CAP-788 DATA SCIENCE TOOLBOX LAB (CA-2)

SUBMITTED TO:
Dr. APASH ROY
(23550)

SUBMITTED BY:
YASH AGARWAL
MCA-DE566
RDE566A03
12000323

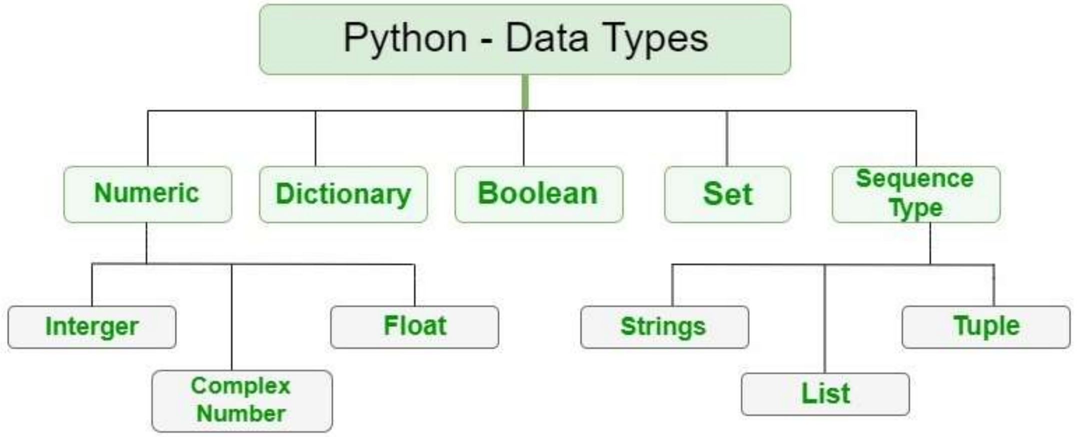
Q. Make a tutorial presentation/ documentation on python for the following topics: introduction, data types, type casting, variables, operator. (Use screenshots of full screen for each step, so that it can be differentiated from other person).

WHAT IS PYTHON?

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

DATA TYPES OF PYTHON

- Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.



TYPE CASTING IN PYTHON

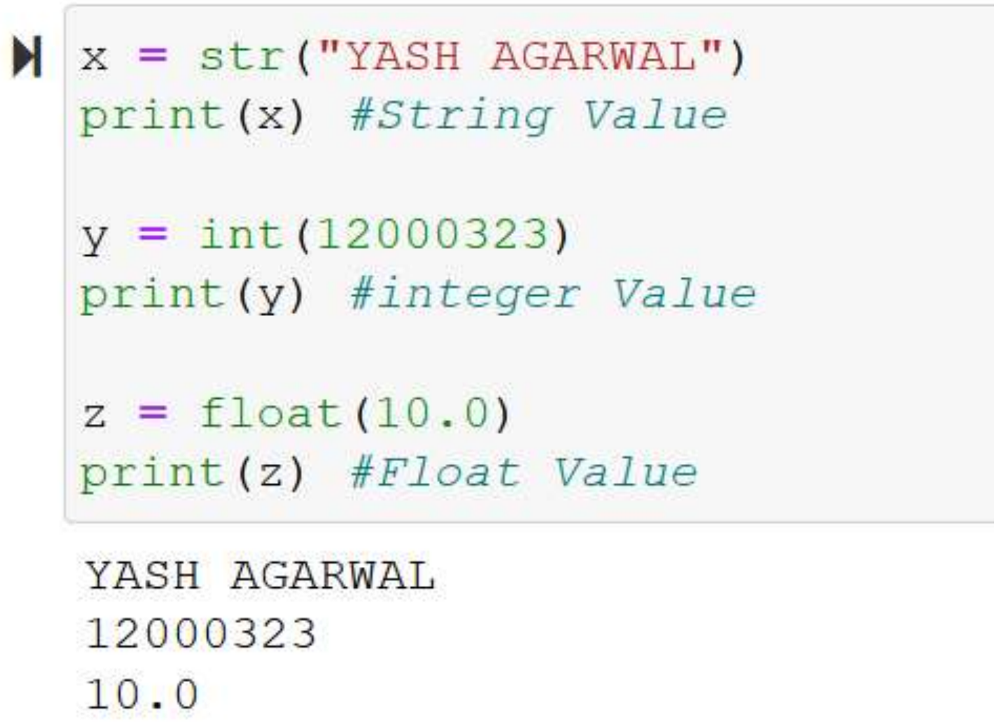
- There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.
- Casting in python is therefore done using constructor functions:
- **int() -** constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
- **float() -** constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- **str() -** constructs a string from a wide variety of data types, including strings, integer literals and float literals

TYPE CASTING IN PYTHON (EXAMPLE)

```
x = str("YASH AGARWAL")
print(x)            #String Value

y = int(12000323)
print(y)            #integer Value

z = float(10.0)
print(z)            #Float Value
```



```
x = str("YASH AGARWAL")
print(x) #String Value

y = int(12000323)
print(y) #integer Value

z = float(10.0)
print(z) #Float Value
```

```
YASH AGARWAL
12000323
10.0
```

VARIABLES IN PYTHON

- A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.
- Every value in Python has a datatype. Different data types in Python are Numbers, List, Tuple, Strings, Dictionary, etc. Variables can be declared by any name or even alphabets like a, aa, abc, etc.

OPERATORS IN PYTHON

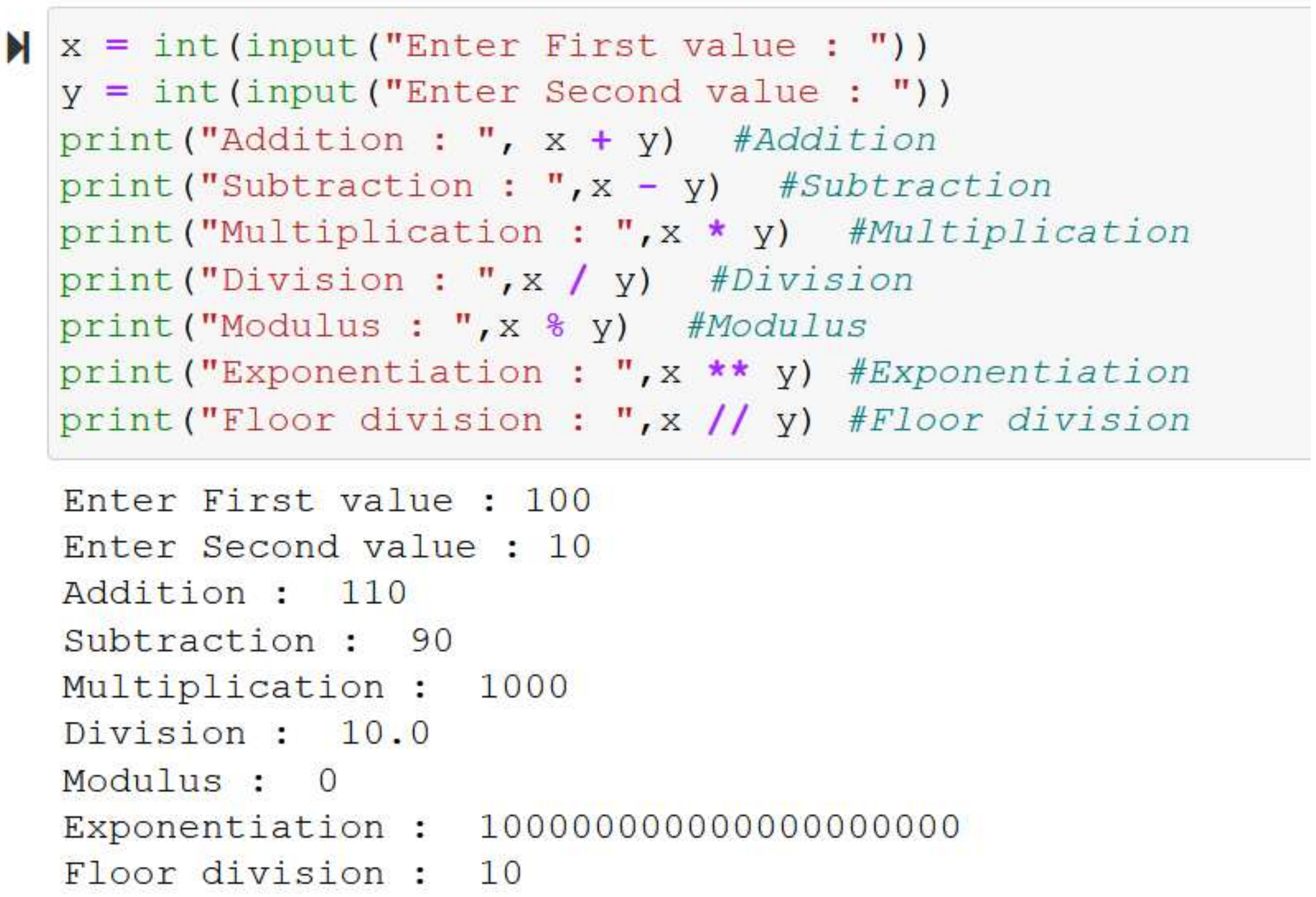
Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

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

EXAMPLE :



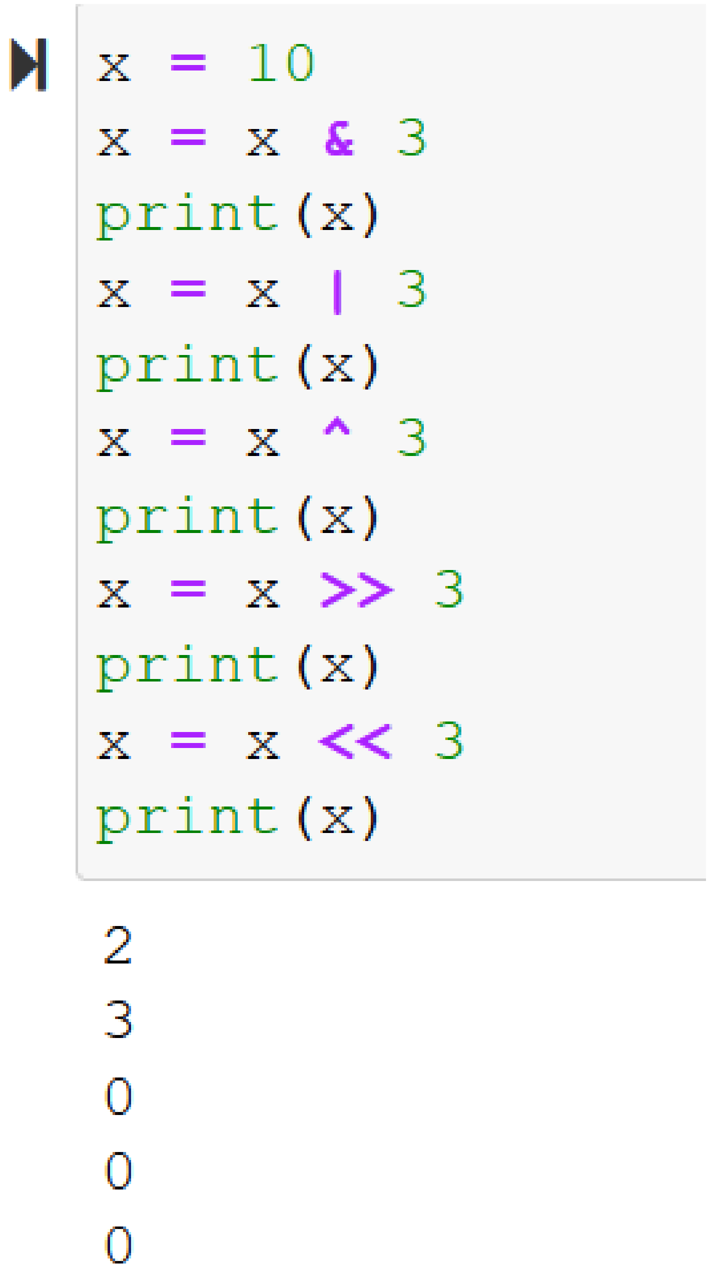
```
x = int(input("Enter First value : "))
y = int(input("Enter Second value : "))
print("Addition : ", x + y)  #Addition
print("Subtraction : ",x - y)  #Subtraction
print("Multiplication : ",x * y)  #Multiplication
print("Division : ",x / y)  #Division
print("Modulus : ",x % y)  #Modulus
print("Exponentiation : ",x ** y) #Exponentiation
print("Floor division : ",x // y) #Floor division
```

```
Enter First value : 100
Enter Second value : 10
Addition :  110
Subtraction :  90
Multiplication :  1000
Division :  10.0
Modulus :  0
Exponentiation :  100000000000000000000
Floor division :  10
```

ASSIGNMENT OPERATORS

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x – 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

EXAMPLE :



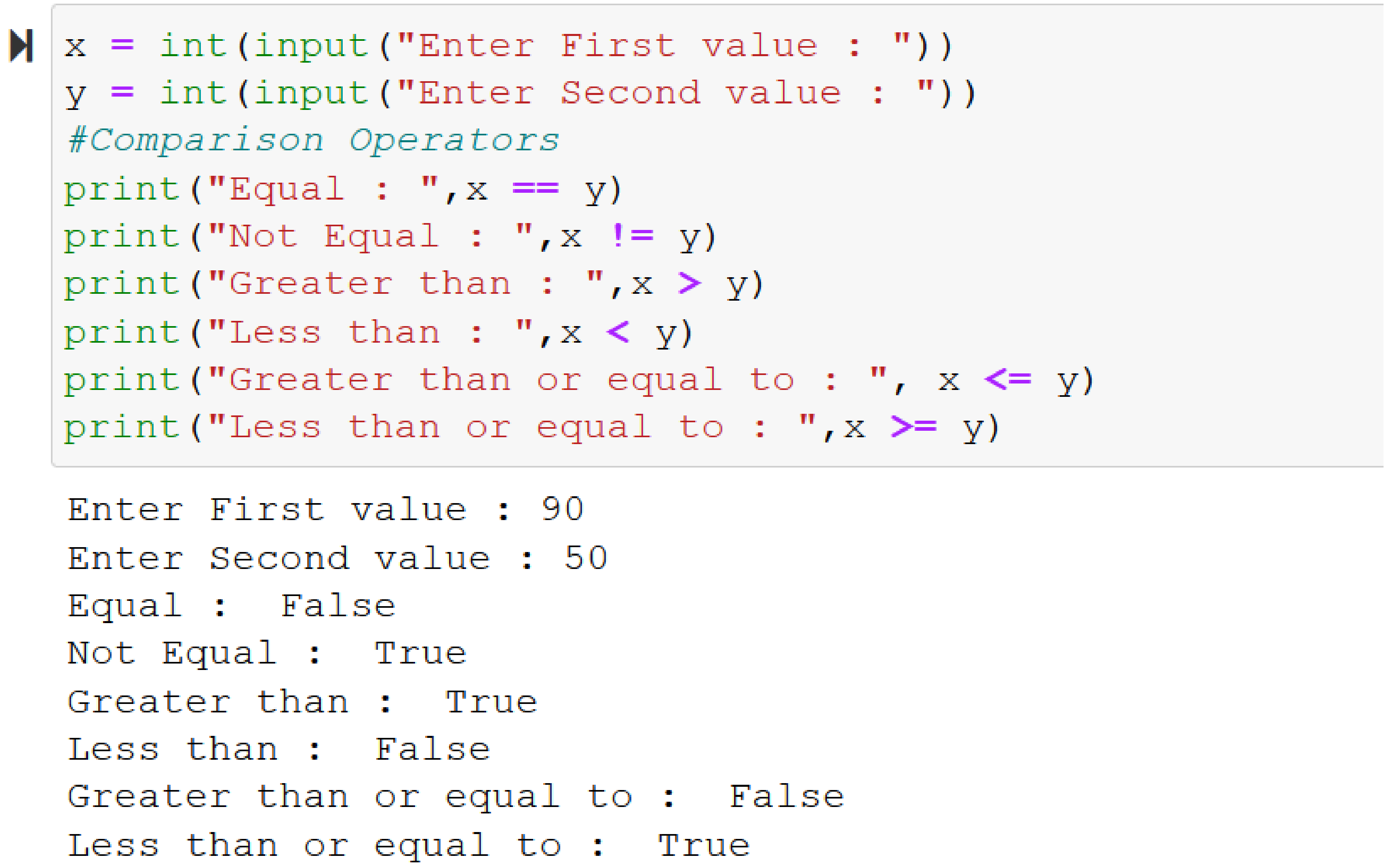
```
x = 10
x = x & 3
print(x)
x = x | 3
print(x)
x = x ^ 3
print(x)
x = x >> 3
print(x)
x = x << 3
print(x)
```

```
2
3
0
0
0
```

COMPARISON OPERATORS

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

EXAMPLE :



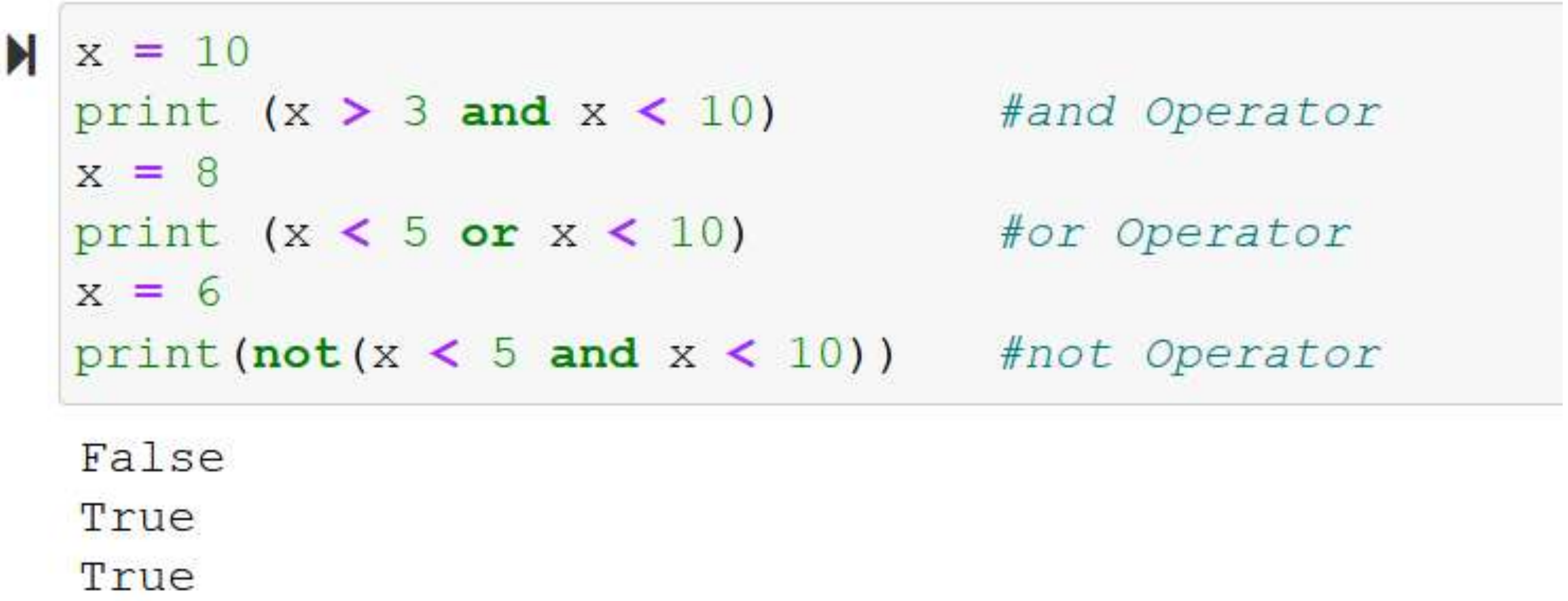
```python
x = int(input("Enter First value : "))
y = int(input("Enter Second value : "))
#Comparison Operators
print("Equal : ",x == y)
print("Not Equal : ",x != y)
print("Greater than : ",x > y)
print("Less than : ",x < y)
print("Greater than or equal to : ", x <= y)
print("Less than or equal to : ",x >= y)
```

```
Enter First value : 90
Enter Second value : 50
Equal :  False
Not Equal :  True
Greater than :  True
Less than :  False
Greater than or equal to :  False
Less than or equal to :  True
```

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 statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

EXAMPLE :



```
x = 10
print (x > 3 and x < 10)          #and Operator
x = 8
print (x < 5 or x < 10)           #or Operator
x = 6
print(not(x < 5 and x < 10))      #not Operator
```

```
False
True
True
```

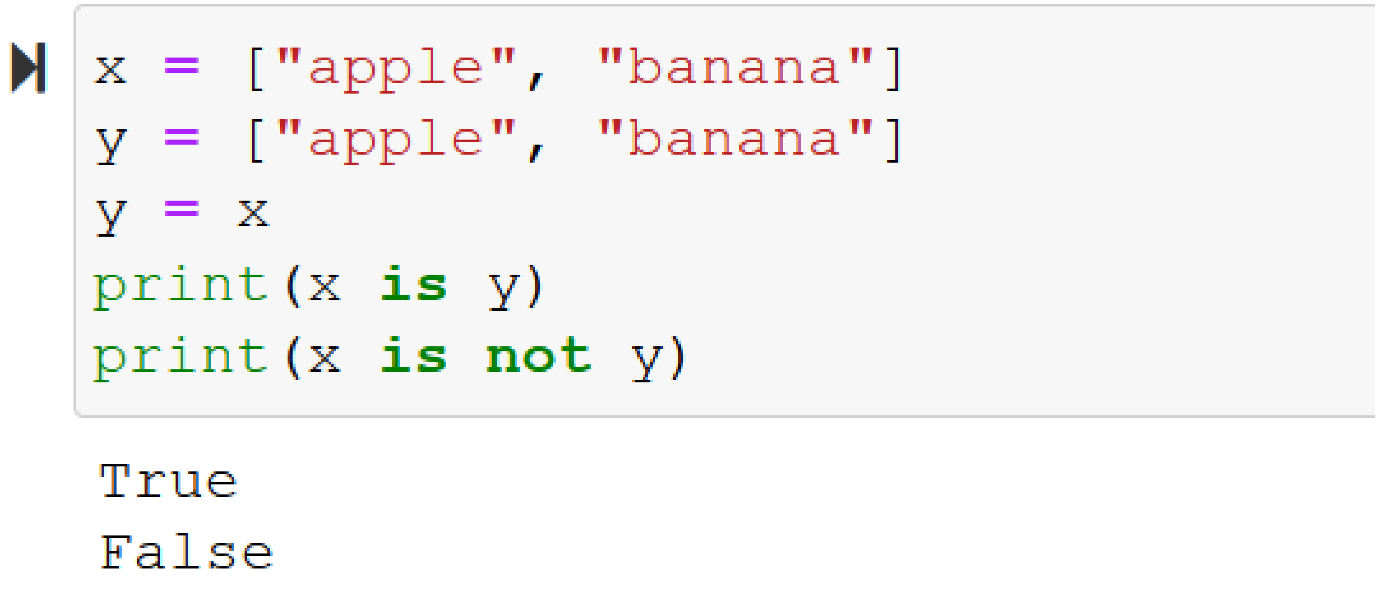
IDENTITY OPERATORS

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

MEMBERSHIP OPERATORS

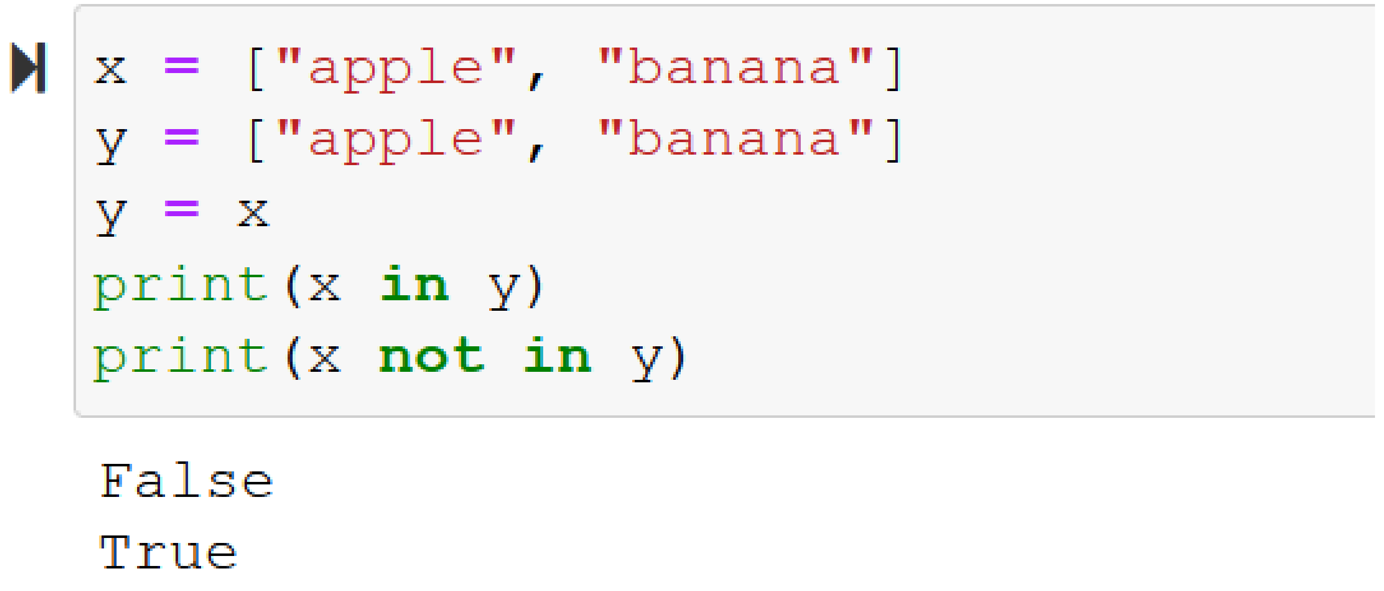
Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

EXAMPLE :



```python
x = ["apple", "banana"]
y = ["apple", "banana"]
y = x
print(x is y)
print(x is not y)
```

```
True
False
```



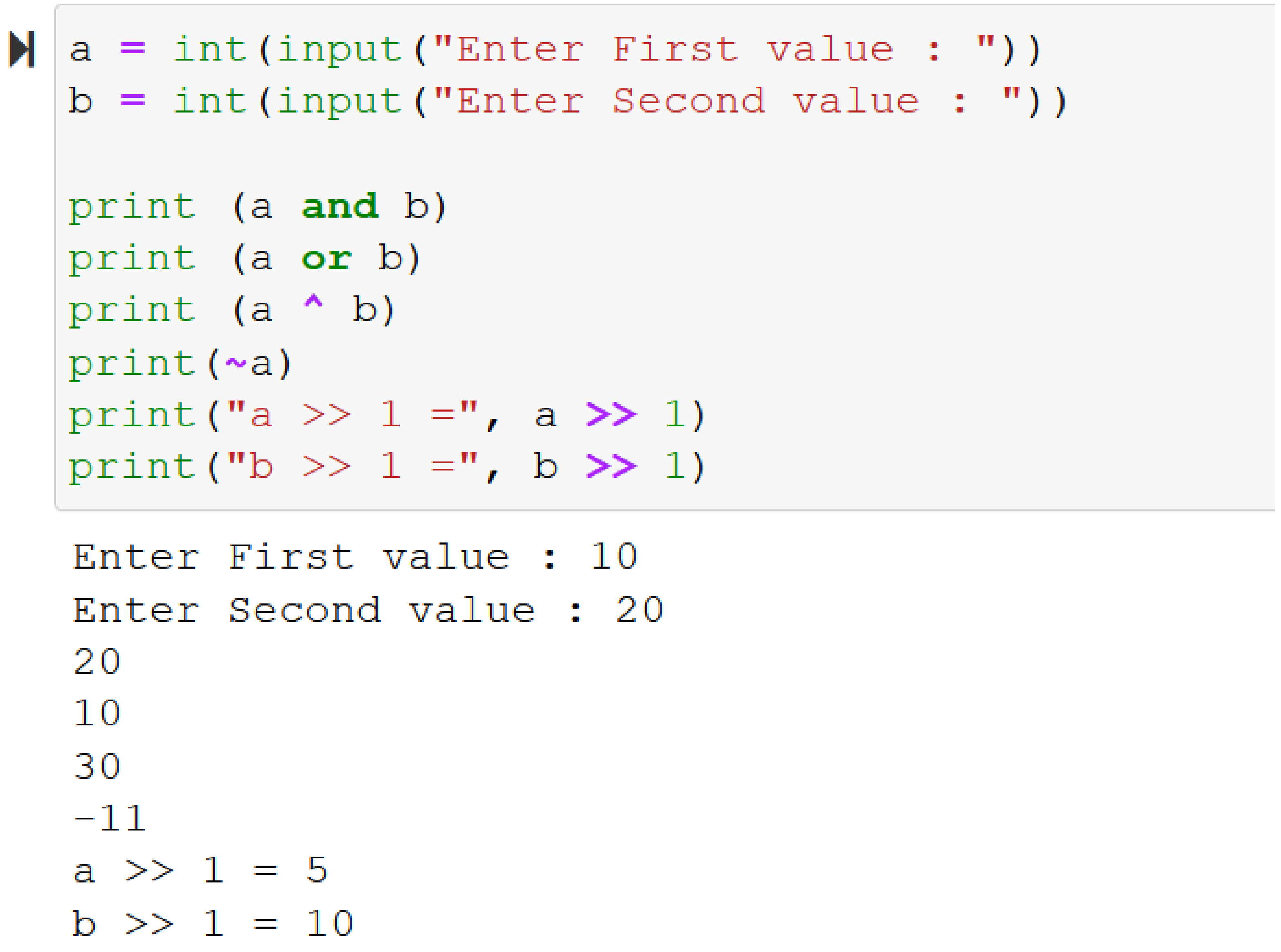
```python
x = ["apple", "banana"]
y = ["apple", "banana"]
y = x
print(x in y)
print(x not in y)
```

```
False
True
```

BITWISE OPERATORS

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

EXAMPLE :



```python
a = int(input("Enter First value : "))
b = int(input("Enter Second value : "))

print (a and b)
print (a or b)
print (a ^ b)
print(~a)
print("a >> 1 =", a >> 1)
print("b >> 1 =", b >> 1)
```

```
Enter First value : 10
Enter Second value : 20
20
10
30
-11
a >> 1 = 5
b >> 1 = 10
```

Thank You!