PRACTICAL 1

Numbers

- a = 5
- print("The type of a", type(a))
-
- b = 40.5
- print("The type of b", type(b))
-
- c = 1+3j
- print("The type of c", type(c))
- print(" c is a complex number", isinstance(1+3j,complex))

Output:

```
The type of a <class 'int'>
The type of b <class 'float'>
The type of c <class 'complex'>
c is complex number: True
```

String

- str1 = 'hello Tanishk' #string str1
- str2 = ' how are you' #string str2
- print (str1[0:2]) #printing first two character using slice operator
- print (str1[4]) #printing 4th character of the string
- print (str1*2) #printing the string twice
- print (str1 + str2) #printing the concatenation of str1 and str2

Output:

```
he
o
hello Tanishkhello Tanishk
hello Tanishk how are you
```

List

- list1 = [1, "hi", "Python", 2]
- #Checking type of given list
- print(type(list1))
-
- #Printing the list1
- print (list1)
-
- # List slicing
- print (list1[3:])
-
- # List slicing
- print (list1[0:2])
-
- # List Concatenation using + operator
- print (list1 + list1)
-
- # List repetation using * operator
- print (list1 * 3)

Output:

```
[1, 'hi', 'Python', 2]
[2]
[1, 'hi']
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
```

Tuple

- tup = ("hi", "Python", 2)
- # Checking type of tup
- **print** (type(tup))
-
- #Printing the tuple
- **print** (tup)
-
- # Tuple slicing
- **print** (tup[1:])
- **print** (tup[0:1])
-
- # Tuple concatenation using + operator
- **print** (tup + tup)
-
- # Tuple repatation using * operator
- **print** (tup * 3)
-
- # Adding value to tup. It will throw an error.
- t[2] = "hi"

Output:

```
<class 'tuple'>
('hi', 'Python', 2)
('Python', 2)
('hi',)
('hi', 'Python', 2, 'hi', 'Python', 2)
('hi', 'Python', 2, 'hi', 'Python', 2, 'hi', 'Python', 2)

Traceback (most recent call last):
  File "main.py", line 14, in <module>
    t[2] = "hi";
TypeError: 'tuple' object does not support item assignment
```

Dictionary

- d = {1:'Tanishk', 2:'Shreesh', 3:'Rishav', 4:'Aditya'}
-
- # Printing dictionary
- **print** (d)
-
- # Accesing value using keys
- **print**("1st name is "+d[1])
- **print**("2nd name is "+ d[4])
-
- **print** (d.keys())
- **print** (d.values())

Output:

```
1st name is Tanishk
2nd name is Aditya
{1: 'Tanishk', 2: 'Shreesh', 3: 'Rishav', 4: 'Aditya'}
dict_keys([1, 2, 3, 4])
dict_values(['Tanishk', 'Shreesh', 'Rishav', 'Aditya'])
```

Boolean

- # Python program to check the boolean type
- **print**(type(True))
- **print**(type(False))
- **print**(false)

Output:

```
<class 'bool'>
<class 'bool'>
NameError: name 'false' is not defined
```

Set

- # Creating Empty set
- set1 = set()
-
- set2 = {'James', 2, 3,'Python'}
-
- #Printing Set value
- print(set2)
-
- # Adding element to the set
-
- set2.add(10)
- print(set2)
-
- #Removing element from the set
- set2.remove(2)
- print(set2)

Output:

```
{3, 'Python', 'James', 2}
{'Python', 'James', 3, 2, 10}
{'Python', 'James', 3, 10}
```

PRACTICAL 2

CONDITIONAL STATEMENT

If Statement

```
•  if <conditional expression>
•   Statement
•  else
•   Statement
```

Code

```
•  # Python program to execute if statement
•
•  a, b = 6, 5
•
•  # Initializing the if condition
•  if a > b:
•      code = "a is greater than b"
•      print(code)
```

Output:

```
a is greater than b
```

Else Statement

```
•  # Python program when else condition does not work
•
•  a, b = 9, 9
•
•  # Initializing the if-else condition
•  if a < b:
•      code = "a is less than b"
```

- else:
-     code = "a is greater than b"
-
- print(code)

Output:

```
a is greater than b
```

Elif Condition

- # Python program to show how to use elif condition
-
- a, b = 9, 9
-
- # Initializing the if-else condition
- if a < b:
-     code = "a is less than b"
- elif a == b:
-     code = "a is equal to b"
- else:
-     code = "a is greater than b"
-
- print(code)

Output:

```
a is equal to b
```

Nested if Statement

```
x = 10
y = 5
if x > 0:
   print("x is positive")
```

```
    if y > 0:
        print("y is also positive")
    else:
        print("y is not positive")
```

Output:

x is positive , y is also positive

FLOW CONTROL STATEMENT

If Statement

```
# Python program to show how if statements control loops

n = 5
for i in range(n):
    if i < 2:
        i += 1
    if i > 2:
        i -= 2
    print(i)
```

Output:

```
1
2
2
1
2
```

Break Statements

```
# Python program to show how to control the flow of loops with the break sta
tement
```

-
- Details = [[19, 'tanishk', 'kolkata'], [16, 'shreesh', 'delhi']]
- for candidate in Details:
-     age = candidate[0]
-     if age <= 18:
-         break
-     print (f"{candidate[1]} of state {candidate[2]} is eligible to vote")

Output:

```
tanishk of state kolkata is eligible to vote
```

Continue Statements

- # Python program to show how to control the flow of a loop using a continue statement
- # Printing only the letters of the string
- for l in 'I am a coder':
-     if l == ' ':
-         continue
-     print ('Letter: ', l)

Output:

```
Letter:  I
Letter:  a
Letter:  m
Letter:  a
Letter:  c
Letter:  o
Letter:  d
Letter:  e
Letter:  r
```

PRACTICAL 3

Parameterized Function

```
def add_numbers(x, y):
    """
    This function adds two numbers and returns the result.

    Parameters:
    x (int): The first number.
    y (int): The second number.

    Returns:
    int: The sum of x and y.
    """
    return x + y

# Call the function with specific numbers
result = add_numbers(5, 3)
print("The sum is:", result)
```

Output:

```
The sum is: 8
```

Non-Parameterized Function

```
def greet():
    """
    This function prints a simple greeting message.
    """
    print("Hello, World!")

# Call the function
greet()
```

Output:

```
 Hello, World!
```

PRACTICAL 4

Bubble Sort

```
def bubbleSort(arr):
        n = len(arr)
        # For loop to traverse through all element in an array
        for i in range(n):
          for j in range(0, n - i - 1):

                # Range of the array is from 0 to n-i-1
                # Swap the elements if the element found
                #is greater than the adjacent element
                if arr[j] > arr[j + 1]:
                     arr[j], arr[j + 1] = arr[j + 1], arr[j]
# Example to test the above code
arr = [ 2, 1, 10, 23 ]
bubbleSort(arr)
print("Sorted array is:")
for i in range(len(arr)):
        print("%d" % arr[i])
```

Output

```
Sorted array is:
1
```

```
2
10
23
```

Selection Sort

```
# Selection Sort algorithm in Python
def selectionSort(array, size):
        for s in range(size):
          min_idx = s
          for i in range(s + 1, size):
               # For sorting in descending order
               # for minimum element in each loop
               if array[i] < array[min_idx]:
                    min_idx = i
          # Arranging min at the correct position
          (array[s], array[min_idx]) = (array[min_idx], array[s])
# Driver code
data = [ 7, 2, 1, 6 ]
size = len(data)
selectionSort(data, size)
print('Sorted Array in Ascending Order is :')
print(data)
```

Output

```
Sorted Array in Ascending Order is :
[1, 2, 6, 7]
```

Insertion Sort

```
# Creating a function for insertion sort algorithm
def insertion_sort(list1):
        # Outer loop to traverse on len(list1)
        for i in range(1, len(list1)):
            a = list1[i]
            # Move elements of list1[0 to i-1],
            # which are greater to one position
            # ahead of their current position
            j = i - 1
            while j >= 0 and a < list1[j]:
                list1[j + 1] = list1[j]
                j -= 1
            list1[j + 1] = a
        return list1
# Driver code
list1 = [ 7, 2, 1, 6 ]
print("The unsorted list is:", list1)
print("The sorted new list is:", insertion_sort(list1))
```

Output

```
The unsorted list is: [7, 2, 1, 6]
The sorted new list is: [1, 2, 6, 7]
```

PRACTICAL 5

Pandas

```
# Importing pandas library
import pandas as pd
# Creating and initializing a nested list
age = [['Aman', 95.5, "Male"], ['Sunny', 65.7, "Female"],
            ['Monty', 85.1, "Male"], ['toni', 75.4, "Male"]]
# Creating a pandas dataframe
df = pd.DataFrame(age, columns=['Name', 'Marks', 'Gender'])
# Printing dataframe
df
```

Output:

```
   Name    Marks    Gender
0   Aman    95.5    Male
1   Sunny    65.7    Female
2   Monty    85.1    Male
3   toni    75.4    Male
```

NUMPY

```
import numpy as np
# Create two NumPy arrays
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
```

```
# Perform element-wise addition
result_addition = array1 + array2
# Display the original arrays and the results
print("Array 1:", array1)
print("Array 2:", array2)
print("Element-wise Addition:", result_addition)
```

Output:

```
Array 1: [1 2 3]
Array 2: [4 5 6]
Element-wise Addition: [5 7 9]
```

Matplotlib

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.plot(xpoints, ypoints)
plt.show()
```

Result