Name : Meith Navlakha
SAP : 60004190068
Python Lab

EXPERIMENT 1

AIM : Exploring basics of python like data types (strings, list, array, set, tuple) and control statements

THEORY :

1) Python Numbers:

Numeric data type represent the data which has numeric value. Numeric value can be integer, floating number or even complex numbers. These values are defined as int, float and complex class in Python.

- **Integers** – This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.
- **Float** – This value is represented by float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
- **Complex Numbers** – Complex number is represented by complex class. It is specified as *(real part) + (imaginary part)j*. For example – 2+3j

type() : function to know which class a variable or a value belongs to.

isinstance() function is used to check if an object belongs to a particular class.

2) Python List

- An ordered sequence of items is called List.
- It is a very flexible data type in Python. It supports heterogeneous values i.e. it need not have only same data types.
- Lists are ordered, it means that the items have a defined order.If a new item is added into the list then it would be appended at the end.
- Allows duplicates as all the values have a unique index.
- Lists are mutable i.e. it can be updated.

3) Python Tuple

- Tuples are ordered collection used to store multiple items in a single variable.
- Tuples are initialised within parentheses (round brackets) with items being separated by commas.

  Eg: Tuple1 = (‘This’, ‘is’ , 1 , 2+5j , ‘Tuple’)

- Creation of Python tuple without the use of parentheses is known as Tuple Packing.
- Tuples differ my list as tuples are immutable i.e. it cannot be modified.
- Tuples are used to write-protect data and are usually faster than lists as they cannot change dynamically.

4) Python String

- In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote.
  Eg. 'hello' is the same as "hello".
- In python there is no character data type, a character is a string of length one. It is represented by str class**.**
- Strings are immutable
- Multi-line strings can be denoted using triple quotes, ''' or """.
- Every individual characters of a String can be accessed by using the method of Indexing. Indexing even allows negative address references to access characters from the back of the String, e.g. -1 refers to the last character, -2 refers to the second last character and so on.

5) Python Set

- Set is an unordered collection of unique items. It arranges the
- Set is defined by values separated by comma inside braces { }. Items in a set are not ordered.
- Eg: set1 = set([1, 2, Hello, 4.0, 'For', 6, Hello])

  Output: {1, 2, Hello , 4.0 , For , 6)

- Sets have unique values. They eliminate duplicates.
- We can perform set operations like union, intersection on two sets.
- Set items cannot be accessed by referring to an index, since sets are unordered the items has no index. We loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

6) Python Dictionary

- Dictionary is an unordered collection of data values, used to store data values like a map.
- Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon : and each key is separated by a 'comma'.
- Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.
- A dictionary can be created by placing a sequence of elements within curly {} braces, separated by 'comma'. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated.
- Dictionary can also be created by the built-in function dict().

  Eg: Dict = {1: 'Meith' , 2: 30 , 3: 'Hello'}

  Dict = {'Name': 'Meith' , 'Age': 20 , 'Marks': [1, 2, 3, 4]}

- Dictionaries are mutable and data can be added ,updated or deleted.
- In order to access the items of a dictionary refer to its key name. Key can be used inside square brackets. There is also a method called get() that will also help in accessing the element from a dictionary.

7) Python Arrays

- An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together.
- Array can be handled in Python by a module named array. They can be useful when we have to manipulate only a specific data type values.

- A user can even treat lists as arrays. However, user cannot constraint the type of elements stored in a list. If you create arrays using the array module, all elements of the array must be of the same type.
- Array in Python can be created by importing array module. **array(data_type, value_list)** is used to create an array with data type and value list specified in its arguments.

8) Loops

Python programming language provides following types of loops to handle looping requirements. Python provides three ways for executing the loops. A). A loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string)While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

A) For
B) While
C) For else
D) Nested Loops

9) Control Statements

In general, statements are executed sequentially. There may be a situation when you need to execute a block of code several number of times. Control statements change execution from its normal sequence. Programming languages provide various control structures that allow for more complicated execution paths.

Python supports the following control statements:

a) Break : Terminates the loop statement and transfers execution to the statement immediately following the loop.
b) Continue : Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
c) Pass : The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

Name : Meith Navlakha
SAP : 60004190068
Python Lab

CODE :

1) Python Numbers

```
m = 10
print (m , 'type of m : ' , type(m))

n = 5.0
print (n , 'type of n : ' , type(n))

k = 1 + 10j
print (k , 'type of k : ' , type(k))
print (k , 'is k complex? ' , isinstance(k , complex))
```

OUTPUT:

```
10 type of m :  <class 'int'>
5.0 type of n :  <class 'float'>
(1+10j) type of k :  <class 'complex'>
(1+10j) is k complex?  True
```

2) Python List

```
# List
a= [10 , 2.4 , 'Hello' , 1+23j , 8]

# Operations on List
print('a[2] : ' , a[2]) # Fetching Single index
print('First 3 elements : ' , a[:3]) # slicing
print(a)

# List are Mutable
a[2] = 'Bye'
print('Updated Element : ' , a[2])
```

OUTPUT:

```
a[2] :  Hello
First 3 elements :  [10, 2.4, 'Hello']
[10, 2.4, 'Hello', (1+23j), 8]
Updated Element :  Bye
```

Name : Meith Navlakha
SAP : 60004190068
Python Lab

3) Python Tuple

```
# Initialise Tuple
a= (10 , 2.4 , 'Hello' , 1+23j , 8)

print(a)
print('a[2] : ' , a[2]) # Fetching by index
print('First 3 elements : ' , a[:3]) # Slicing tuple
```

OUTPUT:

```
(10, 2.4, 'Hello', (1+23j), 8)
a[2] :  Hello
First 3 elements : (10, 2.4, 'Hello')
```

4) Python String

```
s = "I am learning python"

print(s)
print("s[5] : " , s[5])
print("s[6:11] : " , s[6:11])

# Multi Line String
l='''A multiline
string'''
print('\n' , l)
print("s[6:11] : " , s[6:11])
```

OUTPUT:

```
I am learning python
s[5] :  l
s[6:11] :  earni

 A multiline
string
s[6:11] :  earni
```

5) Python Set

```
# Python Set
a={4, 6, 1 , 8 ,9, 0 , 6 ,6}
print("a =" , a)
print(type(a))
```

OUTPUT:

```
a = {0, 1, 4, 6, 8, 9}

<class 'set'>
```

6) Python Dictionary

```
d = {3:'val','meith':2}
print(type(d))
print("d[3] = ", d[3])
print("d['meith'] = ", d['meith'])
```

OUTPUT:

```
<class 'dict'>
d[3] =  val
d['meith'] =  2
```

7) Python Arrays

```
 import array as arr

# Array with Integers
a = arr.array('i' , [1,2,3])

print('Integer Array1 :' , a )

print('Elements of Array1 :' , end=" ")
for i in range(len(a)):
  print(a[i], end=" ")

print()

# Accessing by Index
print('Index = 2 :' , a[2] )
print('Negative Index= -2 :' , a[-2] )

# Index Slicing
print('\nIndex Slicing : ' , a[1:3])
print('Index Slicing : ' , a[2:])
print('Negative Index Slicing : ' , a[-3:-1])
print()

b = arr.array('i' , [1 ,2, 5 ,6 , 8])
print("Initial Array : " , b)
```

Insert , Append , Pop and Delete in Array

```
# Insert()
b.insert(1,4)
print('Updated Array after Insertion : ', b)

# Append()
b.append(11)
print('Updated Array after Append : ', b)

# POP
b.pop() # Removes element from the end.
print('Updated Array after POP : ' , b)

b.pop(2) # Removes the 2nd last element
print('Updated Array after POP of 2nd Last Index : ' , b)

# Delete
del b[3]
print('Updated Array after Deletion of index=3: ' , b)
```

Arrays Update

```
b = arr.array('i' , [1 ,2, 5 ,6 , 8])

print("Array before updation : ", b)
b[2] = 10
print("Array after updation : ", b)

#  Updating in loop
for i in range (len(b)):
    b[i] = b[i]*i

print("Array after updation : ", b)
```

Array Concatenation

```
a = arr.array('d',[1.1 , 2.1 ,3.1,2.6,7.8])
b = arr.array('d',[3.7,8.6])
c = arr.array('d')
c=a+b
print("Array c = ",c)
```

OUTPUT:

```
Integer Array1 : array('i', [1, 2, 3])
Elements of Array1 : 1 2 3
Index = 2 : 3
Negative Index= -2 : 2

Index Slicing :  array('i', [2, 3])
Index Slicing :  array('i', [3])
Negative Index Slicing :  array('i', [1, 2])
```

Insert , Append , Pop and Delete in Array

```
Initial Array :  array('i', [1, 2, 5, 6, 8])
Updated Array after Insertion :  array('i', [1, 4, 2, 5, 6, 8])
Updated Array after Append :  array('i', [1, 4, 2, 5, 6, 8, 11])
Updated Array after POP :  array('i', [1, 4, 2, 5, 6, 8])
Updated Array after POP of 2nd Last Index :  array('i', [1, 4, 5, 6,
8])
Updated Array after Deletion of index=3:  array('i', [1, 4, 5, 8])
```

Array Update

```
Array before updation :  array('i', [1, 2, 5, 6, 8])
Array after updation :  array('i', [1, 2, 10, 6, 8])
Array after updation :  array('i', [0, 2, 20, 18, 32])
```

Array Concatenation

```
Array c =  array('d', [1.1, 2.1, 3.1, 2.6, 7.8, 3.7, 8.6])
```

8) Python Conversions

```
# Int to Float
print(float(5))

# Float to Int
print(int(10.5))
print(int(-5.2))

# Float to String
print(str(5.2))

# String to List
print(list('hello'))
```

OUTPUT:

```
5.0
10
-5
5.2
['h', 'e', 'l', 'l', 'o']
```

Name : Meith Navlakha
SAP : 60004190068
Python Lab

9) Loops

A) For Loop

```
# Arrays
print('\nLooping through Array: ')
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)

#  String
print('\nLooping through String: ')
for x in "banana":
  print(x)

print('\nLooping with break')
for x in fruits:
  print(x)
  if x == "banana":
    break
```

OUTPUT:

```
Looping through Array:
apple
banana
cherry

Looping through String:
b
a
n
a
n
a

Looping with break
apple
banana
```

B) Nested For

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

OUTPUT:

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

C) For Else

```python
# for else
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

OUTPUT:

```
0

1

2

3

4

5

Finally finished!
```

D) While

```python
count = 0
while (count < 3):
    count = count + 1
    print(count)

count = 0
print('\n Printing Even Numbers in Loop: ')
while (count < 10):
   if count%2 ==0:
      print(count)

   count = count + 1
```

OUTPUT:

```
1

2

3


Printing Even Numbers in Loop:
0
2
4
6
8
```

10) Control Statements

a) break

```
for letter in 'thisismycode':
    if letter == 'i' or letter == 's':
         break
    print('Current Letter :', letter)
```

b) continue

```
for letter in 'thisismycode':
  if letter == 'i' or letter == 's':
       continue
  print('Current Letter :', letter)
```

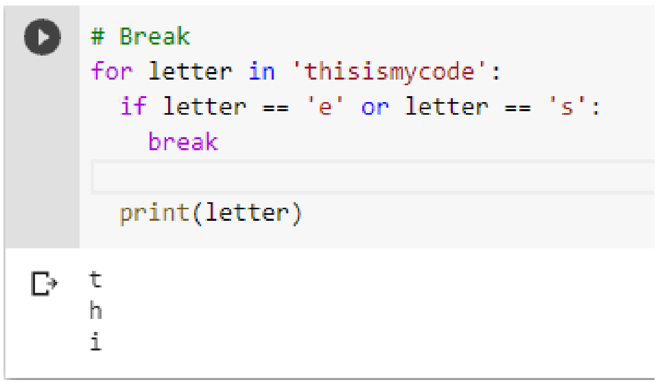
c) pass

```
for letter in 'thisismycode':
  if letter == 'i' or letter == 's':
       pass
  print('Current Letter :', letter)
```

Name : Meith Navlakha
SAP : 60004190068
Python Lab

OUTPUT

a) Break

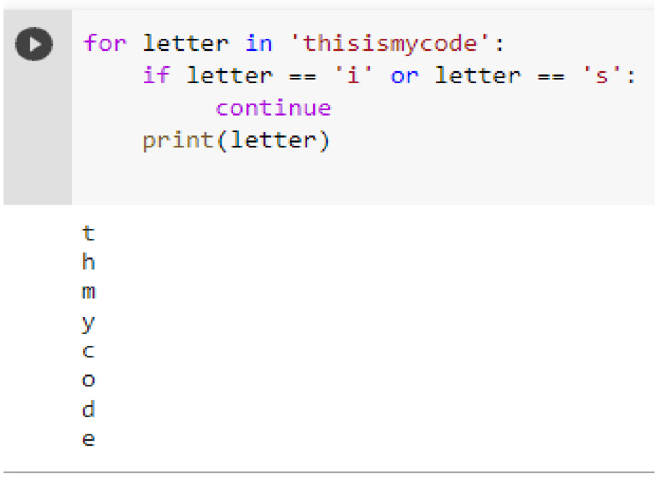


```
# Break
for letter in 'thisismycode':
  if letter == 'e' or letter == 's':
    break

  print(letter)
```

```
t
h
i
```

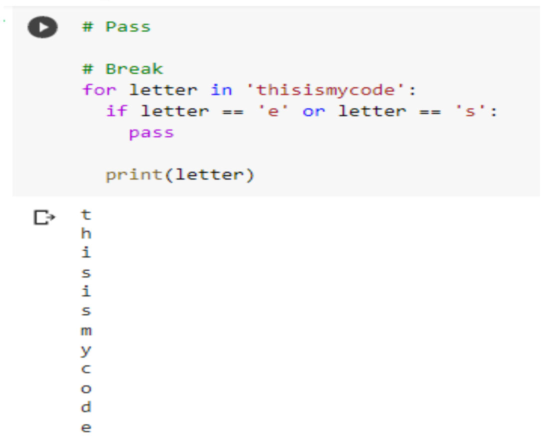
b) continue



```
for letter in 'thisismycode':
    if letter == 'i' or letter == 's':
        continue
    print(letter)
```

```
t
h
m
y
c
o
d
e
```

c) pass



```
# Pass

# Break
for letter in 'thisismycode':
  if letter == 'e' or letter == 's':
    pass

  print(letter)
```

```
t
h
i
s
i
s
m
y
c
o
d
e
```

Name : Meith Navlakha
SAP : 60004190068
Python Lab

CONCLUSION :

In this experiment we learnt about various data types in python programming. Python numbers which store numeric values – integers, float and complex. Python tuples and list which store a collection of data. Both are ordered sequence. The only difference between them is that tuples are immutable whereas lists are mutable. Python Sets which are a collection of unordered items. It only stores unique values i.e. does not duplicate entries. Python dictionary is an unordered collection of data values which stores the data in key-value pair. Python String is array of bytes representing Unicode characters. In python we have normal string and multiline string. Strings are immutable. Python arrays help in storing many data together in continuous memory location and it helps in storing the data together and access it easily. The data can be accessed easily by Indexing and are mutable. In loops we can execute a set of instructions repeatedly without repeating the code. Loops also have condition and other control statements to give us more control over the execution. Finally, we have Control Statements in python which are used to change the flow of execution for a certain condition. Python has 3 control statements viz, break, pass and continue.