Name: Maheen fatima

Roll No: BIT-24S-033

GitHub link: https://github.com/maheen6327

LAB: 01

Task 1:

Make 2-2 programs of each datatype.

- NUMERIC TYPES:
- ❖ Integer:

```
num1 = 4
num2 = 8
sum = num1 + num2
print("The sum is:", sum)
The sum is: 12
```

```
if num % 2 == 0:
    print("The number is even")
else:
    print("The number is odd")
```

The number is even

❖ Float:

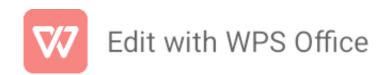
```
num1 = 3.5
num2 = 8.5
result = num1 - num2
print("The result is:", result)
```

The result is: -5.0

```
num1 = 1.0
num2 = 5.0
result = num1 / num2
print("The answer is:", result)
```

The answer is: 0.2

Complex:



```
num1 = 4 + 4j
num2 = 5 + 4j
result = num1 + num2
print("The sum is:", result)

The sum is: (9+8j)

num1 = 6 + 3j
num2 = 2 + 2j

result = num1 * num2

print("The product is:", result)

The product is: (6+18j)
```

• SEQUENCE TYPES:

String

```
first_name = "Maheen"
last_name = "Fatima"
full_name = first_name + " " + last_name
print("Full name is:", full_name)

Full name is: Maheen Fatima

message = "Hello, world!"
print(message)

Hello, world!

List:
```



```
fruits = ["orange", "banana", "watermelon"]
for fruit in fruits:
    print(fruit)

orange
banana
watermelon

numbers = [1, 2, 3]
numbers.append(5)
print("Updated list:", numbers)
```

opuateu 11st. [1, 2, 3, 3

❖ Tuple:

```
colors = ("pink", "brown", "blue")
print("Second color is:", colors[1])

Second color is: brown

fruits = ("grapes", "banana", "cherry")

for fruit in fruits:
    print(fruit)

grapes
banana
cherry
```

* Range:



```
11]: for num in range(1, 6):
    print(num)

1
2
3
4
5

12]: for num in range(2, 11, 2):
    print(num)

2
4
6
8
10
```

• SET TYPES:

* Set:

```
fruits = {"apple", "banana", "mango"}

print("Fruits set:", fruits)

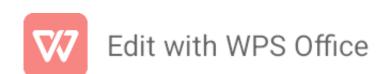
Fruits set: {'apple', 'banana', 'mango'}

numbers = {1, 2, 3}
numbers.add(4)

print("Updated set:", numbers)

Updated set: {1, 2, 3, 4}
```

❖ Frozen set:



```
5]: num = frozenset([1,2,3,4])
    print("num frozenset:", num)

num frozenset: frozenset({1, 2, 3, 4})

6]: set1 = frozenset([1, 2, 3])
    set2 = frozenset([3, 4, 5])
    common = set1.intersection(set2)

print("Common items:", common)

Common items: frozenset({3})
```

MAPPING TYPE:

Dictionary dict:

```
student = {
    "name": "maheen",
    "age": 18,
    "class": "BS IT"
}

print("Student Info:", student)

Student Info: {'name': 'maheen', 'age': 18, 'class': 'BS IT'}

person = {
    "name": "maheen",
    "city": "Karachi"
}

print("Name is:",person["name"])

Name is: maheen
```

• BOLEAN TYPE:



```
a = 30
b = 10

result = a > b
print("Is a greater than b?", result)

Is a greater than b? True

i: x = 10
y = 10

print("Are x and y equal?", x == y)

Are x and y equal? True
```

Task 2:

Make up to 5 shapes programs using *

```
print("Square Shape:")
print("* * * * *")
print("Right-Angled Triangle:")
print("*")
print("* *")
print("* * *")
print("* * * *")
print("* * * * *")
print("Inverted Triangle:")
print("* * * * *")
print("* * * *")
print("* * *")
print("* *")
print("*")
```

```
Square Shape:
 Right-Angled Triangle:
 Inverted Triangle:
print("Pyramid Shape:")
print(" *")
print(" * *")
print(" * * *")
print(" * * * *")
print("* * * * *")
print("Diamond Shape:")
print(" *")
print("
print(" * * *")
print(" * *")
print(" *")
   Pyramid Shape:
   Diamond Shape:
```

Task 3:

Make same shapes you have made in task 2, using * multiple by number.

Program:

```
|: | print("Square Shape:")
   print("* " * 5)
   print("Right-Angled Triangle:")
   print("* " * 1)
   print("* " * 2)
   print("* " * 3)
   print("* " * 4)
   print("* " * 5)
   print("Inverted Triangle:")
   print("* " * 5)
   print("* " * 4)
   print("* " * 3)
   print("* " * 2)
   print("* " * 1)
   print("Pyramid Shape:")
   print(" " * 4 + "* ")
   print(" " * 3 + "* " * 2)
   print(" " * 2 + "* " * 3)
   print(" " * 1 + "* " * 4)
   print(" " * 0 + "* " * 5)
print("Diamond Shape:")
print(" " * 4 + "* ")
print(" " * 3 + "* " * 2)
print(" " * 2 + "* " * 3)
print(" " * 3 + "* " * 2)
print(" " * 4 + "* ")
```

```
Square Shape:

* * * * * *

* * * * *

* * * * *

Right-Angled Triangle:

* * * *

* * * *

Inverted Triangle:

* * * *

* * * *

* * *

* * *

Pyramid Shape:

* * *

* * *

* * *

* * *

* * *

* * *

Diamond Shape:

* *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* * *

* *

* * *

* * *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

*
```