

Experiment 1.2

Aim: To understand the basic data types, operators, expressions and input output statements.

Theory:

- A. Data types: A data type defines the type of data. The data types in python are divided in two categories:
 - 1. Immutable data types - Values cannot be changed, which are :- Numbers, string, tuple.
 - 2. Mutable data types - Values can be changed, which are :- List, Dictionaries, sets
- 1. Number / Numeric Data Type

- a. Integer - In Python, integers are zero, positive or negative whole numbers without a fractional part and having unlimited precision.

Ex:

Input

num = 100

print(num)

Output

100

>

- b. Long - A long is an integer type value that has unlimited length.

- c. Float - Values with decimal points are the float values, there is no need to specify the data type in python.

Ex.

Input

fnum = 34.45

print(fnum)

Output

34.45

>

d. Complex number : Numbers with real and imaginary parts are known as complex numbers. Python is able to identify these complex numbers with values.

Ex.

Input :	Output
<code>(num = 3+4j)</code>	<code>(3+4j)</code>
<code>print(num)</code>	>

e. Binary, Octal and Hexadecimal

Binary : `0b(zero + 'b')` and `0B(zero + 'B')`

Ex.

Input	Output
<code>num = 0b101</code>	5
<code>print(num)</code>	

Octal : `0o(zero + 'o')` and `0O(zero + 'O')`

Ex.

Input	Output
<code>num = 0o32</code>	26
<code>print(num)</code>	

Hex decimal : `0x(zero + 'x')` and `0X(zero + 'X')`

Ex.

Input	Output
<code>num = 0xFF</code>	255
<code>print(num)</code>	

2. String data type : String is a sequence of characters in python. The data type of string in python is called "str".

Ex.

Input	Output
<code>s = "This is a string"</code>	<code>This is a string</code>
<code>print(s)</code>	

3. Tuple data type - It is an ordered collection of elements enclosed in round brackets and separated by commas.

Ex. Input

```
t = (1, 2, 3, 4, 5)
print(t)
```

Output

(1, 2, 3, 4, 5)

4. List data type - It is similar to tuple, but list is enclosed in square bracket and elements are separated by commas.

Ex:

Input

```
lis = [1, 2, 3, 4, 5]
```

```
print(lis)
```

Output

[1, 2, 3, 4, 5]

5. Dictionary data type - Dictionary is a collection of key and value pairs. A dictionary does not allow duplicate keys but values can be duplicate.

Ex:

Input

```
dic = {1: 'apple', 2: 'ball'}
```

```
print(dic)
```

Output

{1: 'apple', 2: 'ball'}

6. Set data type - Sets are used to store multiple items in a single variable. A set is an unordered and unindexed collection of items.

Ex:

Input

```
set = {2, 'bye'}
```

```
print(set)
```

Output

{2, 'bye'}

B.

Operators: Operators are used to perform operations on variables and values. Python divides the operators in the following groups:

1. Arithmetic operators

Operator	Name	Example
+	Addition	$x+y$
-	Subtraction	$x-y$
*	multiplication	$x*y$
/	Division	x/y
%	modulus	$x \% y$
**	Exponentiation	$x^{**}y$
//	Floor division	$x//y$

2. Assignment operators

Operator	Example
=	$y = a+b$
+=	$m += 10$
-=	$m -= 10$
*=	$m *= 10$
/=	$m /= 10$
%=	$m \% = 10$
**=	$m ** = 10$
//=	$m // = 10$

3. Logical operators:

Operator	Description	Example
and	Returns true if both statements are true	$x < 5$ and $x < 10$
or	Returns true if one of the statement is true	$x < 5$ or $x < 4$.
not	Reverse the result, returns False if true	not($x < 5$ and $x < 4$).

4. Membership operators:

Operator	Description	Example
in	Returns true if a sequence with specified value is present	$x \text{ in } y$
not in	Returns true if a sequence with specified value is not present	$x \text{ not in } y$

Example: Input

```
x = ["apple", "banana"]
print ("banana" in x)
```

Output : True

5. Bitwise operators:

Operator	Name
&	AND
	OR
^	XOR
~	NOT
<<	zero fill left shift
>>	Signed right shift

C. Expressions:

Expressions are representations of value. They are different from statements in the fact that statement do something while expressions are representation of value. Python expressions only contain identifiers, literals and operators.

D. Input and Output functions:

taking input from user. Sometimes a developer might want to take input from the user at some point of program. To do this python provides an `input()` function. To show the output in python, we can use `print("")` function to display the output on prompt.

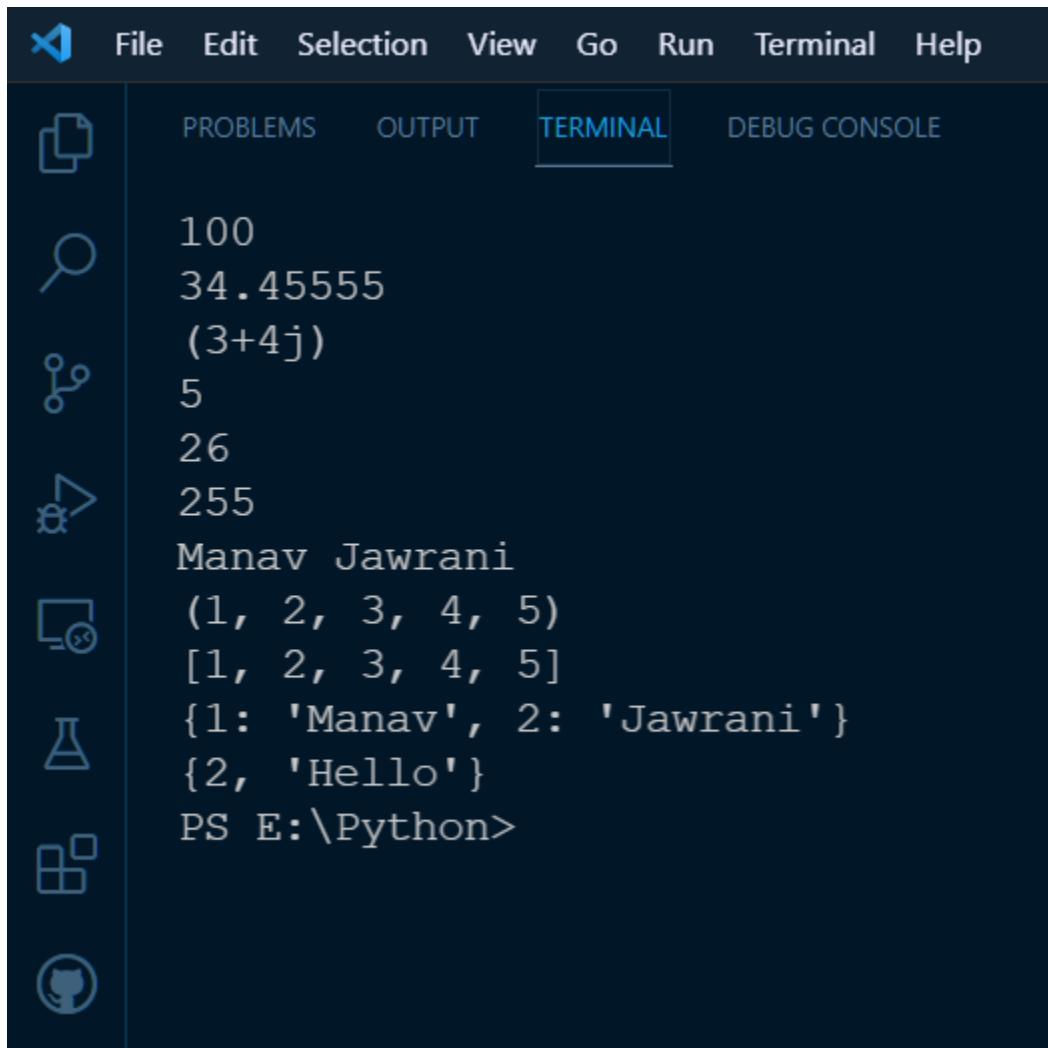
Programs:

A. Data types

```
datatype.py > ...
1 num=100      #integer#
2 print(num)
3
4 fnum=34.45555  #floating number#
5 print(fnum)
6
7 cnum=3+4j    #complex number#
8 print(cnum)
9
10 bnum=0b101   #binary number#
11 print(bnum)
12
13 onum=0o32    #octal number#
14 print(onum)
15
16 hnum=0xFF    #hexadecimal number#
17 print(hnum)
18
19 str="Manav Jawrani"  #string#
20 print(str)
21
22 tup=(1,2,3,4,5)    #tuple#
23 print(tup)
24
25 lis=[1,2,3,4,5]    #list#
26 print(lis)
27
28 dic={1:'Manav', 2:'Jawrani'}  #dictionary#
29 print(dic)
30
31 set={2,'Hello'}    #set#
32 print(set)
33
34
```

Python 3.10.1 64-bit ⑧ 0 Δ 0 ⌂ Manav 🔒 Live Share

Output:



A screenshot of a terminal window in Visual Studio Code. The window title is "Terminal". The content of the terminal shows the following Python code execution:

```
100
34.45555
(3+4j)
5
26
255
Manav Jawrani
(1, 2, 3, 4, 5)
[1, 2, 3, 4, 5]
{1: 'Manav', 2: 'Jawrani'}
{2, 'Hello'}
PS E:\Python>
```

B. Operators

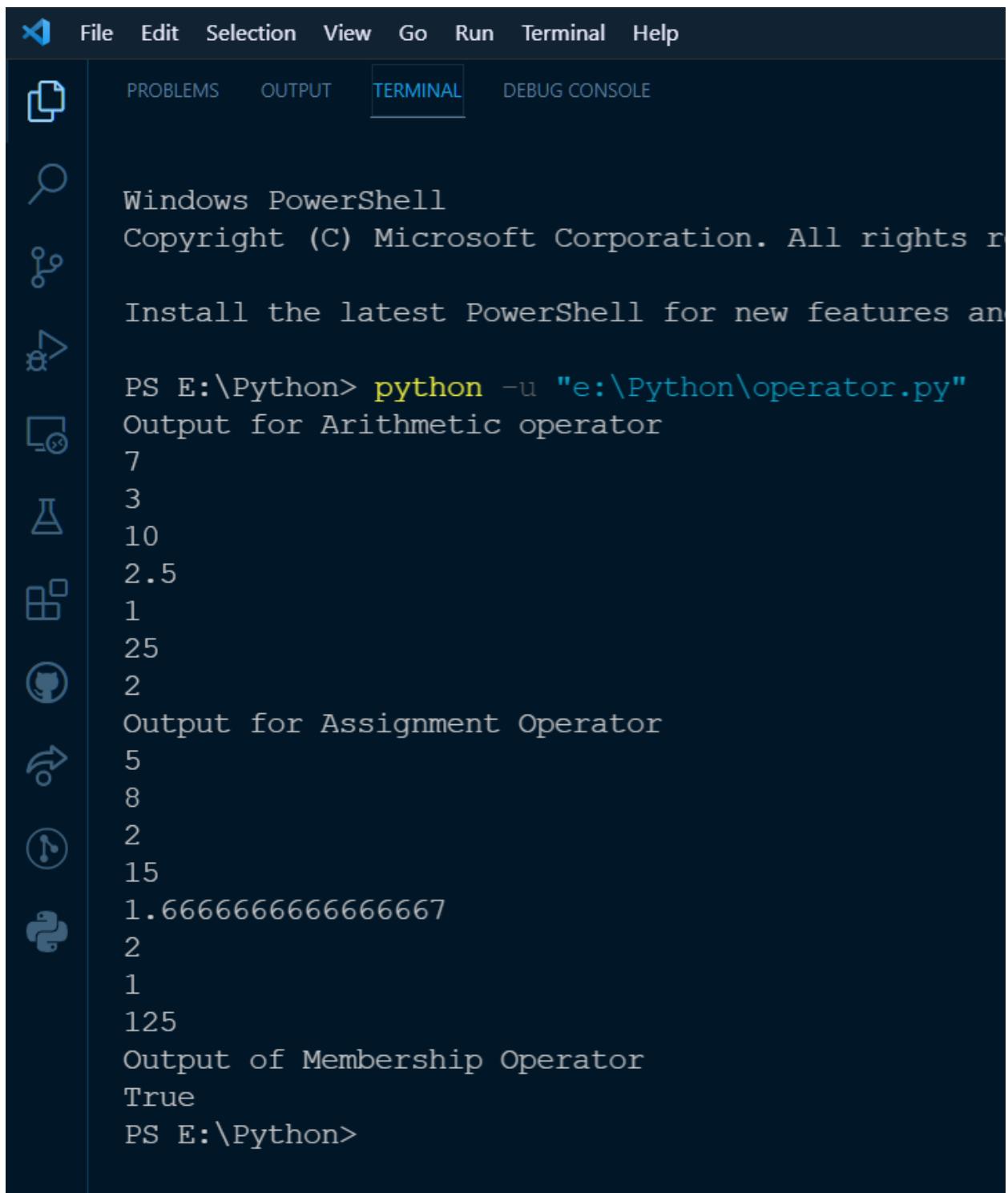
The screenshot shows a Python code editor interface with a dark theme. On the left is a vertical toolbar with various icons. The main area displays a Python script named `operator.py`. The code is color-coded for syntax:

```
1  #Arithmetic Operators#
2
3  print("Output for Arithmetic operator")
4  x = 5
5  y = 2
6  print(x + y)
7  print(x - y)
8  print(x*y)
9  print(x/y)
10 print(x % y)
11 print(x**y)
12 print(x//y)
13
14 #Assignment Operators#
15
16 print("Output for Assignment Operator")
17 x = 5
18 print(x)
19 x = 5
20 x += 3
21 print(x)
22 x = 5
23 x -= 3
24 print(x)
25 x = 5
26 x *= 3
27 print(x)
28 x = 5
29 x /= 3
30 print(x)
31 x = 5
32 x = x % 3
33 print(x)
34 x = 5
```

A screenshot of a Python code editor interface. The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar contains various icons for file operations like copy, paste, search, and refresh. The main pane shows a script named 'operator.py' with the following code:

```
operator.py
operator.py > ...
13
14     #Assignment Operators#
15
16     print("Output for Assignment Operator")
17     x = 5
18     print(x)
19     x = 5
20     x += 3
21     print(x)
22     x = 5
23     x -= 3
24     print(x)
25     x = 5
26     x *= 3
27     print(x)
28     x = 5
29     x /= 3
30     print(x)
31     x = 5
32     x = x % 3
33     print(x)
34     x = 5
35     x//=3
36     print(x)
37     x = 5
38     x **= 3
39     print[x]
40
41     #Membership Operator#
42
43     print("Output of Membership Operator")
44     x=["apple","pineapple"]
45     print("pineapple" in x)
```

Output:



The screenshot shows a terminal window in Visual Studio Code. The terminal tab is selected, and the output is displayed in white text on a dark background. The terminal shows the execution of a Python script named 'operator.py' located at 'E:\Python'. The script demonstrates various operators: arithmetic operators (7, 3, 10, 2.5, 1, 25), assignment operators (5, 8, 2, 15), and membership operators (True). The terminal concludes with the prompt 'PS E:\Python>'.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improved security.

PS E:\Python> python -u "e:\Python\operator.py"
Output for Arithmetic operator
7
3
10
2.5
1
25
2
Output for Assignment Operator
5
8
2
15
1.6666666666666667
2
1
125
Output of Membership Operator
True
PS E:\Python>
```

Conclusion: We have understood different data types , operators , expressions and input output statements.