MATH LIBRARY & OPRTATIONS & CONDITIONS

Abdelrahman A. Mohamed

Main Headlines:

- Math library
- Comparison Operators
- Assignment Operators
- Logical Operators
- Identity operators
- Bitwise operators
- Python Operator Precedence
- Conditions(If……elif…..else. statements)

Math library

-How to import the library to your programe ?

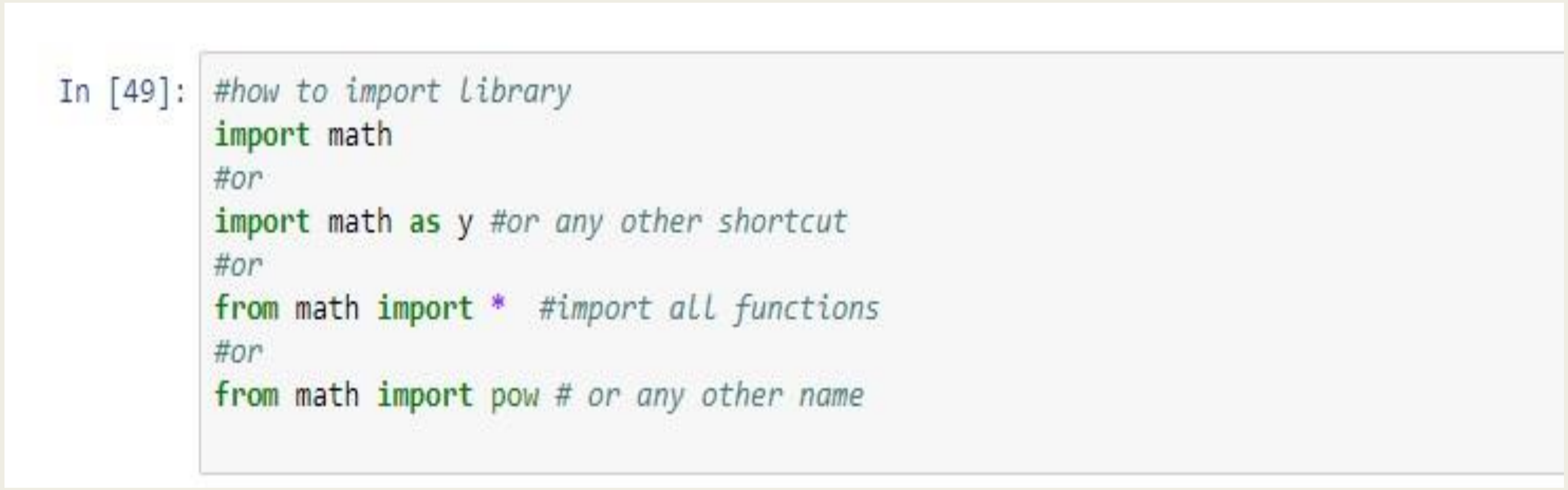
-We have four ways to do that:

1- import math

2- import math as (m or any shortcut you want)

3-from math import *   (this import all the math functions)

4-from math import (function name you want to use)

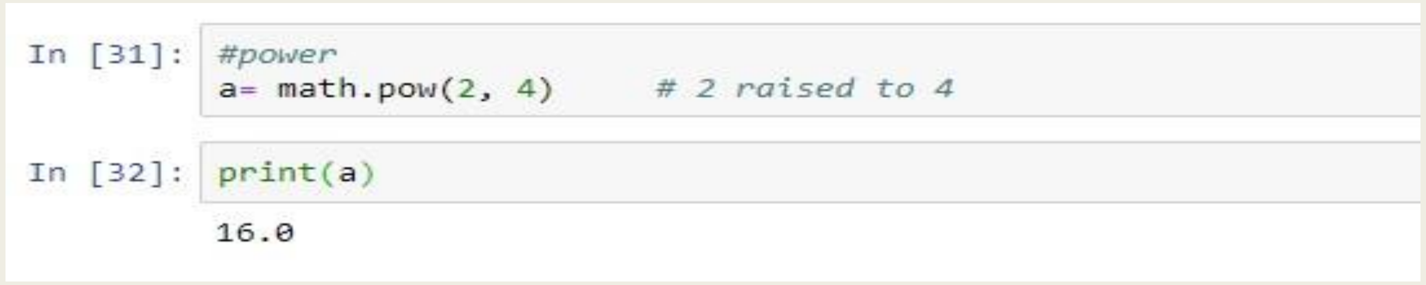


```
In [49]: #how to import library
         import math
         #or
         import math as y #or any other shortcut
         #or
         from math import *  #import all functions
         #or
         from math import pow # or any other name
```

Math library

- **math.pow(x, y) :**

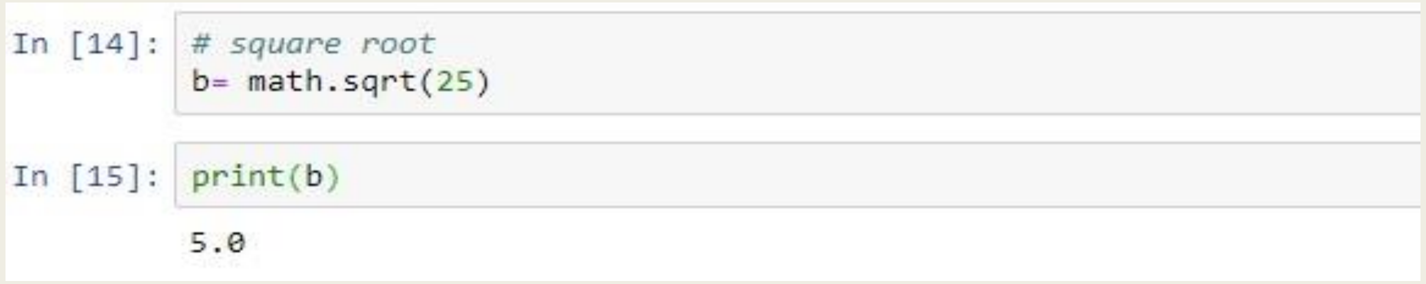
Returns x raised to the power y



```
In [31]: #power
         a= math.pow(2, 4)     # 2 raised to 4

In [32]: print(a)

         16.0
```

- **math.sqrt(x):**

Returns the square root of x

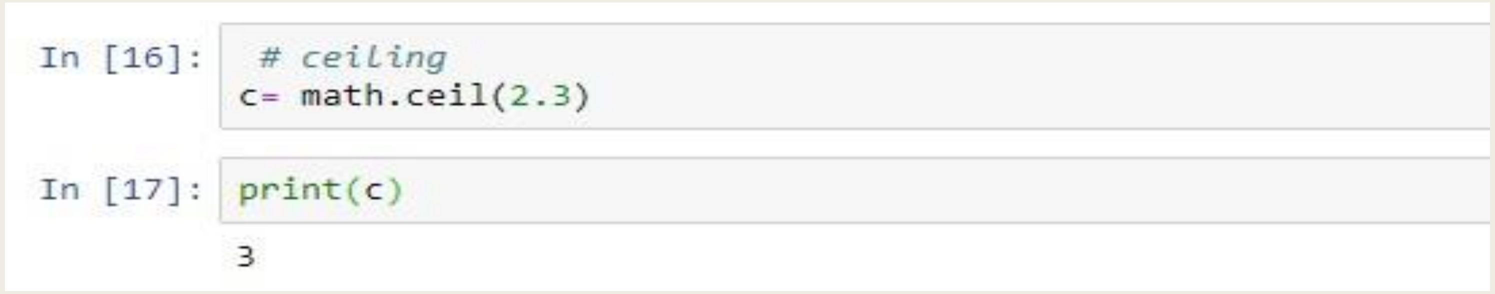


```
In [14]: # square root
         b= math.sqrt(25)

In [15]: print(b)

         5.0
```

Math library

- **math.ceil(x):**

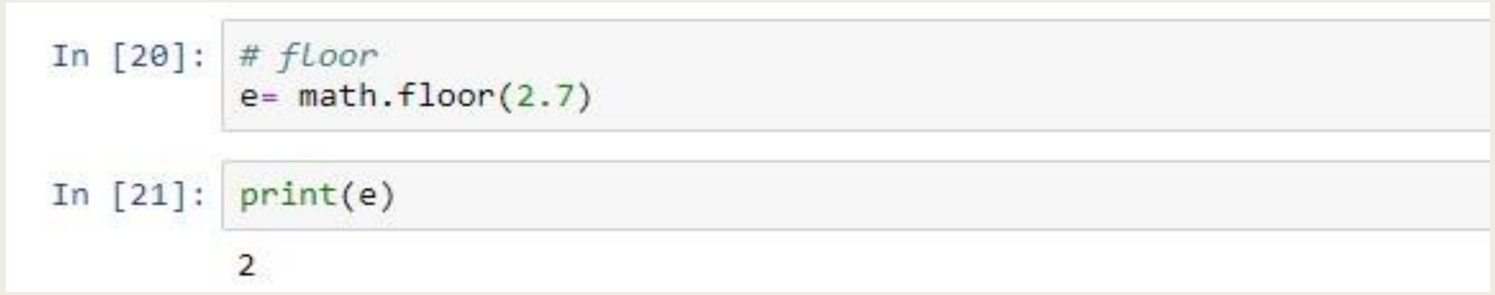
Returns the smallest integer greater than or equal to x.



```
In [16]:  # ceiling
         c= math.ceil(2.3)

In [17]: print(c)

         3
```

- **math.floor(x):**

Returns the smallest integer greater than or equal to x.

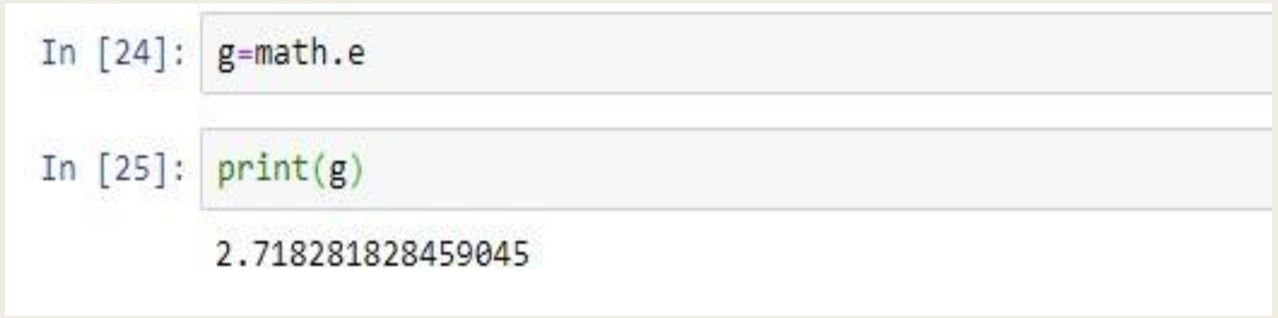


```
In [20]: # floor
         e= math.floor(2.7)

In [21]: print(e)

         2
```

Math library

- **math.e:**

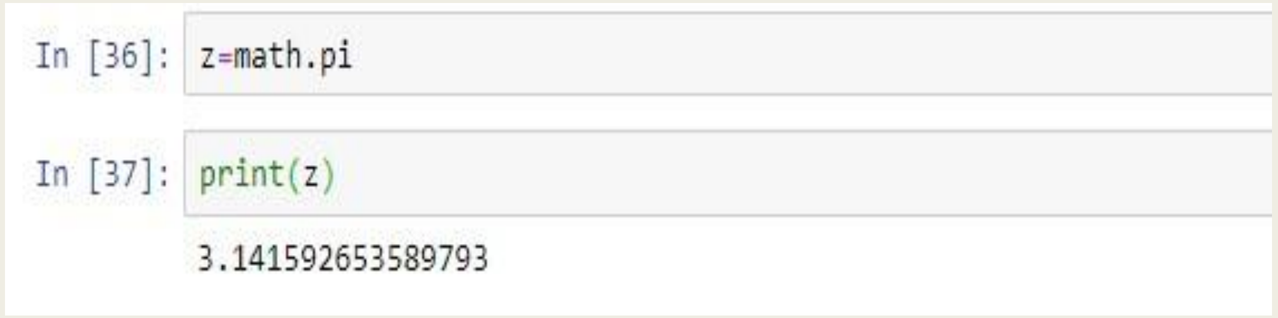
mathematical constant e (2.71828...)



```
In [24]: g=math.e

In [25]: print(g)
         2.718281828459045
```

- **math.pi:**

Mathematical constant, the ratio of circumference of a circle to it's diameter (3.14159...)

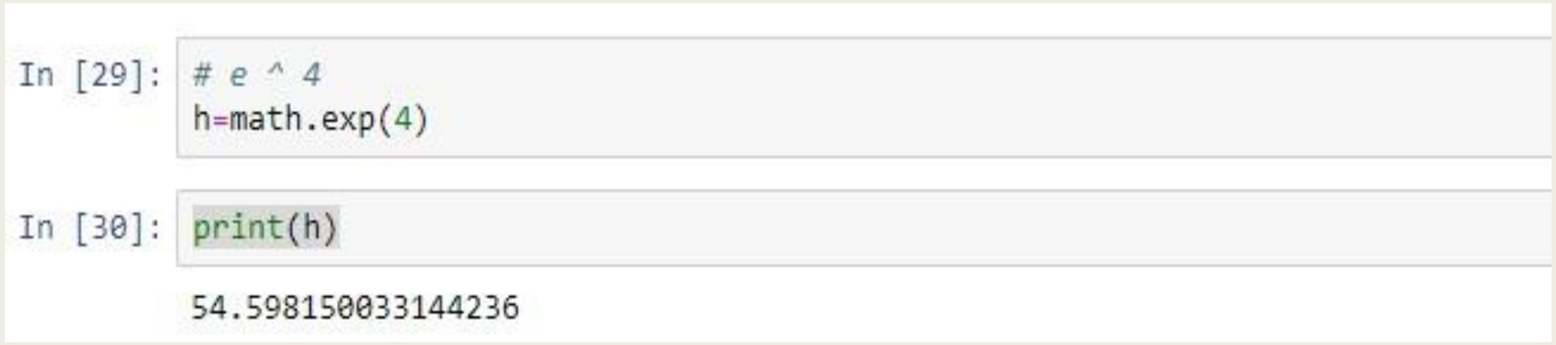


```
In [36]: z=math.pi

In [37]: print(z)
         3.141592653589793
```

Math library

- **math.exp(x):**

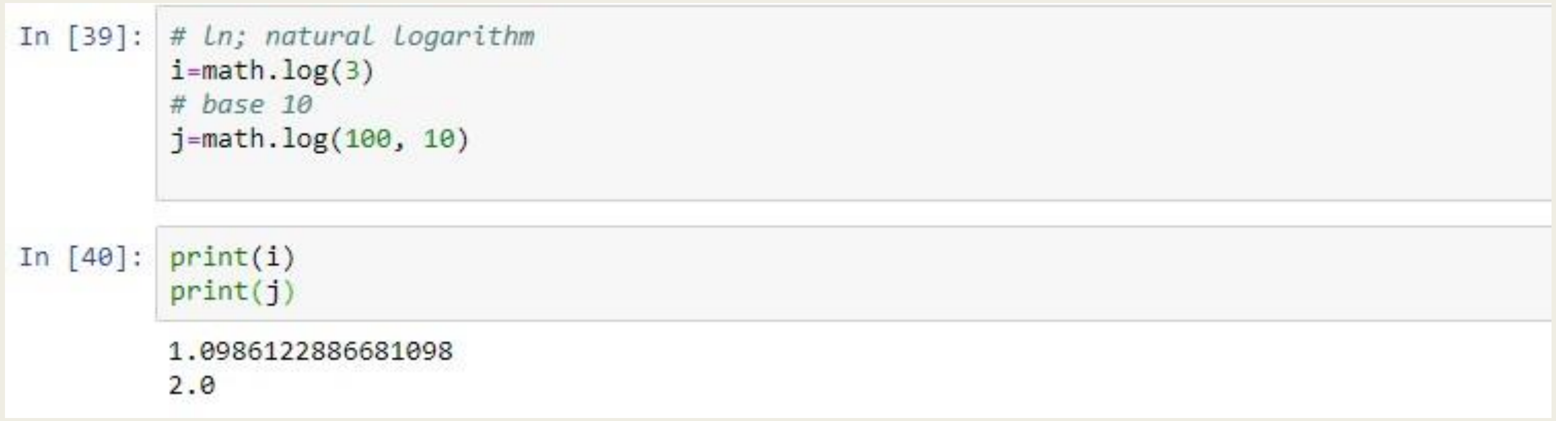
Returns e**x



```
In [29]: # e ^ 4
         h=math.exp(4)

In [30]: print(h)

         54.598150033144236
```

- **math.log(x) && math.log(x,base):**

Returns the logarithm of x to the base (defaults to e)

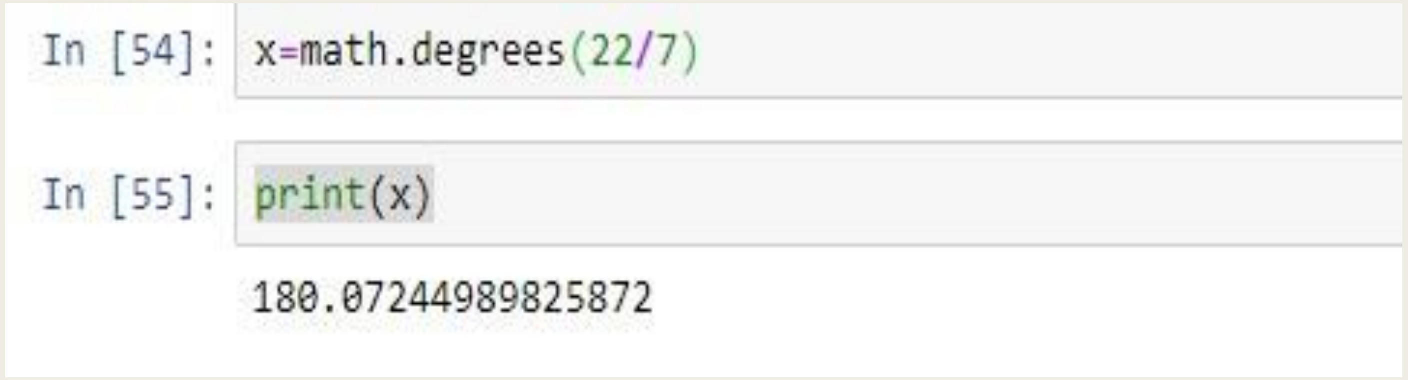


```
In [39]: # ln; natural logarithm
         i=math.log(3)
         # base 10
         j=math.log(100, 10)

In [40]: print(i)
         print(j)

         1.0986122886681098
         2.0
```

Math library

- **math.degrees(x):**

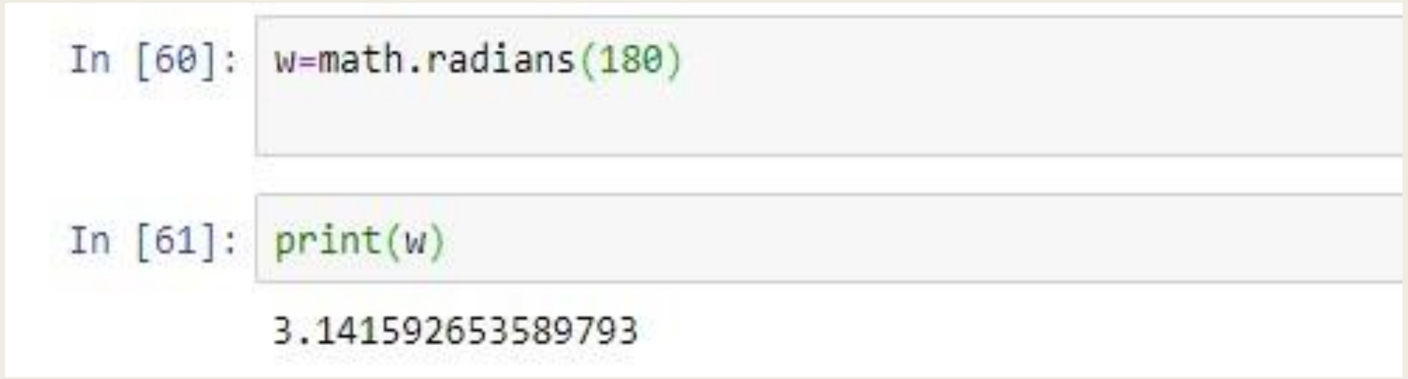
Converts angle x from radians to degrees



```
In [54]: x=math.degrees(22/7)

In [55]: print(x)

         180.07244989825872
```

- **math.radians(x):**

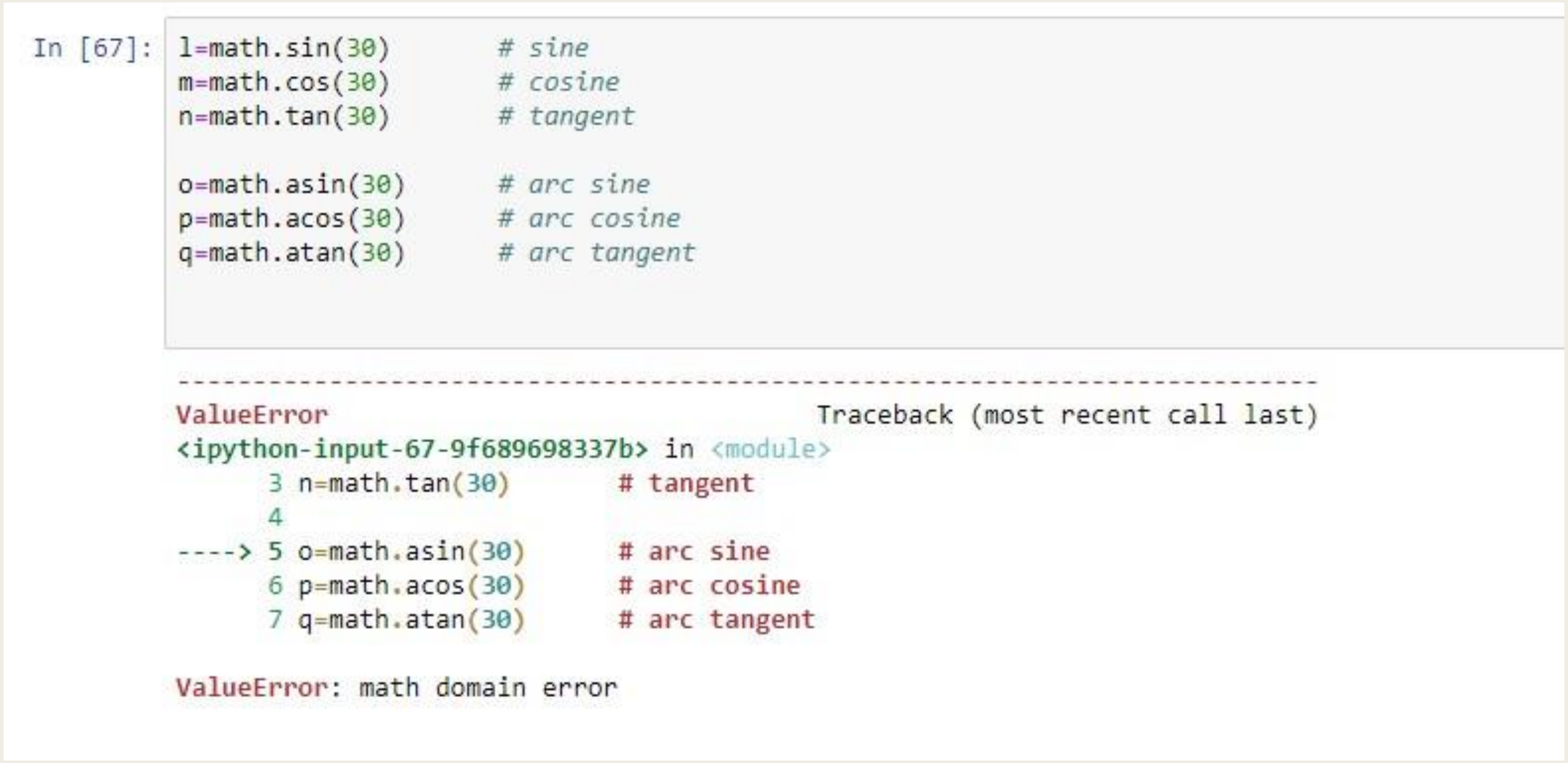
Converts angle x from degrees to radians



```
In [60]: w=math.radians(180)

In [61]: print(w)

         3.141592653589793
```

Math library

- Trigonometric Functions

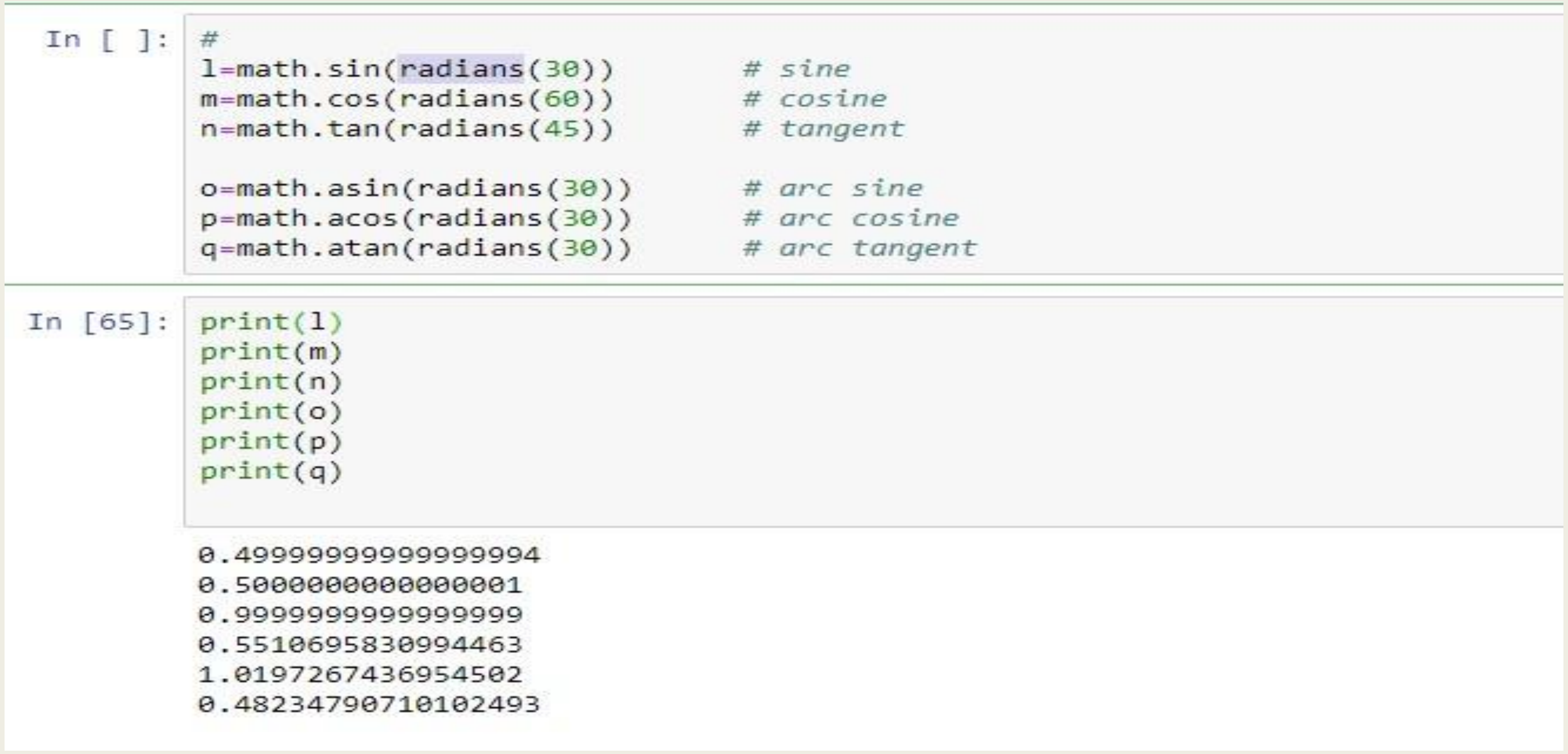


```
In [67]: l=math.sin(30)       # sine
         m=math.cos(30)       # cosine
         n=math.tan(30)       # tangent

         o=math.asin(30)      # arc sine
         p=math.acos(30)      # arc cosine
         q=math.atan(30)      # arc tangent
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-67-9f689698337b> in <module>
      3 n=math.tan(30)       # tangent
      4
----> 5 o=math.asin(30)      # arc sine
      6 p=math.acos(30)      # arc cosine
      7 q=math.atan(30)      # arc tangent

ValueError: math domain error
```

Math library

-As we see in the previous slide we got an error because the Trigonometric functions only take radians angle

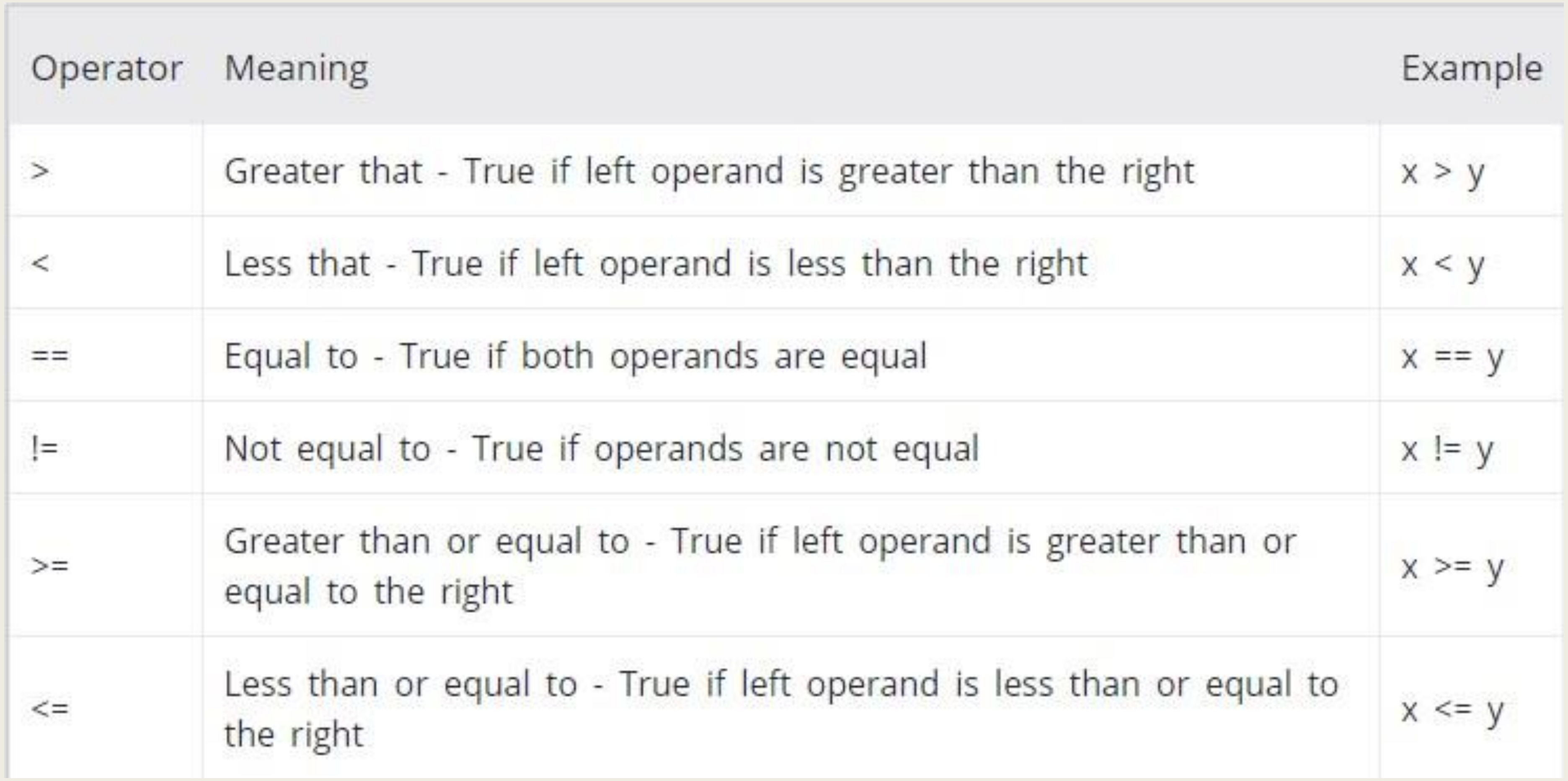


```
In [ ]:  #
         l=math.sin(radians(30))        # sine
         m=math.cos(radians(60))        # cosine
         n=math.tan(radians(45))        # tangent

         o=math.asin(radians(30))       # arc sine
         p=math.acos(radians(30))       # arc cosine
         q=math.atan(radians(30))       # arc tangent
```

```
In [65]: print(l)
         print(m)
         print(n)
         print(o)
         print(p)
         print(q)
```

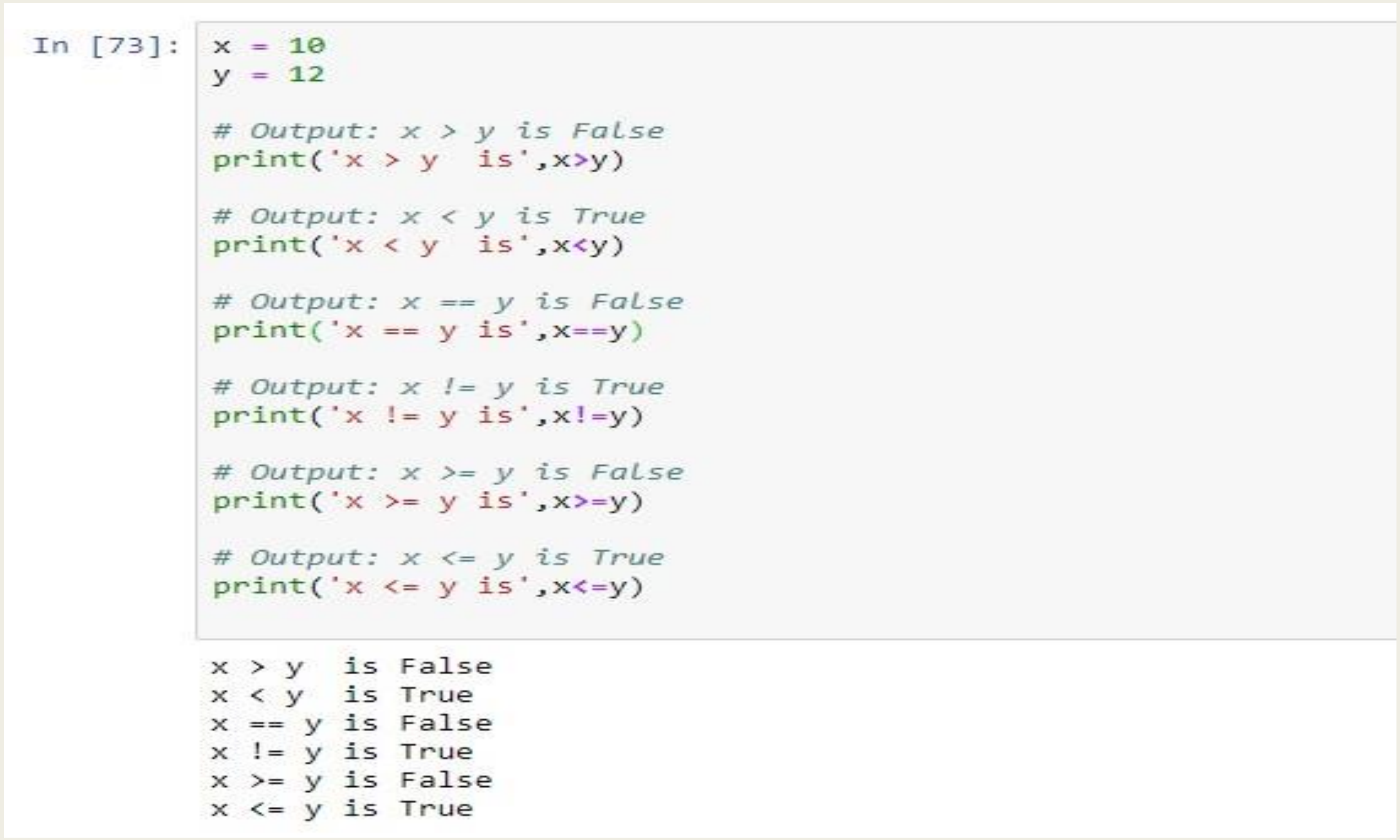
```
0.49999999999999994
0.5000000000000001
0.9999999999999999
0.5510695830994463
1.0197267436954502
0.48234790710102493
```

Comparison Operators



Operator	Meaning	Example
>	Greater that - True if left operand is greater than the right	x > y
<	Less that - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

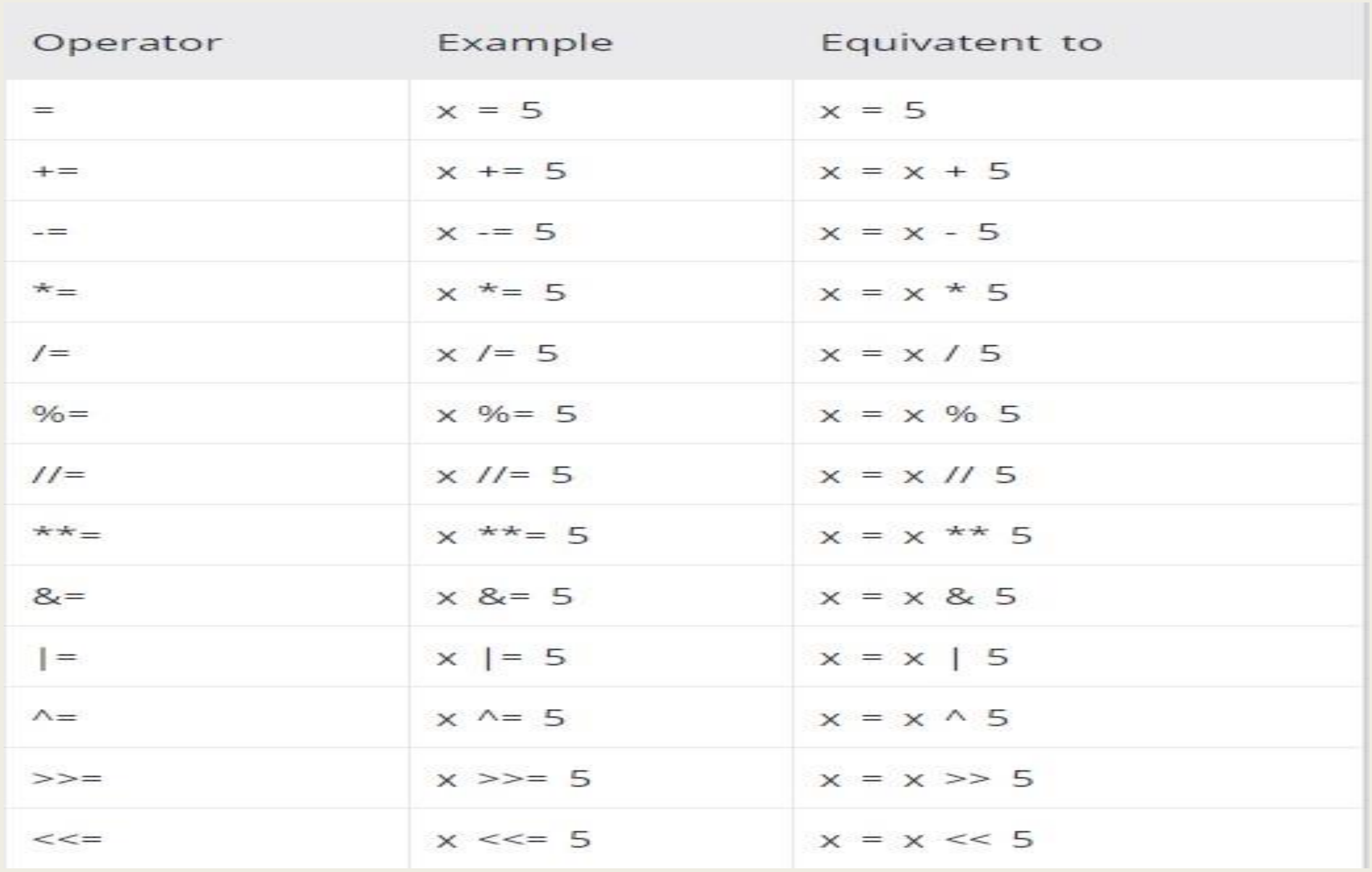
Comparison Operators (Example)



```
In [73]: x = 10
         y = 12

         # Output: x > y is False
         print('x > y  is',x>y)

         # Output: x < y is True
         print('x < y  is',x<y)

         # Output: x == y is False
         print('x == y is',x==y)

         # Output: x != y is True
         print('x != y is',x!=y)

         # Output: x >= y is False
         print('x >= y is',x>=y)

         # Output: x <= y is True
         print('x <= y is',x<=y)
```

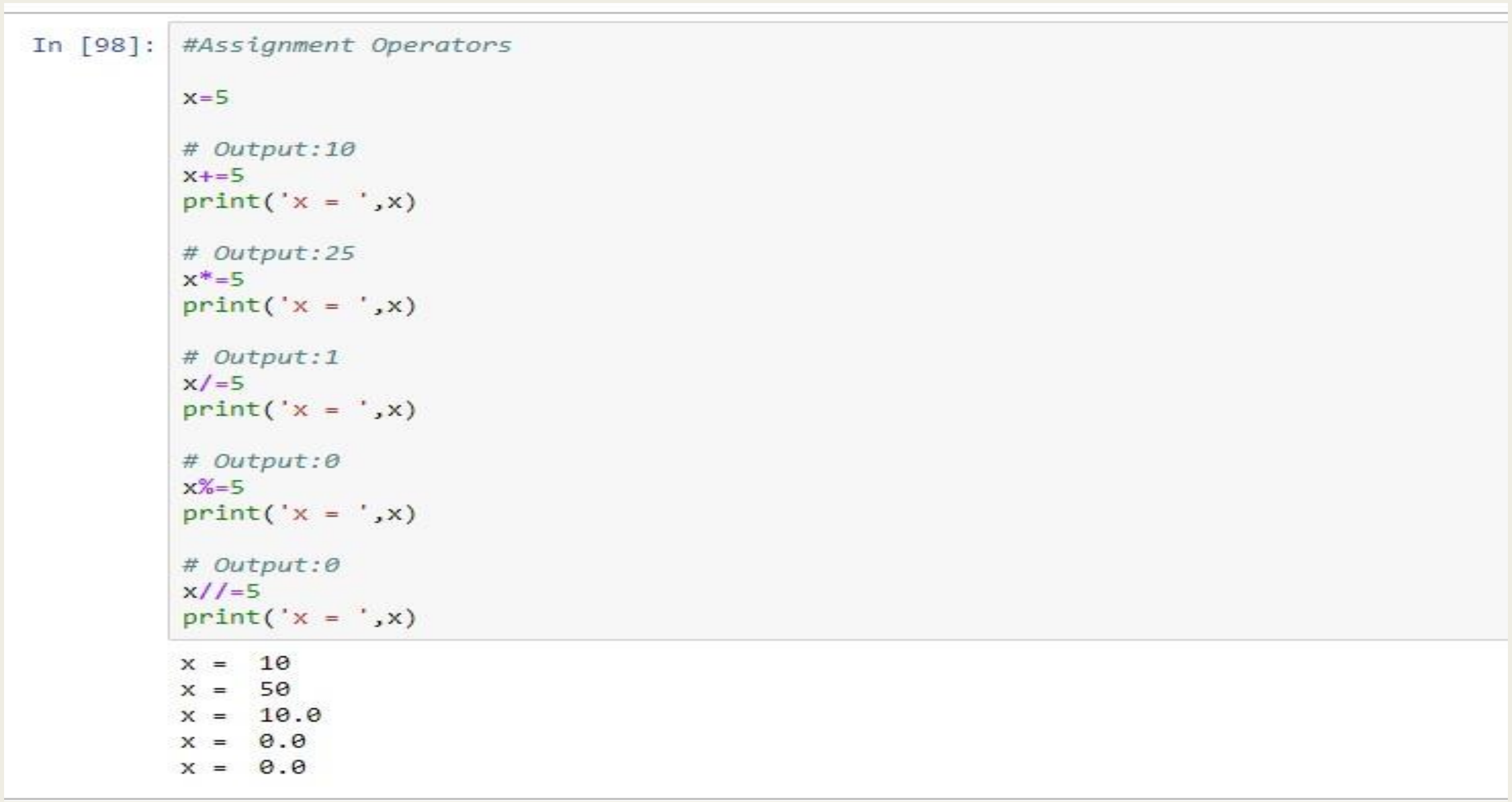
```
x > y  is False
x < y  is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

Assignment Operators



Operator	Example	Equivatent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Assignment Operators (Example)

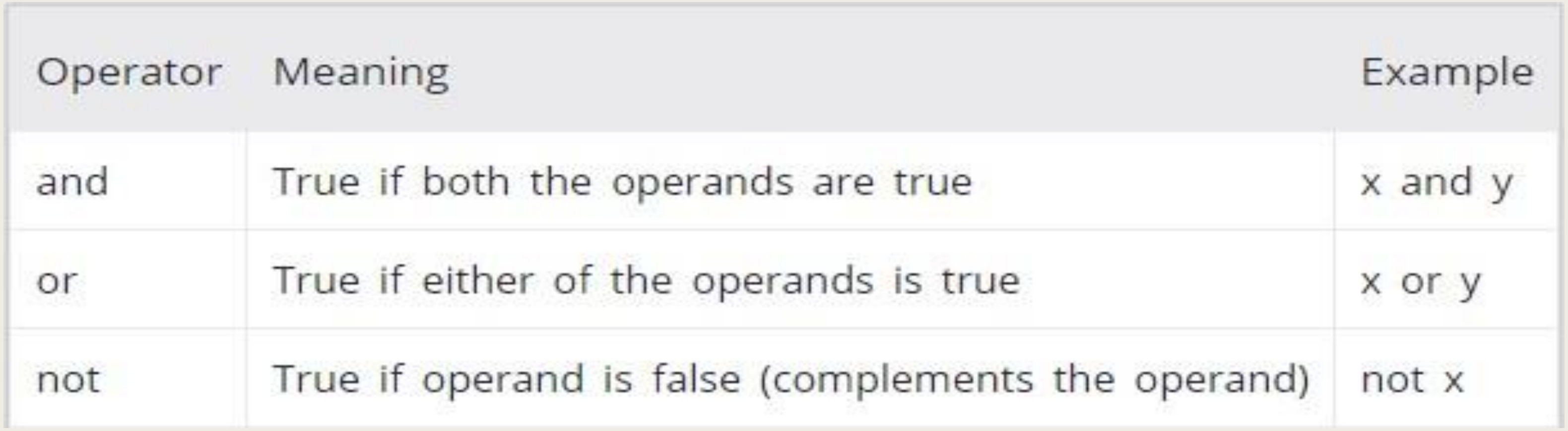


```
In [98]: #Assignment Operators

         x=5

         # Output:10
         x+=5
         print('x = ',x)

         # Output:25
         x*=5
         print('x = ',x)

         # Output:1
         x/=5
         print('x = ',x)

         # Output:0
         x%=5
         print('x = ',x)

         # Output:0
         x//=5
         print('x = ',x)
```

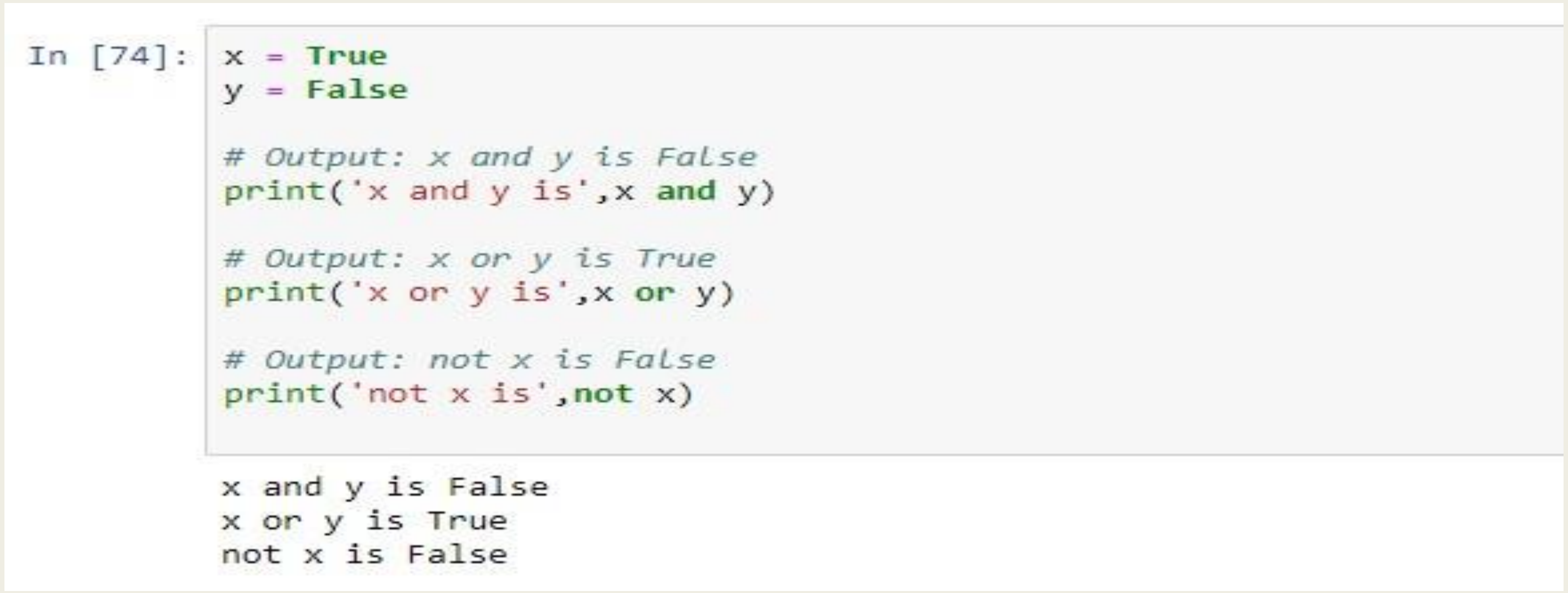
```
x =  10
x =  50
x =  10.0
x =  0.0
x =  0.0
```

Logical operators

Logical operators are the “and” , “or” ,”not”



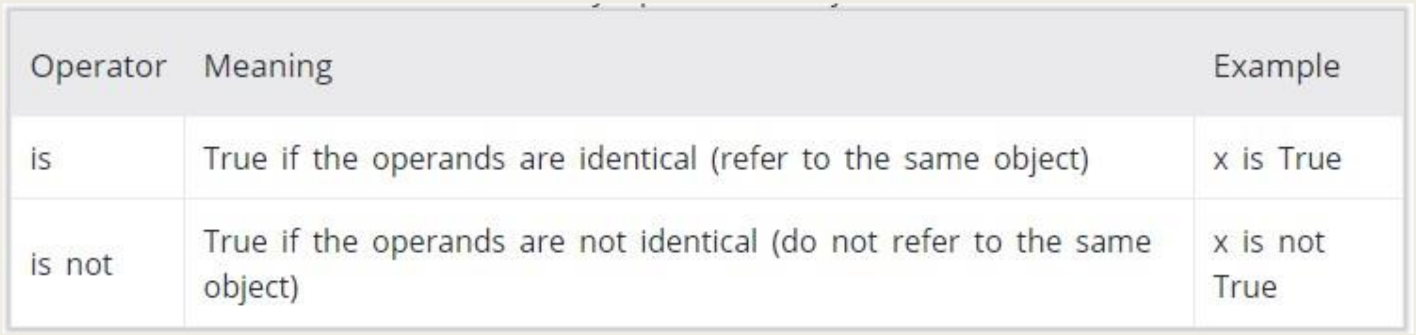
Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x



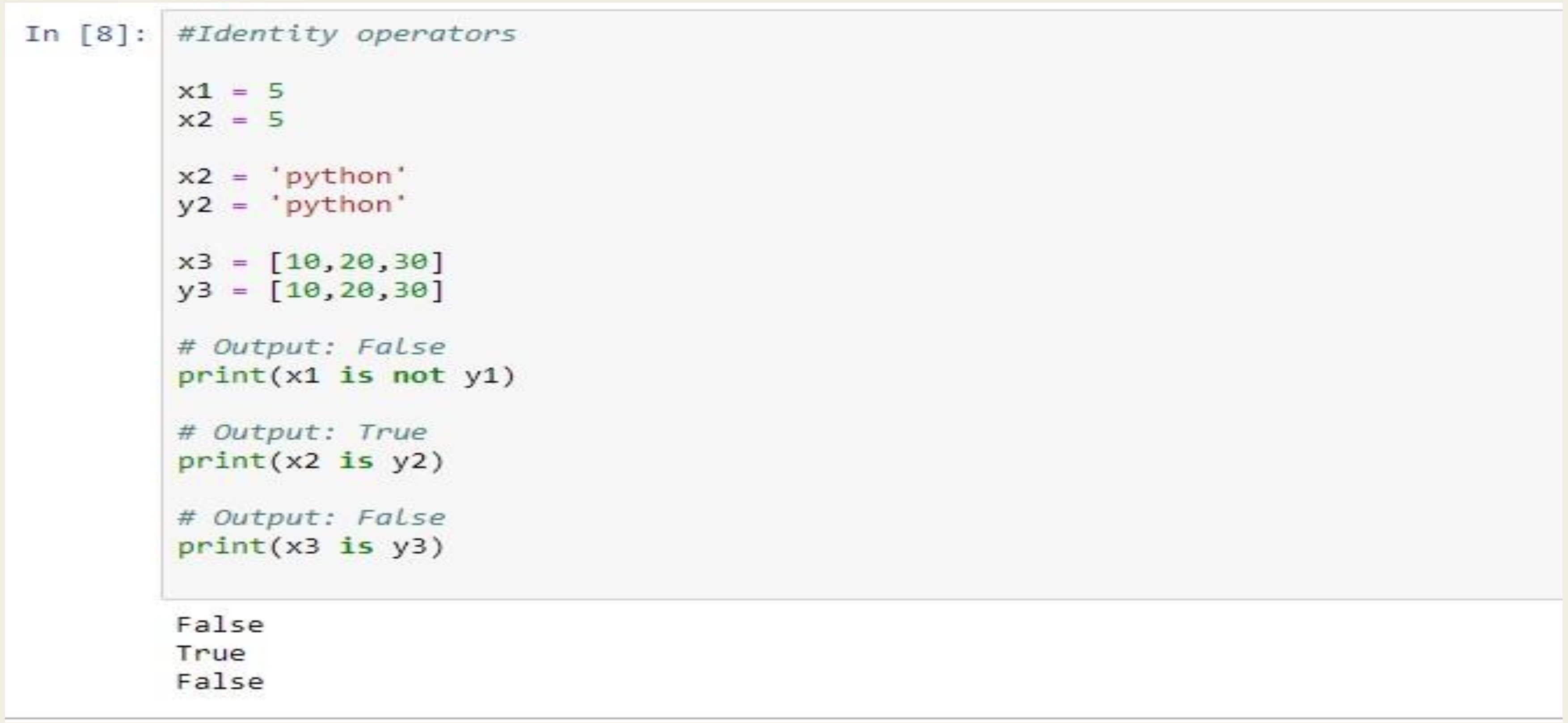
```
In [74]: x = True
         y = False

         # Output: x and y is False
         print('x and y is',x and y)

         # Output: x or y is True
         print('x or y is',x or y)

         # Output: not x is False
         print('not x is',not x)

         x and y is False
         x or y is True
         not x is False
```

Abdelrahman A. Mohamed

Identity operators



Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True



```
In [8]: #Identity operators

        x1 = 5
        x2 = 5

        x2 = 'python'
        y2 = 'python'

        x3 = [10,20,30]
        y3 = [10,20,30]

        # Output: False
        print(x1 is not y1)

        # Output: True
        print(x2 is y2)

        # Output: False
        print(x3 is y3)
```

```
False
True
False
```

Membership operators

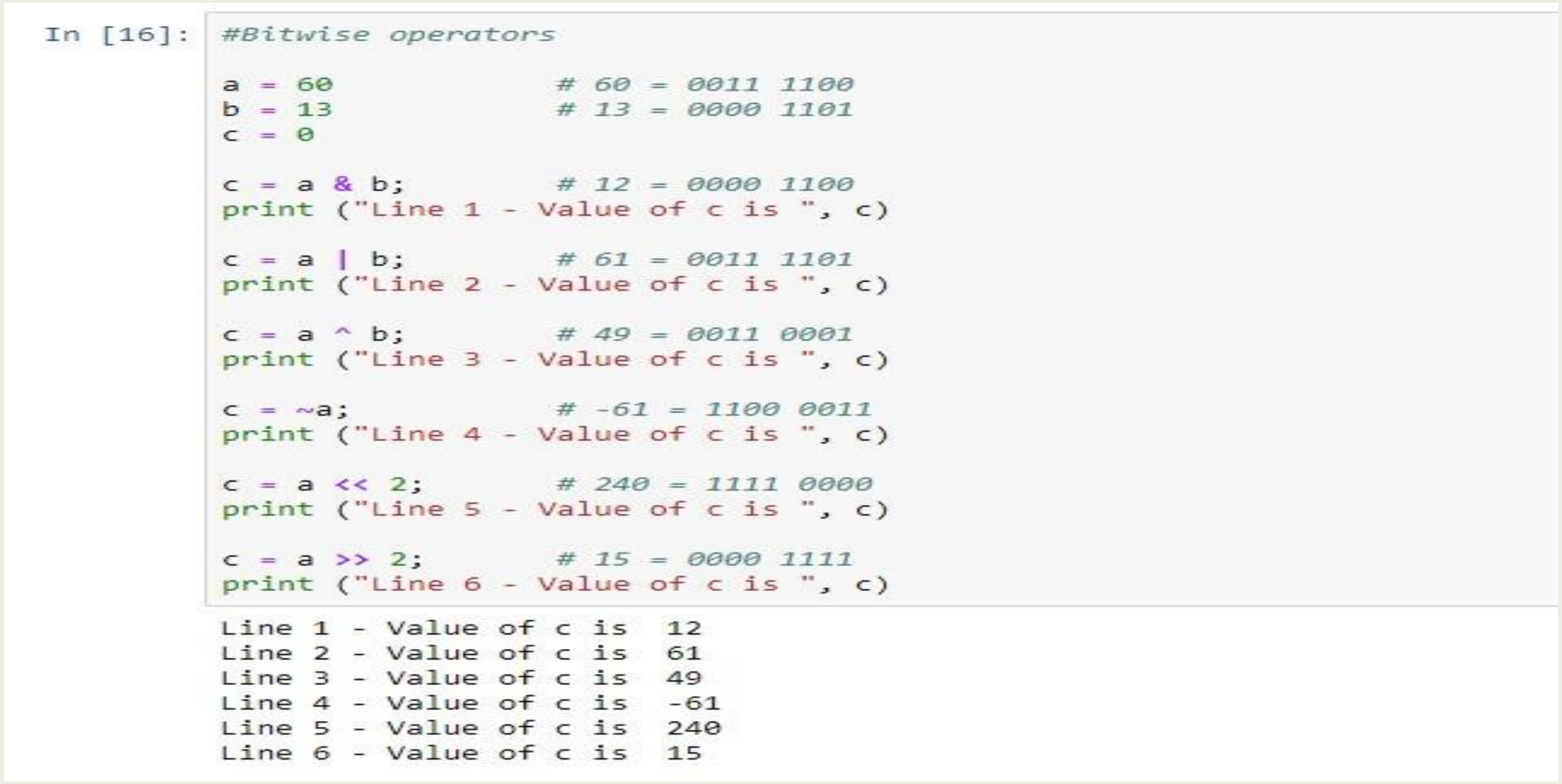
In and **not in** are the membership operators in Python. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary "we will explain them in the following lectures").

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Bitwise operators

Operator	Meaning	Example
&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

Bitwise operators (Example)



```
In [16]: #Bitwise operators

         a = 60              # 60 = 0011 1100
         b = 13              # 13 = 0000 1101
         c = 0

         c = a & b;          # 12 = 0000 1100
         print ("Line 1 - Value of c is ", c)

         c = a | b;          # 61 = 0011 1101
         print ("Line 2 - Value of c is ", c)

         c = a ^ b;          # 49 = 0011 0001
         print ("Line 3 - Value of c is ", c)

         c = ~a;             # -61 = 1100 0011
         print ("Line 4 - Value of c is ", c)

         c = a << 2;         # 240 = 1111 0000
         print ("Line 5 - Value of c is ", c)

         c = a >> 2;         # 15 = 0000 1111
         print ("Line 6 - Value of c is ", c)

         Line 1 - Value of c is  12
         Line 2 - Value of c is  61
         Line 3 - Value of c is  49
         Line 4 - Value of c is  -61
         Line 5 - Value of c is  240
         Line 6 - Value of c is  15
```

Python Operator Precedence

()	Parentheses (grouping)
**	Exponentiation
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

“IF” Statement

If the condition(test expression) is True will implement the body of “if” then continue the program otherwise continue the program without implementing the body of “if”

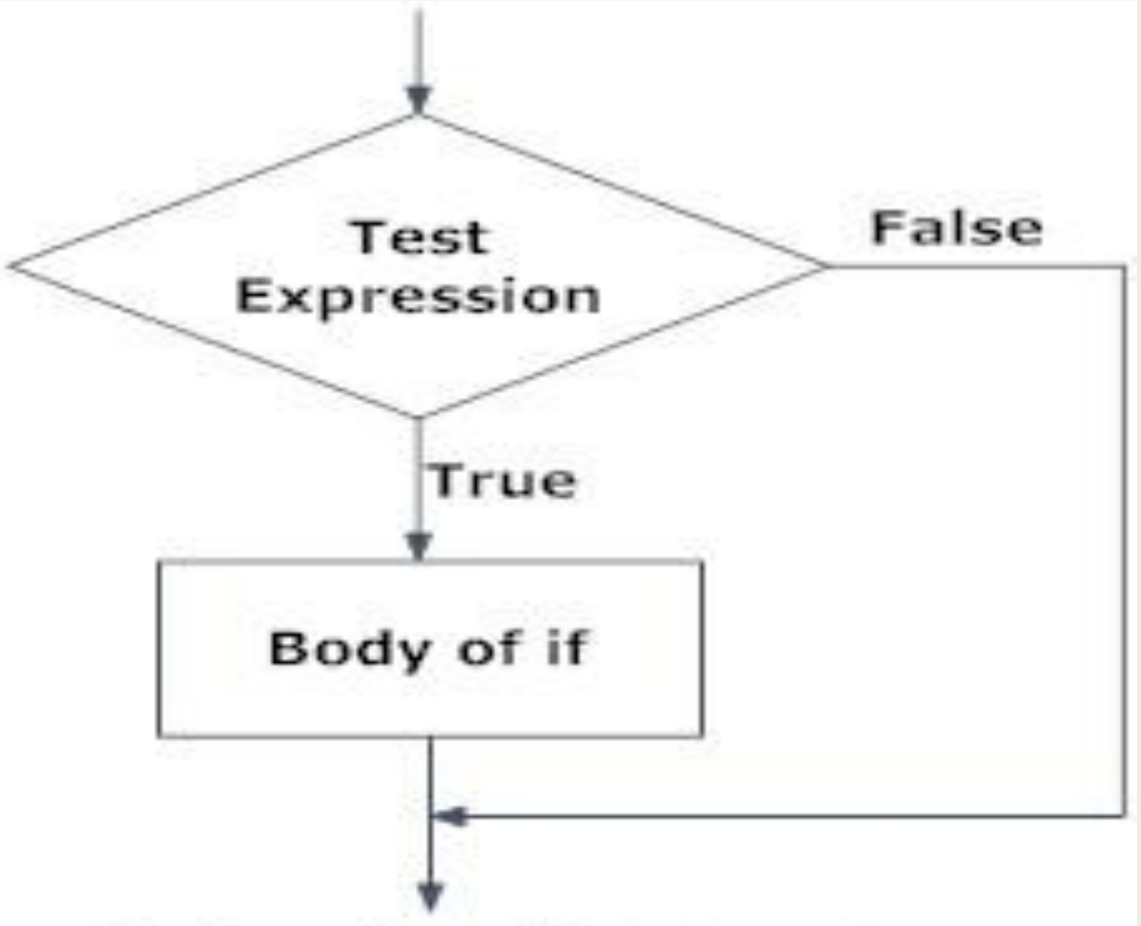


Fig: Operation of if statement

“IF”…..”Else” Statements

- If the condition (test expression) was True, will implement the body of “if” ,then continue the program , false will implement body of “else” then continue the program.

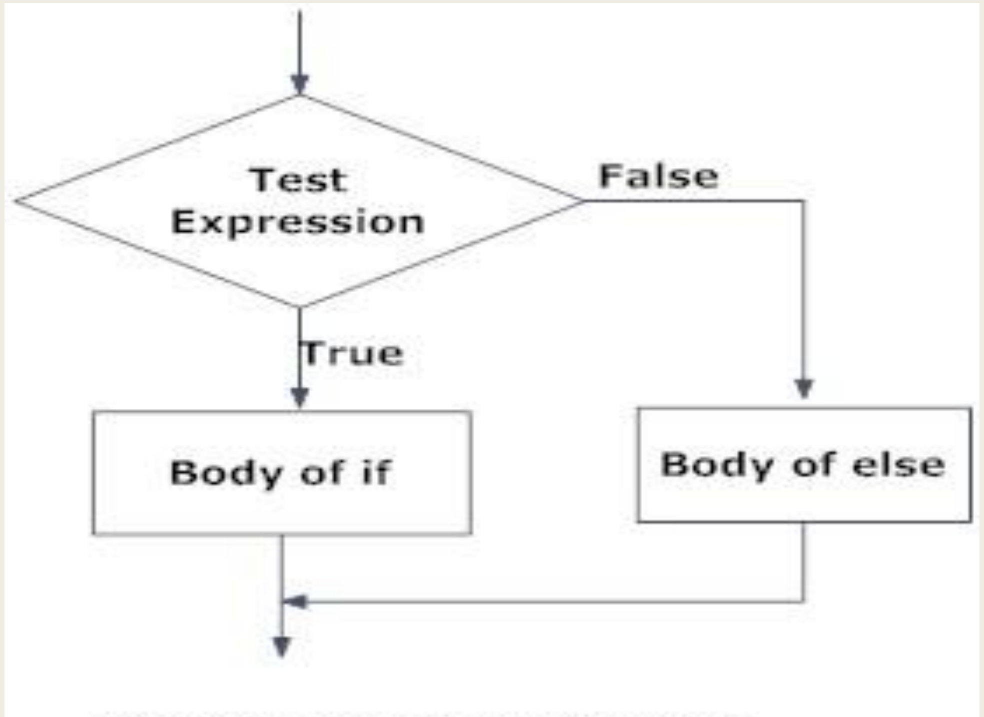


Fig: Operation of if...else statement

Abdelrahman A. Mohamed

“IF”.....”Elif......””Else” Statements

- The “elif” is short for “else if”. It allows us to check for multiple expressions.
- If the condition for “if “ is it checks the condition of the next “elif” lock and so on. If all the conditions are false body of “else” is executed.
- Only one block among the several “IF”.....”Elif......””Else” Statements blocks is executed according to the condition.
- The “if” block can have only one “else” block. But it can have multiple ”elif” blocks.

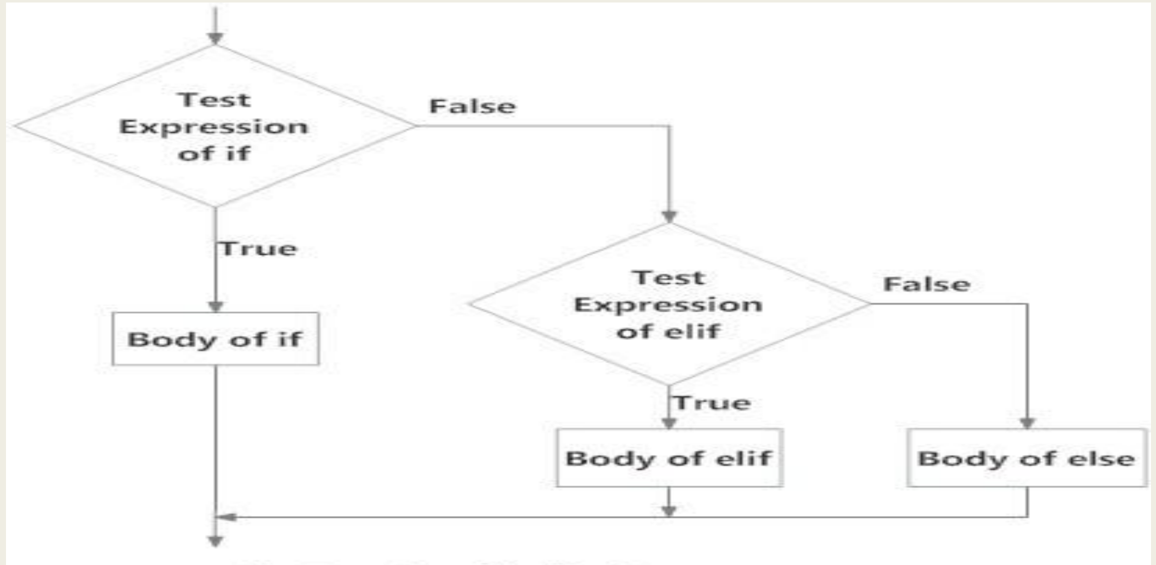


Fig: Operation of if...elif...else statement

Abdelrahman A. Mohamed