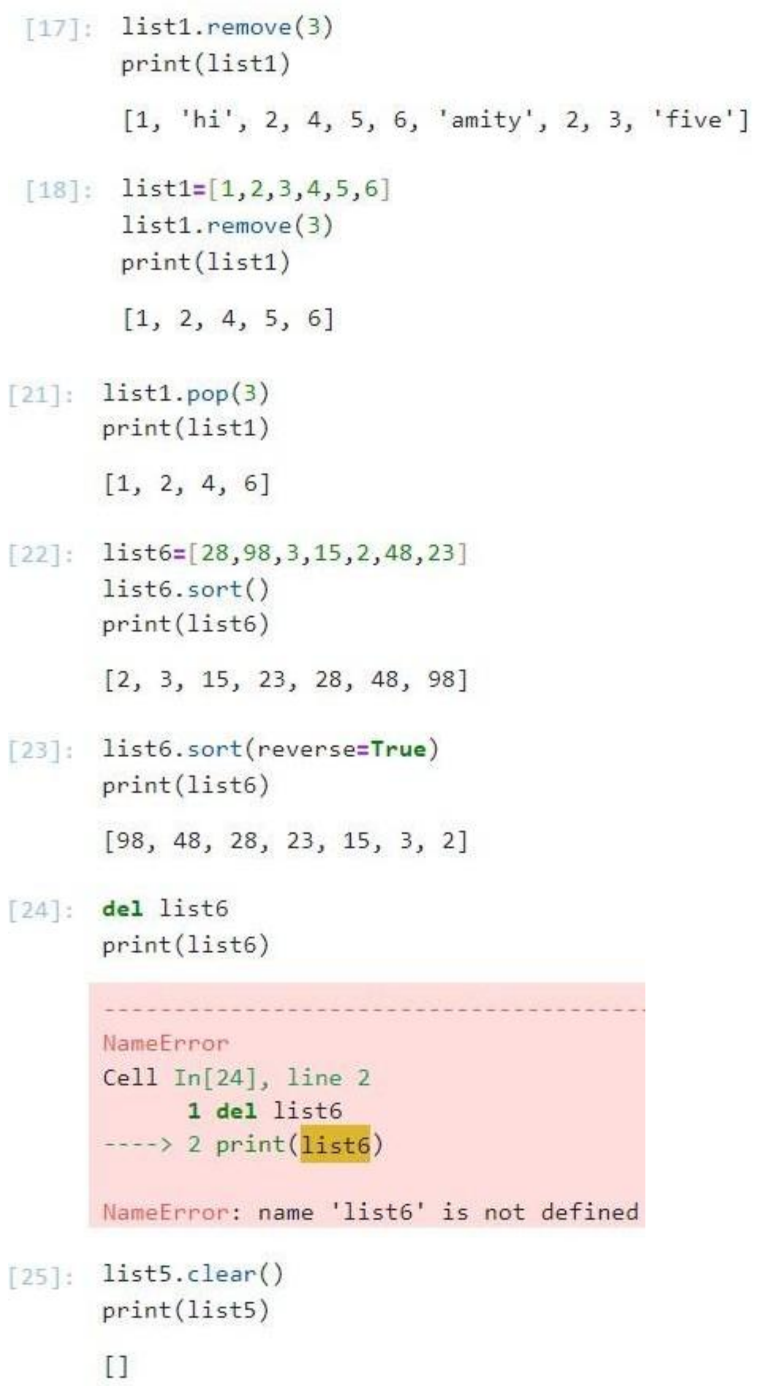
```
[5]:  list3[0]

[5]:  True

[6]:  list1[0]

[6]:  1

[9]:  list1=[1,2,3,4,5,6]
      list1[1:4]

[9]:  [2, 3, 4]
```



```
[10]:  list1[-1]

[10]:  6

[11]:  list1[-4:-1]

[11]:  [3, 4, 5]

[12]:  list1[:4]

[12]:  [1, 2, 3, 4]

[13]:  list1[2:]

[13]:  [3, 4, 5, 6]

[14]:  list1.append('amity')
       print(list1)

       [1, 2, 3, 4, 5, 6, 'amity']

[15]:  list1.insert(1,'hi')
       print(list1)

       [1, 'hi', 2, 3, 4, 5, 6, 'amity']
```



```
[16]:  list5=[2,3,'five']
       list1.extend(list5)
       print(list1)

       [1, 'hi', 2, 3, 4, 5, 6, 'amity', 2, 3, 'five']
```



```
[17]:  list1.remove(3)
       print(list1)

       [1, 'hi', 2, 4, 5, 6, 'amity', 2, 3, 'five']

[18]:  list1=[1,2,3,4,5,6]
       list1.remove(3)
       print(list1)

       [1, 2, 4, 5, 6]

[21]:  list1.pop(3)
       print(list1)

       [1, 2, 4, 6]

[22]:  list6=[28,98,3,15,2,48,23]
       list6.sort()
       print(list6)

       [2, 3, 15, 23, 28, 48, 98]

[23]:  list6.sort(reverse=True)
       print(list6)

       [98, 48, 28, 23, 15, 3, 2]

[24]:  del list6
       print(list6)

       ---------------------------------------
       NameError
       Cell In[24], line 2
             1 del list6
       ----> 2 print(list6)

       NameError: name 'list6' is not defined

[25]:  list5.clear()
       print(list5)

       []
```

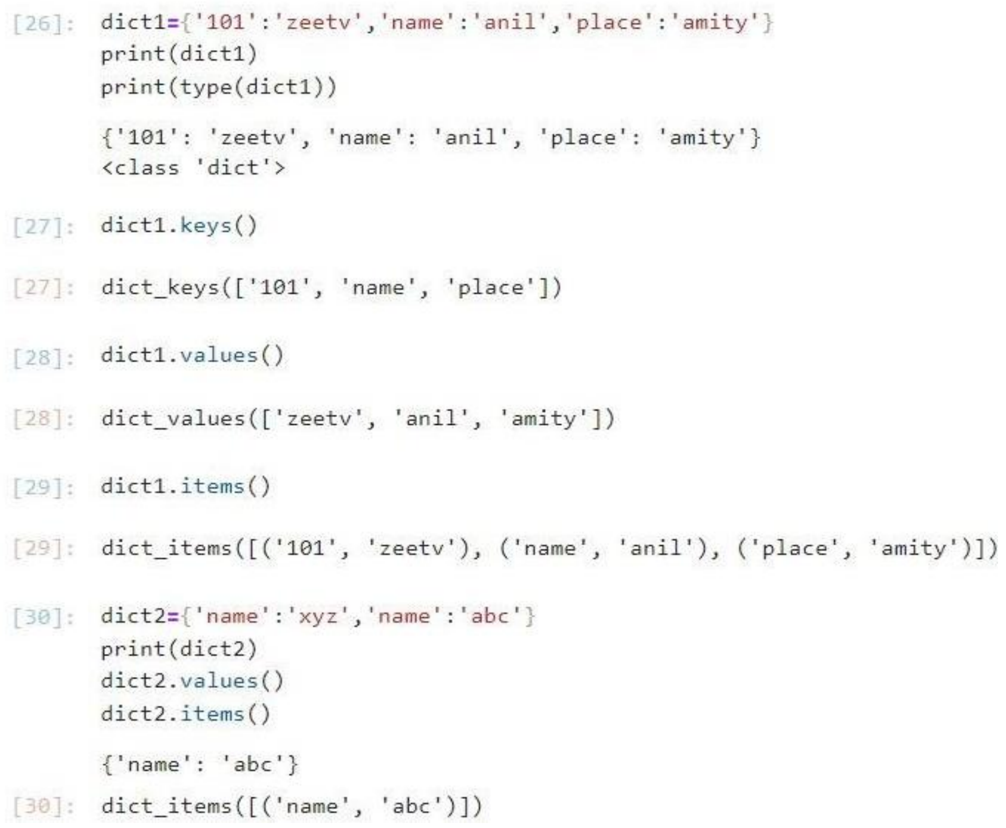
Experiment 3

Aim: Usage of Dictionary data type

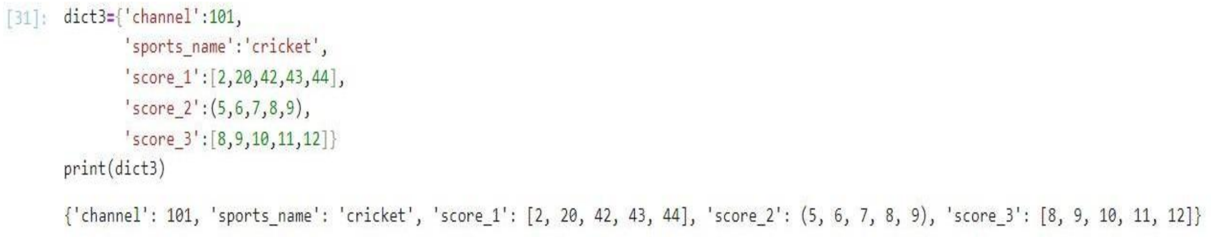
Theory:

A dictionary in Python is a data structure that stores the value in key: value pairs.

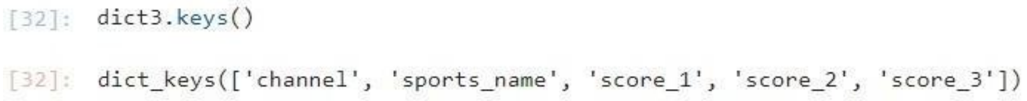
Source code/Output:



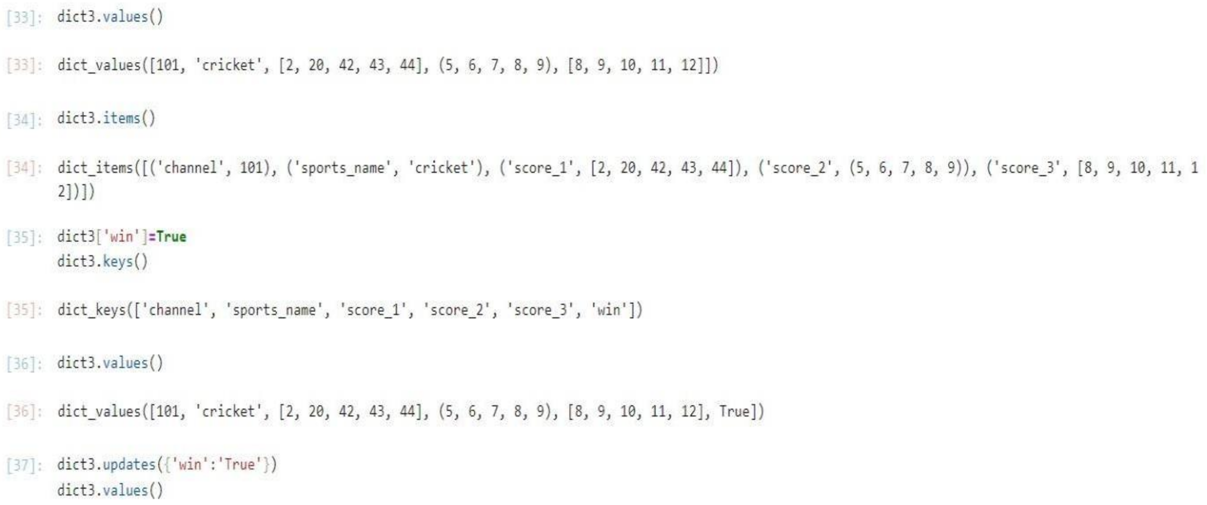
```
[26]:  dict1={'101':'zeetv','name':'anil','place':'amity'}
       print(dict1)
       print(type(dict1))

       {'101': 'zeetv', 'name': 'anil', 'place': 'amity'}
       <class 'dict'>

[27]:  dict1.keys()

[27]:  dict_keys(['101', 'name', 'place'])

[28]:  dict1.values()

[28]:  dict_values(['zeetv', 'anil', 'amity'])

[29]:  dict1.items()

[29]:  dict_items([('101', 'zeetv'), ('name', 'anil'), ('place', 'amity')])

[30]:  dict2={'name':'xyz','name':'abc'}
       print(dict2)
       dict2.values()
       dict2.items()

       {'name': 'abc'}
[30]:  dict_items([('name', 'abc')])
```



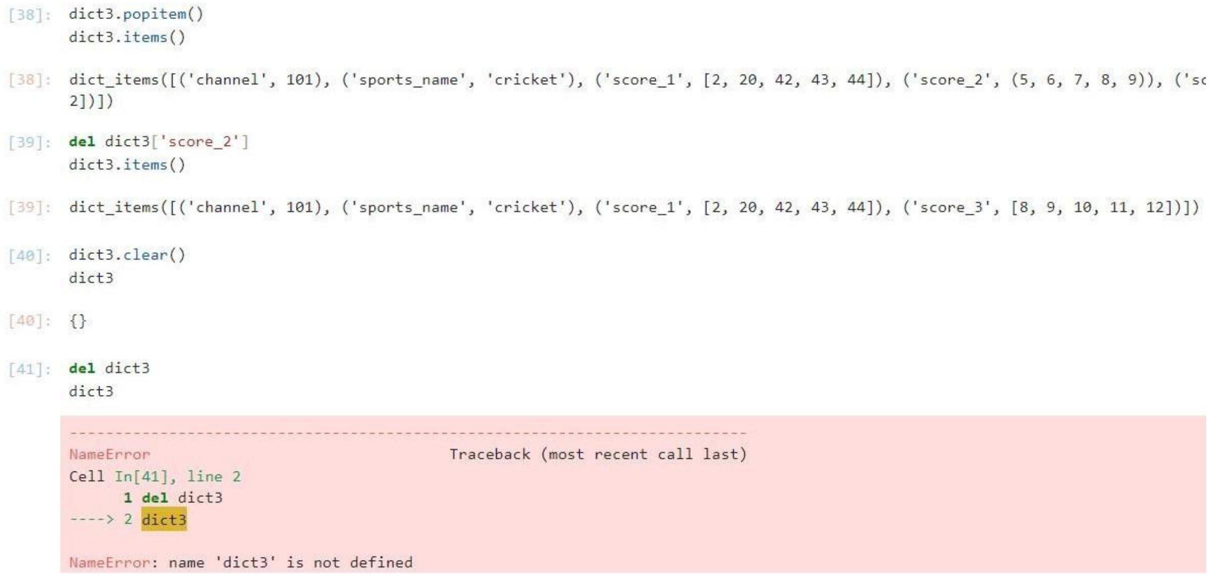
```
[31]:  dict3={'channel':101,
              'sports_name':'cricket',
              'score_1':[2,20,42,43,44],
              'score_2':(5,6,7,8,9),
              'score_3':[8,9,10,11,12]}
       print(dict3)

       {'channel': 101, 'sports_name': 'cricket', 'score_1': [2, 20, 42, 43, 44], 'score_2': (5, 6, 7, 8, 9), 'score_3': [8, 9, 10, 11, 12]}
```



```
[32]:  dict3.keys()

[32]:  dict_keys(['channel', 'sports_name', 'score_1', 'score_2', 'score_3'])
```



```
[33]: dict3.values()

[33]: dict_values([101, 'cricket', [2, 20, 42, 43, 44], (5, 6, 7, 8, 9), [8, 9, 10, 11, 12]])

[34]: dict3.items()

[34]: dict_items([('channel', 101), ('sports_name', 'cricket'), ('score_1', [2, 20, 42, 43, 44]), ('score_2', (5, 6, 7, 8, 9)), ('score_3', [8, 9, 10, 11, 1
      2])])

[35]: dict3['win']=True
      dict3.keys()

[35]: dict_keys(['channel', 'sports_name', 'score_1', 'score_2', 'score_3', 'win'])

[36]: dict3.values()

[36]: dict_values([101, 'cricket', [2, 20, 42, 43, 44], (5, 6, 7, 8, 9), [8, 9, 10, 11, 12], True])

[37]: dict3.updates({'win':'True'})
      dict3.values()
```



```
[38]: dict3.popitem()
      dict3.items()

[38]: dict_items([('channel', 101), ('sports_name', 'cricket'), ('score_1', [2, 20, 42, 43, 44]), ('score_2', (5, 6, 7, 8, 9)), ('sc
      2])])

[39]: del dict3['score_2']
      dict3.items()

[39]: dict_items([('channel', 101), ('sports_name', 'cricket'), ('score_1', [2, 20, 42, 43, 44]), ('score_3', [8, 9, 10, 11, 12])])

[40]: dict3.clear()
      dict3

[40]: {}

[41]: del dict3
      dict3

      ---------------------------------------------------------------------------
      NameError                                 Traceback (most recent call last)
      Cell In[41], line 2
            1 del dict3
      ----> 2 dict3

      NameError: name 'dict3' is not defined
```

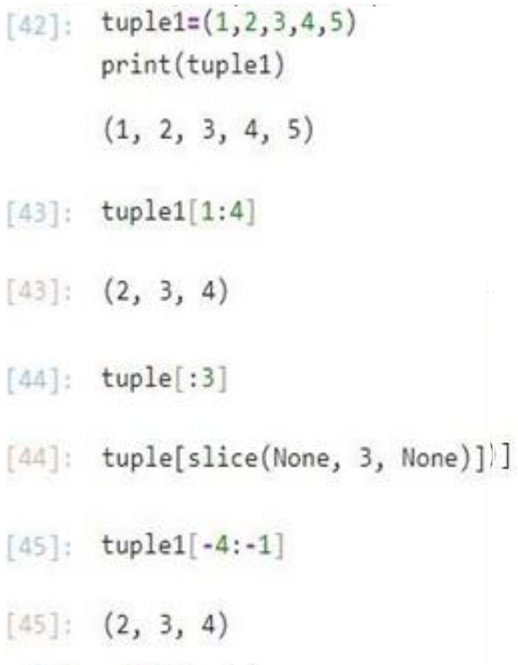
Experiment 4

Aim: Usage of Tuple data type

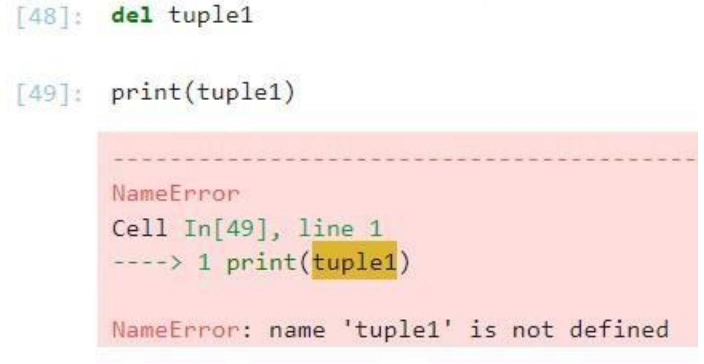
Theory:

Tuple is a collection of objects separated by commas. In some ways, a tuple is similar to a Python list in terms of indexing, nested objects, and repetition but the main difference between both is Python tuple is immutable, unlike the Python list which is mutable.

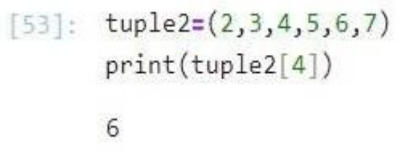
Source code/Output:



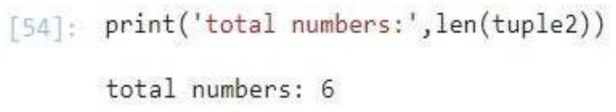
```
[42]:  tuple1=(1,2,3,4,5)
       print(tuple1)

       (1, 2, 3, 4, 5)

[43]:  tuple1[1:4]

[43]:  (2, 3, 4)

[44]:  tuple[:3]

[44]:  tuple[slice(None, 3, None)]]

[45]:  tuple1[-4:-1]

[45]:  (2, 3, 4)
```



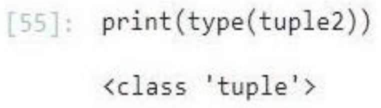
```
[48]:  del tuple1

[49]:  print(tuple1)

       ----------------------------------------
       NameError
       Cell In[49], line 1
       ----> 1 print(tuple1)

       NameError: name 'tuple1' is not defined
```



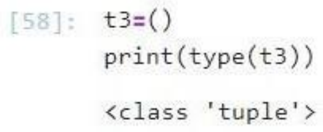
```
[53]:  tuple2=(2,3,4,5,6,7)
       print(tuple2[4])

       6
```



```
[54]:  print('total numbers:',len(tuple2))

       total numbers: 6
```



```
[55]:  print(type(tuple2))

       <class 'tuple'>
```



```
[58]:  t3=()
       print(type(t3))

       <class 'tuple'>
```

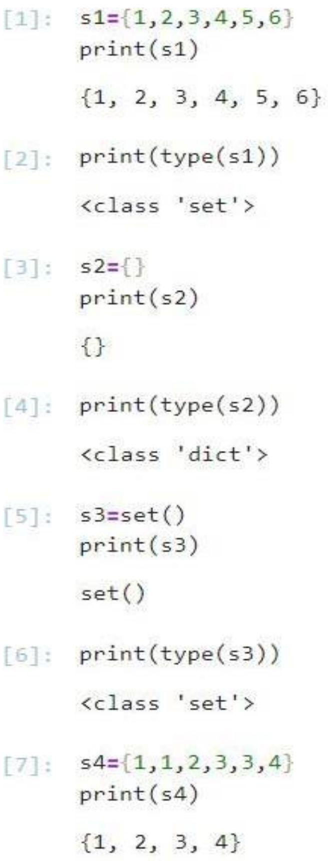
Experiment 5

Aim: Usage of Set data type

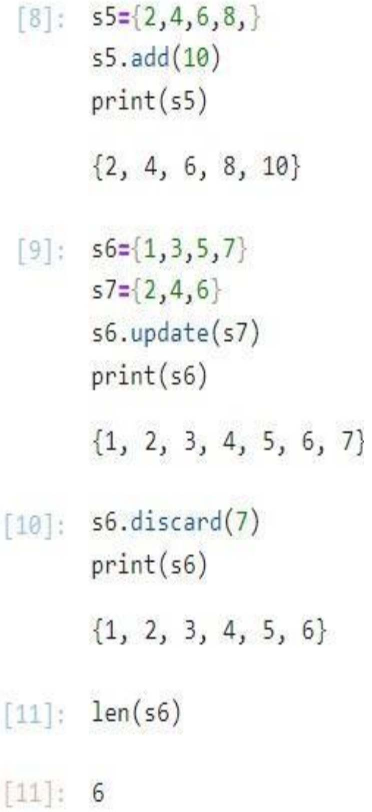
Theory:

A set is a collection which is unordered, unchangeable, and unindexed. Sets are written withcurly brackets.

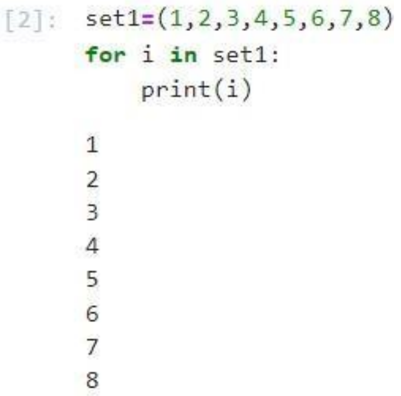
Source code/Output:



```
[1]:  s1={1,2,3,4,5,6}
      print(s1)

      {1, 2, 3, 4, 5, 6}

[2]:  print(type(s1))

      <class 'set'>

[3]:  s2={}
      print(s2)

      {}

[4]:  print(type(s2))

      <class 'dict'>

[5]:  s3=set()
      print(s3)

      set()

[6]:  print(type(s3))

      <class 'set'>

[7]:  s4={1,1,2,3,3,4}
      print(s4)

      {1, 2, 3, 4}
```



```
[8]:  s5={2,4,6,8,}
      s5.add(10)
      print(s5)

      {2, 4, 6, 8, 10}

[9]:  s6={1,3,5,7}
      s7={2,4,6}
      s6.update(s7)
      print(s6)

      {1, 2, 3, 4, 5, 6, 7}

[10]: s6.discard(7)
      print(s6)

      {1, 2, 3, 4, 5, 6}

[11]: len(s6)

[11]: 6
```

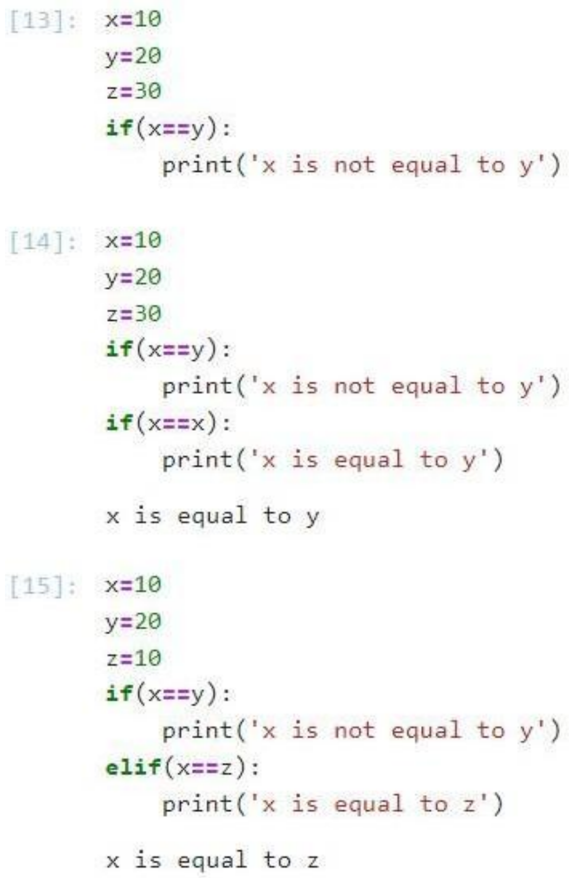


```
[2]:  set1=(1,2,3,4,5,6,7,8)
      for i in set1:
          print(i)

      1
      2
      3
      4
      5
      6
      7
      8
```

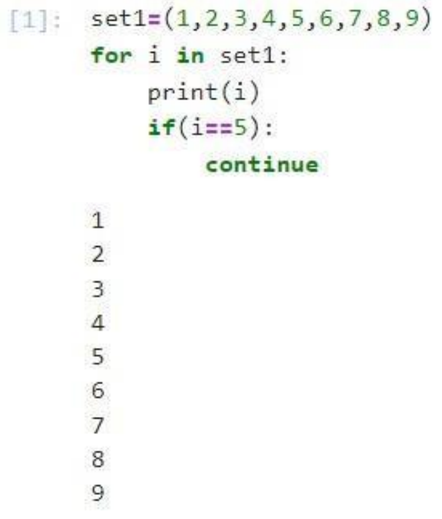
Experiment 6

Aim: Usage of if-else statement and continue and break statements

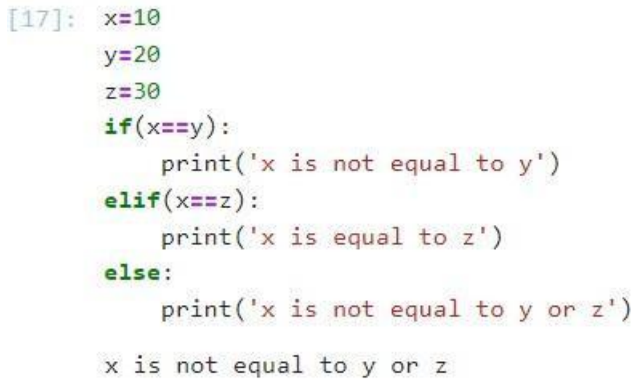
Source code/Output:



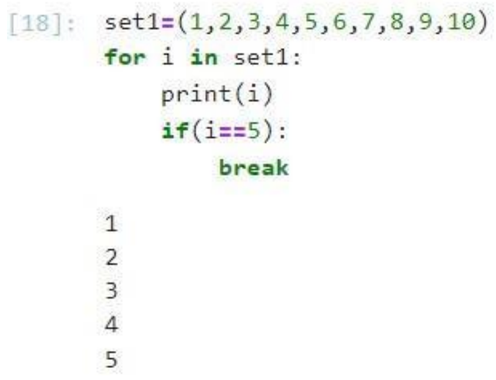
```
[13]: x=10
      y=20
      z=30
      if(x==y):
          print('x is not equal to y')

[14]: x=10
      y=20
      z=30
      if(x==y):
          print('x is not equal to y')
      if(x==x):
          print('x is equal to y')

      x is equal to y

[15]: x=10
      y=20
      z=10
      if(x==y):
          print('x is not equal to y')
      elif(x==z):
          print('x is equal to z')

      x is equal to z
```



```
[1]: set1=(1,2,3,4,5,6,7,8,9)
     for i in set1:
         print(i)
         if(i==5):
             continue

     1
     2
     3
     4
     5
     6
     7
     8
     9
```



```
[17]: x=10
      y=20
      z=30
      if(x==y):
          print('x is not equal to y')
      elif(x==z):
          print('x is equal to z')
      else:
          print('x is not equal to y or z')

      x is not equal to y or z
```



```
[18]: set1=(1,2,3,4,5,6,7,8,9,10)
      for i in set1:
          print(i)
          if(i==5):
              break

      1
      2
      3
      4
      5
```

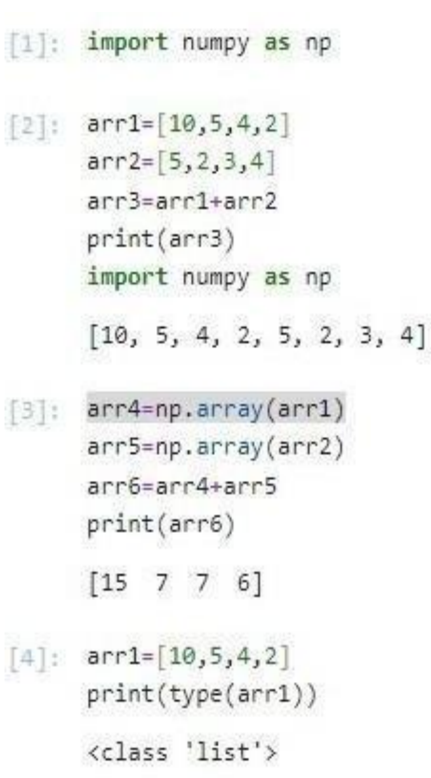
Experiment 7

Aim: Usage of numpy library

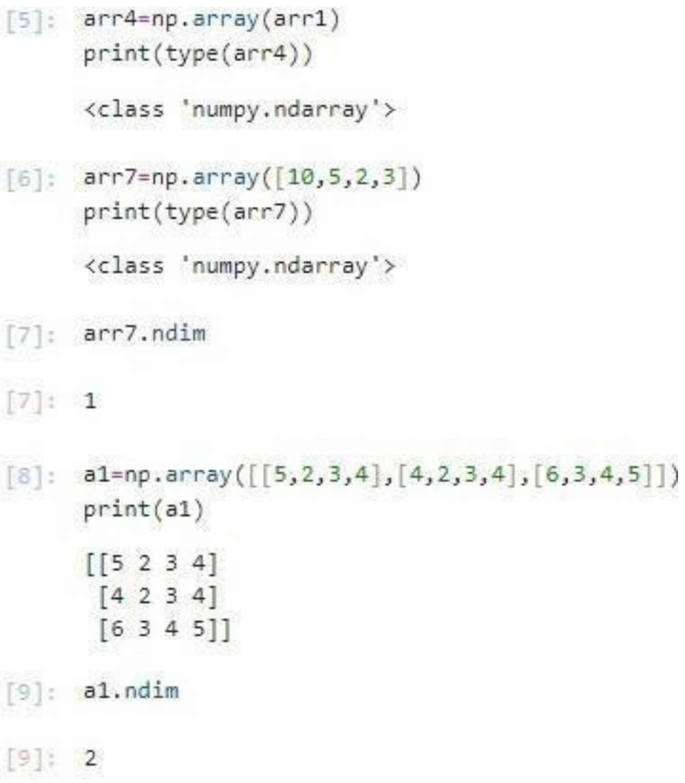
Theory:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.

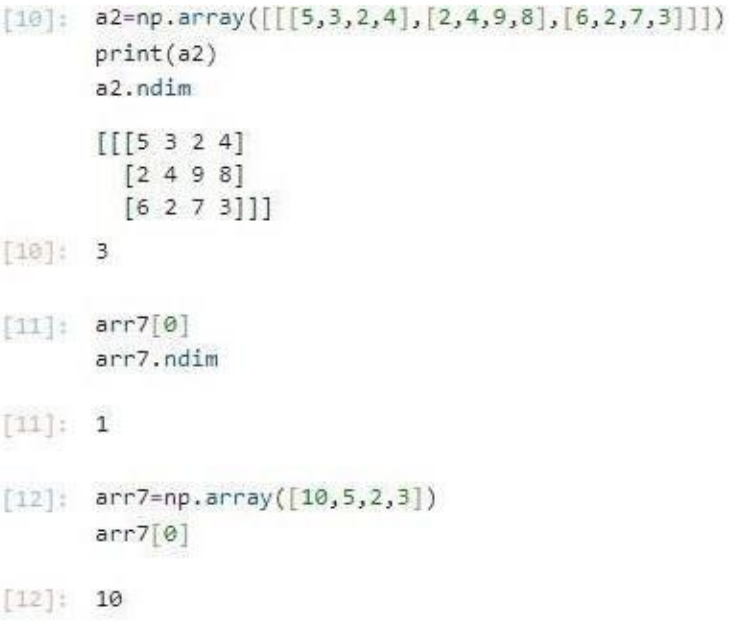
Source code/Output:



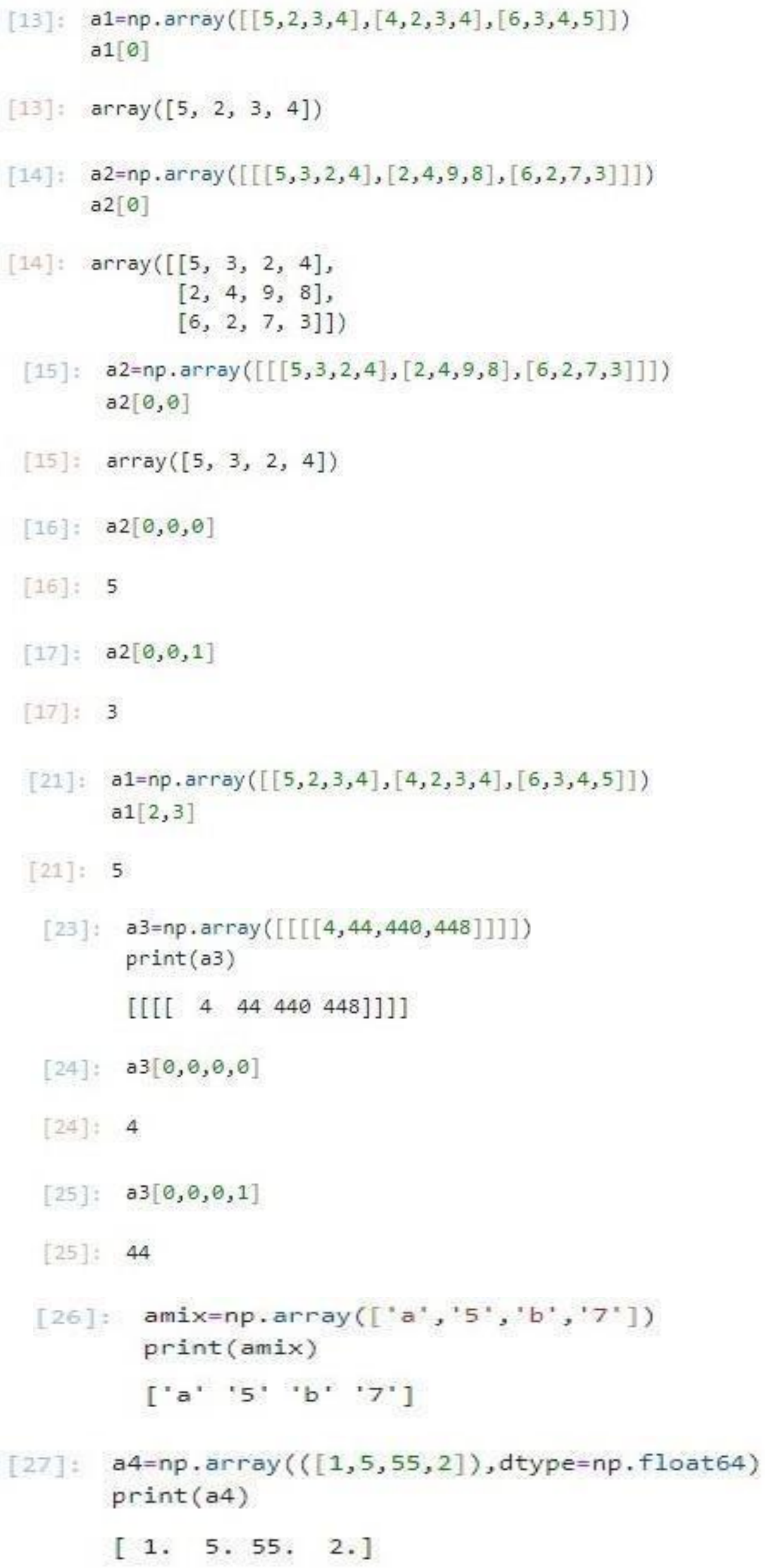
```
[1]:  import numpy as np

[2]:  arr1=[10,5,4,2]
      arr2=[5,2,3,4]
      arr3=arr1+arr2
      print(arr3)
      import numpy as np

      [10, 5, 4, 2, 5, 2, 3, 4]

[3]:  arr4=np.array(arr1)
      arr5=np.array(arr2)
      arr6=arr4+arr5
      print(arr6)

      [15  7  7  6]

[4]:  arr1=[10,5,4,2]
      print(type(arr1))

      <class 'list'>
```



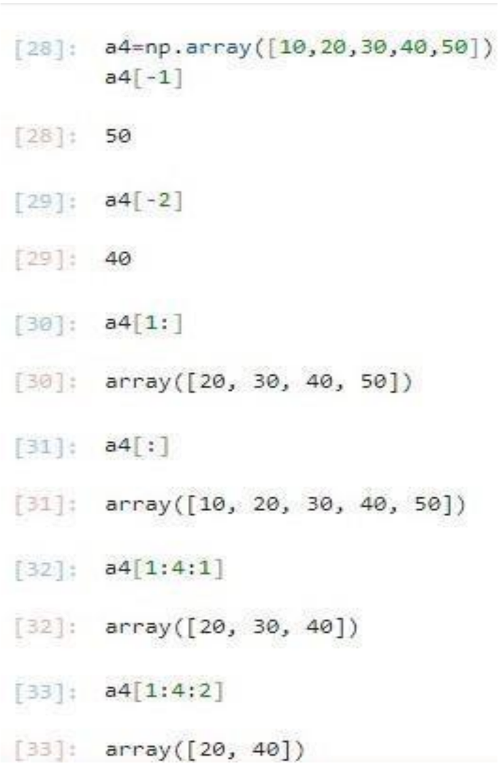
```
[5]:  arr4=np.array(arr1)
      print(type(arr4))

      <class 'numpy.ndarray'>

[6]:  arr7=np.array([10,5,2,3])
      print(type(arr7))

      <class 'numpy.ndarray'>

[7]:  arr7.ndim

[7]:  1

[8]:  a1=np.array([[5,2,3,4],[4,2,3,4],[6,3,4,5]])
      print(a1)

      [[5 2 3 4]
       [4 2 3 4]
       [6 3 4 5]]

[9]:  a1.ndim

[9]:  2
```



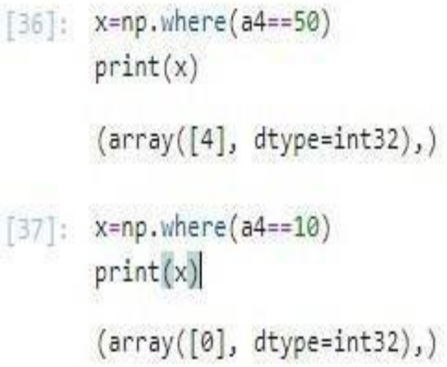
```
[10]:  a2=np.array([[[5,3,2,4],[2,4,9,8],[6,2,7,3]]])
       print(a2)
       a2.ndim

       [[[5 3 2 4]
         [2 4 9 8]
         [6 2 7 3]]]

[10]:  3

[11]:  arr7[0]
       arr7.ndim

[11]:  1

[12]:  arr7=np.array([10,5,2,3])
       arr7[0]

[12]:  10
```



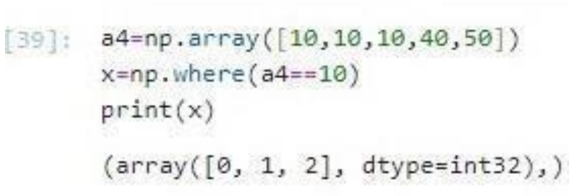
```
[13]: a1=np.array([[5,2,3,4],[4,2,3,4],[6,3,4,5]])
      a1[0]

[13]: array([5, 2, 3, 4])

[14]: a2=np.array([[[5,3,2,4],[2,4,9,8],[6,2,7,3]]])
      a2[0]

[14]: array([[5, 3, 2, 4],
             [2, 4, 9, 8],
             [6, 2, 7, 3]])

[15]: a2=np.array([[[5,3,2,4],[2,4,9,8],[6,2,7,3]]])
      a2[0,0]

[15]: array([5, 3, 2, 4])

[16]: a2[0,0,0]

[16]: 5

[17]: a2[0,0,1]

[17]: 3

[21]: a1=np.array([[5,2,3,4],[4,2,3,4],[6,3,4,5]])
      a1[2,3]

[21]: 5

[23]: a3=np.array([[[[4,44,440,448]]]])
      print(a3)

      [[[[  4  44 440 448]]]]

[24]: a3[0,0,0,0]

[24]: 4

[25]: a3[0,0,0,1]

[25]: 44

[26]: amix=np.array(['a','5','b','7'])
      print(amix)

      ['a' '5' 'b' '7']

[27]: a4=np.array(([1,5,55,2]),dtype=np.float64)
      print(a4)

      [ 1.  5. 55.  2.]
```



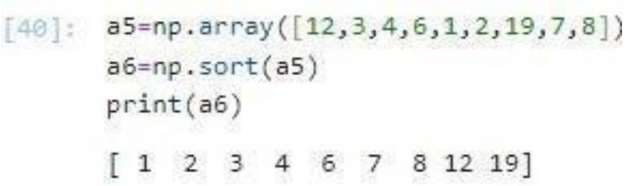
```
[28]:  a4=np.array([10,20,30,40,50])
       a4[-1]

[28]:  50

[29]:  a4[-2]

[29]:  40

[30]:  a4[1:]

[30]:  array([20, 30, 40, 50])

[31]:  a4[:]

[31]:  array([10, 20, 30, 40, 50])

[32]:  a4[1:4:1]

[32]:  array([20, 30, 40])

[33]:  a4[1:4:2]

[33]:  array([20, 40])
```



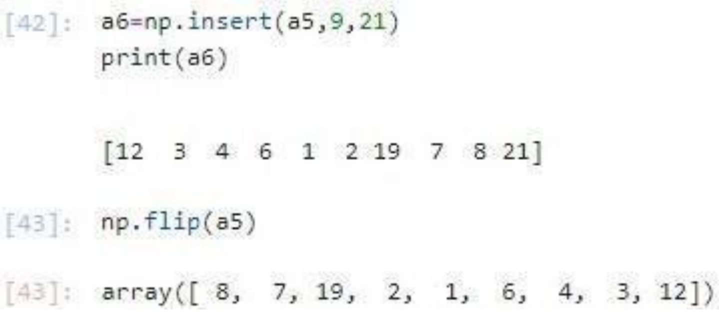
```
[36]:  x=np.where(a4==50)
       print(x)

       (array([4], dtype=int32),)

[37]:  x=np.where(a4==10)
       print(x)

       (array([0], dtype=int32),)
```



```
[39]:  a4=np.array([10,10,10,40,50])
       x=np.where(a4==10)
       print(x)

       (array([0, 1, 2], dtype=int32),)
```



```
[40]:  a5=np.array([12,3,4,6,1,2,19,7,8])
       a6=np.sort(a5)
       print(a6)

       [ 1  2  3  4  6  7  8 12 19]
```

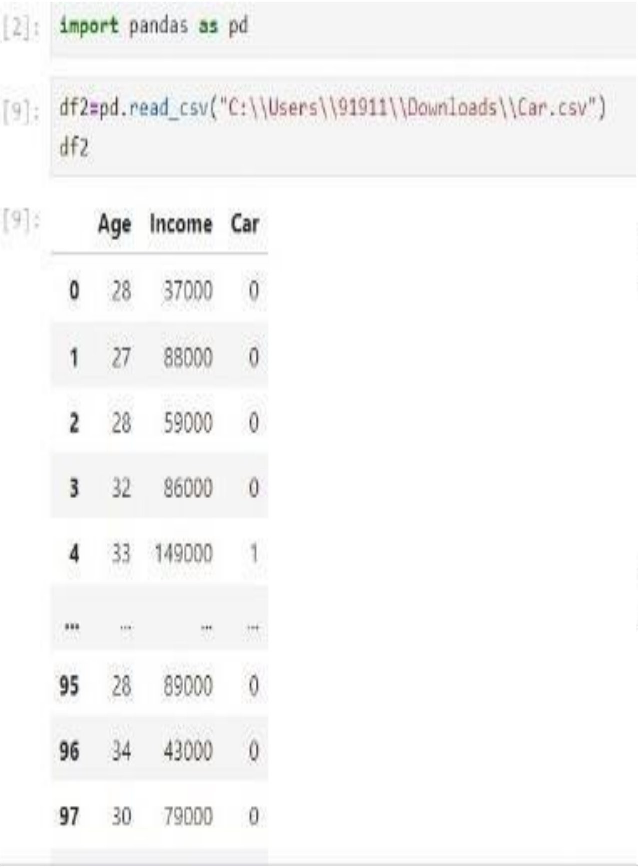


```
[42]:  a6=np.insert(a5,9,21)
       print(a6)


       [12  3  4  6  1  2 19  7  8 21]

[43]:  np.flip(a5)

[43]:  array([ 8,  7, 19,  2,  1,  6,  4,  3, 12])
```

Experiment 8

Aim: Usage of pandas library

Source code/Output:

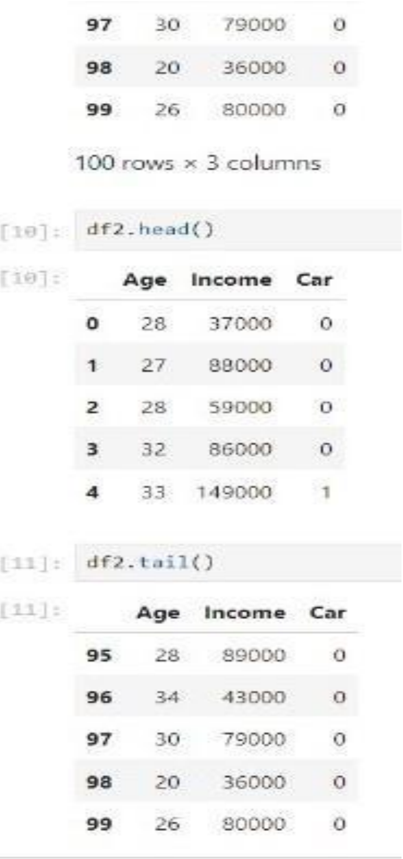


```
[2]: import pandas as pd
```

```
[9]: df2=pd.read_csv("C:\\Users\\91911\\Downloads\\Car.csv")
     df2
```

[9]:

	Age	Income	Car
0	28	37000	0
1	27	88000	0
2	28	59000	0
3	32	86000	0
4	33	149000	1
...	...	...	...
95	28	89000	0
96	34	43000	0
97	30	79000	0
98	20	36000	0
99	26	80000	0



100 rows × 3 columns

```
[10]: df2.head()
```

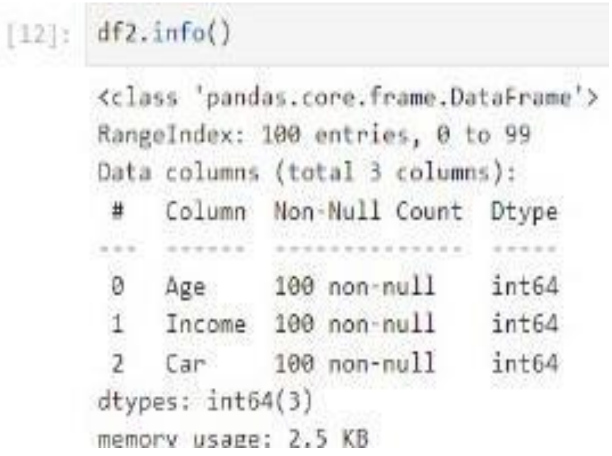
[10]:

	Age	Income	Car
0	28	37000	0
1	27	88000	0
2	28	59000	0
3	32	86000	0
4	33	149000	1

```
[11]: df2.tail()
```

[11]:

	Age	Income	Car
95	28	89000	0
96	34	43000	0
97	30	79000	0
98	20	36000	0
99	26	80000	0



```
[12]: df2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Age     100 non-null    int64
 1   Income  100 non-null    int64
 2   Car     100 non-null    int64
dtypes: int64(3)
memory usage: 2.5 KB
```