Experiment No1

October 28, 2024

[1]:
```
print("Demo of basic data types: Numbers")
x = 3
y = 2.5
print("x = ",x)
print("y = ",y)
print("Datatype of variable x: ",type(x))
print("Datatype of variable y: ",type(y))
print("Addition: ",x+y)
print("Subtraction: ",x-y)
print("Mutiplication: ",x*y)
print("Exponentiation: ",x**2)
```

```
Demo of basic data types: Numbers
x =  3
y =  2.5
Datatype of variable x:  <class 'int'>
Datatype of variable y:  <class 'float'>
Addition:  5.5
Subtraction:  0.5
Mutiplication:  7.5
Exponentiation:  9
```

[2]:
```
print("Demo of basic data types: Boolean")
t = True
f = False
print("t = ",t)
print("f = ",f)
print("Data type of variable t:",type(t))
print("Data type of variable f:",type(f))
print("Logical AND operation:",t and f)
print("Logical OR operation:",t or f)
print("Logical NOT operation:",not t)
print("Logical XOR operation:",t != f)
```

```
Demo of basic data types: Boolean
t =  True
f =  False
Data type of variable t: <class 'bool'>
```

```
Data type of variable f: <class 'bool'>
Logical AND operation: False
Logical OR operation: True
Logical NOT operation: False
Logical XOR operation: True
```

[3]:
```
print("Demo of basic data types: String")
s = "Hello"
t = "World"
print("String1 = ",s)
print("String2 = ",t)
d = s+","+t
print("String Concantenation:",d)
print("Capitalize: ",d.capitalize())
print("Converted to Uppercase: ",s.upper())
print("Right justify a string: ",s.rjust(7))
print("String at center: ",s.center(7))
print("After replacing l with ell:  ",s.replace('l','(ell)'))
print("String after striping leading to and trailling white spaces : ",'world '.
 ↪strip())
```

```
Demo of basic data types: String
String1 =  Hello
String2 =  World
String Concantenation: Hello,World
Capitalize:  Hello,world
Converted to Uppercase:  HELLO
Right justify a string:    Hello
String at center:   Hello
After replacing l with ell:   He(ell)(ell)o
String after striping leading to and trailling white spaces :  world
```

[4]:
```
print("Containers:Lists")
nums = list(range(5))
print("List 'nums' contains:",nums)
nums[4] ='abc'
print("List can contain elements of different types. Example: ",nums)
nums.append("xyz")
print("'nums' after inserting a new element a the end: ")
print("Sublists:")
print("A slice from index 2 to 4: ",nums[2:4])
print("A slice from index 2 to the end: ",nums[2:])
print("A slice from  start index to the end: ",nums[:2])
print("SA Slice of the whole list: ",nums[:])
nums[4:] = [8,9]
print("After assigning a new sublist to nums:")
for idx, i in enumerate(nums):
```

```
    print('%d:%s' %(idx+1, idx))
even_squares = [x**2 for x in nums if x%2==0]
print("List of squares of even numbers from 'nums'",even_squares)
```

```
Containers:Lists
List 'nums' contains: [0, 1, 2, 3, 4]
List can contain elements of different types. Example:  [0, 1, 2, 3, 'abc']
'nums' after inserting a new element a the end:
Sublists:
A slice from index 2 to 4:  [2, 3]
A slice from index 2 to the end:  [2, 3, 'abc', 'xyz']
A slice from  start index to the end:  [0, 1]
SA Slice of the whole list:  [0, 1, 2, 3, 'abc', 'xyz']
After assigning a new sublist to nums:
1:0
2:1
3:2
4:3
5:4
6:5
List of squares of even numbers from 'nums' [0, 4, 64]
```

[5]:
```
print("Containers:Dictionaries")
d= dict()
d = {'cat':'cute', 'dog':'furry'}
print("Dictionary: ",d)
print("Is the dictionary has the key 'cat'?",'cat' in d)
d['fish'] = 'wet'
print("After adding new entry to 'd': ",d)
print("Get an element 'monkey':", d.get('monkey',"N/A"))
print("Get an element 'fish':", d.get('fish',"N/A"))
del d['fish']
print("After deleting the newly added entry from 'd': ",d)
print("Demo of dictionary comprehension: ")
squares = {x:x*x for x in range(10)}
print("Squares of integers of range 10:")
for k,v in squares.items():
    print(k," ", v)
```

```
Containers:Dictionaries
Dictionary:  {'cat': 'cute', 'dog': 'furry'}
Is the dictionary has the key 'cat'? True
After adding new entry to 'd':  {'cat': 'cute', 'dog': 'furry', 'fish': 'wet'}
Get an element 'monkey': N/A
Get an element 'fish': wet
After deleting the newly added entry from 'd':  {'cat': 'cute', 'dog': 'furry'}
Demo of dictionary comprehension:
Squares of integers of range 10:
```

```
0   0
1   1
2   4
3   9
4   16
5   25
6   36
7   49
8   64
9   81
```

[6]:
```
print("Containers:Sets")
num1 = {100,110,120}
print("Set'num1': ",num1)
num1.add(90)
print("'num1' after inserting 90: ",num1)
num1.update([50,60,70])
print("'num1' after inserting multiple elements: ",num1)
num1.remove(60)
print("'num1' after removing 60: ",num1)
print("Set comprehension and set options:")
n1 = {x for x in range(10)}
print("n1 = ",n1)
n2 = {x for x in range(10) if x%2!=0}
print("n2 = ",n2)
print("n1 union n2: ",n1|n2)
print("n1 intersection n2: ",n1&n2)
print("n1 difference n2: ",n1-n2)
```

```
Containers:Sets
Set'num1':  {120, 100, 110}
'num1' after inserting 90:  {120, 90, 100, 110}
'num1' after inserting multiple elements:  {100, 70, 110, 50, 120, 90, 60}
'num1' after removing 60:  {100, 70, 110, 50, 120, 90}
Set comprehension and set options:
n1 =  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
n2 =  {1, 3, 5, 7, 9}
n1 union n2:  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
n1 intersection n2:  {1, 3, 5, 7, 9}
n1 difference n2:  {0, 2, 4, 6, 8}
```

[7]:
```
print("CONTAINERS : TUPLES")
d = {(x,x+1):x for x in range(10)}
print("Dictionary with tuple keys: ")
for k,v in d.items():
    print(k," : ",v)
t = (5,6)
```

```
print("Tuple t: ",t)
print(d[t])
print(d[1,2])
```

```
CONTAINERS : TUPLES
Dictionary with tuple keys:
(0, 1)  :  0
(1, 2)  :  1
(2, 3)  :  2
(3, 4)  :  3
(4, 5)  :  4
(5, 6)  :  5
(6, 7)  :  6
(7, 8)  :  7
(8, 9)  :  8
(9, 10)  :  9
Tuple t:  (5, 6)
5
1
```

[8]:
```
print("Demo of function: Program to find factorial of a number")
def fact(n):
    if n == 1:
        return 1
    else:
        return(n*fact(n-1))
n = int(input("Enter a number: "))
print("Factorial: ",fact(n))
```

```
Demo of function: Program to find factorial of a number

Enter a number:  5

Factorial:  120
```

[9]:
```
class Greeter:
    def __init__(self,name):
        self.name = name
    def greet(self,loud=False):
        if loud:
            print('HELLO,%s!'%self.name.upper())
        else:
            print('Hello,%s'%self.name)
g = Greeter('Fred')
g.greet()
g.greet(loud=True)
```

```
Hello,Fred
HELLO,FRED!
```

[ ]: