# Session 3

* len() function
* Built-in list methods
* Mutable vs Immutable data types
* Operators in Python
  + Arithmetic Operator
  + Assignment Operator
  + Logical Operator
  + Bitwise Operator
  + Operator Precedence

## 1) len() function

len() function is a built-in Python method that returns to us the length/count of the number of elements in a collection datatype(lists, strings, dictionary, etc.) The len() when called on a nested multi-dimensional list only the number of elements in the first dimension is returned.

my\_list = [1,2,3]  
print(my\_list)  
print(len(my\_list))

[1, 2, 3]  
3

my\_list = ['A string',23,100.232,'o']  
print(my\_list)  
print(len(my\_list))

['A string', 23, 100.232, 'o']  
4

my\_list = [[1,2,3],[4,5,6],[7,8,9]]  
print(my\_list)  
print(len(my\_list))

[[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
3

**NOTE**: While using the len function even the spaces in strings are counted as an element

x = "hello world"  
print(len(x))

## 2) Built-in List methods

**All datatypes including list are an object in Python. This means we can call methods on the objects created using these classes.** Try creating a list then write the list name followed by a . and press the tab key

my\_list = [11,22,33,44,55,66,77,88,99]  
my\_list.

This gives us a dropdown of all the available built-in list methods we will be looking at a few important ones. **The methods can be :**

* In place (affects the actual list)
* not in place (the actual list is not affected)

depending on the type we might have to reassign the variable to see the changes.



The append() method adds an element to the end of the list The append() method is an inplace method.

my\_list = [11,22,33,44,55,66,77,88,99]  
my\_list.append(101)

my\_list

[11, 22, 33, 44, 55, 66, 77, 88, 99, 101]

The insert() method adds an element at the specified index The append() method is an inplace method.

my\_list = [11,22,33,44,55,66,77,88,99]  
my\_list.insert(4,"I am added before index 4")  
print(my\_list)

[11, 22, 33, 44, 'I am added before index 4', 55, 66, 77, 88, 99]

The pop()method removes and returns the last object from the list

l = [1, 2, 3, 4]  
print("Popped element:", l.pop())  
print("List after pop():", l)

Popped element: 4  
List after pop(): [1, 2, 3]

11

## 3) Mutable and Immutable data types

All the collection datatypes in Python can be classified into 2 types.

* mutable (Individual elements can be reassigned)
* immutable (Individual elements cannot be reassigned)
* **Lists are mutable**
* **Strings are immutable**

my\_list = [11,22,33,44,55,66]  
print(my\_list)  
my\_list[3] = 99  
print(my\_list)

[11, 22, 33, 44, 55, 66]  
[11, 22, 33, 99, 55, 66]

my\_string = "hello"  
print(my\_string)  
my\_string[3] = 'r'  
print(my\_string)

hello

---------------------------------------------------------------------------  
TypeError Traceback (most recent call last)  
<ipython-input-3-7587cc2651c9> in <module>  
 1 my\_string = "hello"  
 2 print(my\_string)  
----> 3 my\_string[3] = 'r'  
 4 print(my\_string)  
  
TypeError: 'str' object does not support item assignment

**We Get the above error because of the Immutability of the string Datatype**

## TASK 1

my\_list = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54,   
 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98]

* **Try accessing the last element of the above list without counting the number of elements** Hint: use len() function

## TASK 2

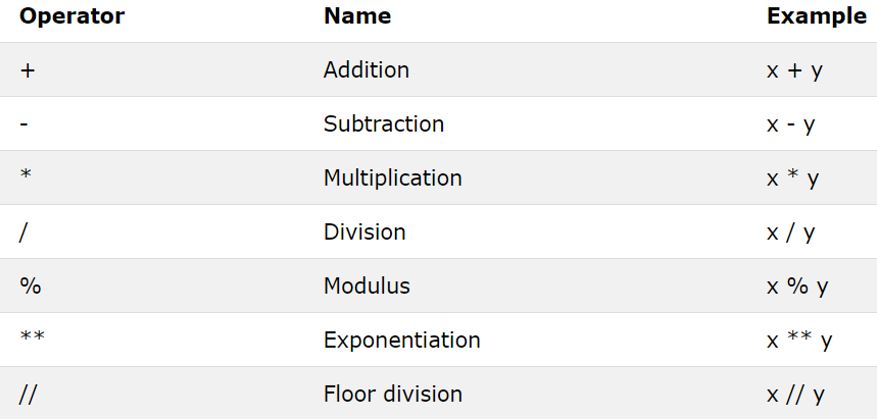
**Try using a negative index inside the slicing syntax**

## 4) Operators in Python

1. Arithmetic Operator
2. Assignment Operator
3. Logical Operator
4. Bitwise Operator
5. Operator Precedence



### Arithmetic Operators



Addition, Multiplication, subtraction, and division are the standard operations.

The Modulus (%) operator gives us the remainder of a division operation

a = 10  
b = 2  
  
print(a%b)

0

a = 10  
b = 3  
  
print(a%b)

1

The Exponential (\*\*) operator gives us the x raised to y output of 2 numbers (x^y)

a = 8  
b = 2  
print(a\*\*b)

64

We can also calculate square roots using the exponential operator

16\*\*(1/2)

4.0

4\*\*2

16

The floor division (//) operator is used to return the quotient of a division operator without the decimal places

a = 8  
b = 2  
print(a//b)

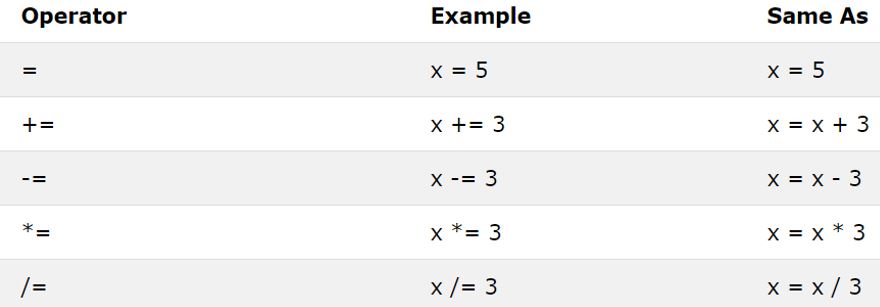
4

a = 8  
b = 3  
print(a//b)

2



### Assignment Operators



The = sign is used to assign a data type to a variable

a = 10  
print(a)

10

a += 10  
print(a)

20

The above line of code is similar to a = a + 10

a = a + 10  
print(a)

30

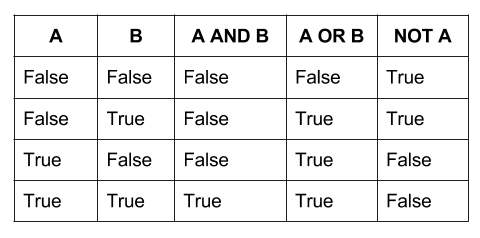
The same logic applies to all the assignment operators



### Logical Operators

* and
* or
* not

Gives the output depending on the truth table or logic table



We can try to emulate the truth table above by doing the same operations

True **and** True

True

False **and** True

False

False **or** True

True

**not** True

False

However, the logical operators are never used to directly operate on Boolean values. Instead, we generate these Boolean values from comparison operators most of the time

**not** b>=a

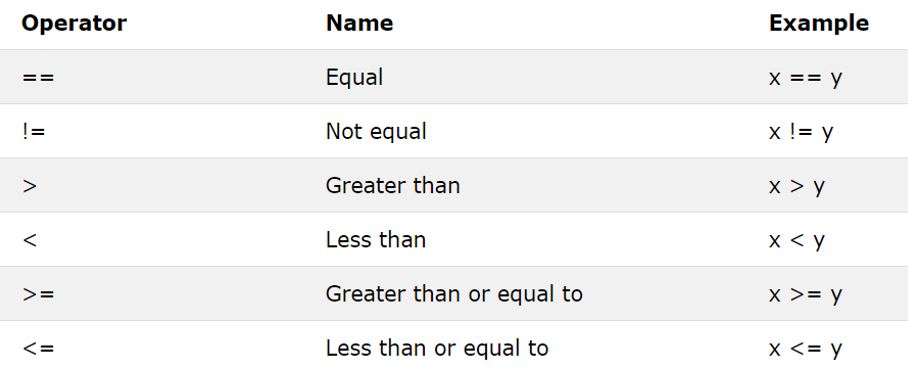
False

**not** True

False



### Comparison Operator



The comparison operator generates Boolean output depending on whether the condition being checked

a = 10  
b = 20  
c = 10  
  
print(a==b)  
print(a==c)  
print(a!=b)

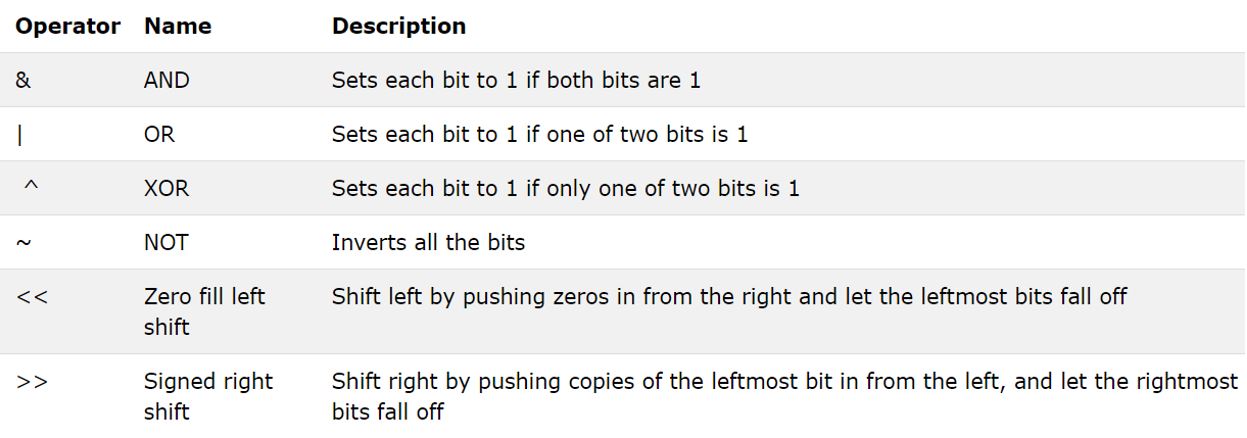
False  
True  
True

a = 10  
b = 10  
  
print(a<b)  
print(a>b)  
print(a<=b)  
print(a>=b)

False  
False  
True  
True



### Bitwise operators



Bitwise operators are also like logical operators they give output based on the truth table. But instead of acting directly on the Boolean datatype, they return results by operating on the Bit representation of the integer, float, or string datatype.

* True is considered as the bit **1**
* False is considered as the bit **0**

a = 5 *#0101 - Bitwise representation of integer 5*  
b = 10 *#1010 - Bitwise representation of integer 10*  
a&b *#0000 - Bitwise and(&) of bit representation of 5 and 10 flowing the truth table*

0

a = 5 *#0101 - Bitwise representation of integer 5*  
b = 10 *#1010 - Bitwise representation of integer 10*  
a|b *#1111 - Bitwise or(|) of bit representation of 5 and 10 flowing the truth table*

15

a = 5 *#0101 - Bitwise representation of integer 5*  
a = ~5

We might expect the output of a = ~5 to be 10

a = 5 #0101   
a = ~5 #1010 - Bitwise not(~) of bit representation of 5

Which is the bit representation of 10. But this is **not true**



When we do Bitwise not(~) in Python 2 things are happening

* The positive number gets converted to a negative number
* The 2's complement of the number is returned

a = 5 *#0101 - Bitwise representation of integer 5*  
a = ~5  
a

-6

**It is totally ok if You don't understand the Bitwise not operator. The only thing that you must remember is that doing a bitwise operation does not simply return the Decimal representation of the inverted Bitwise number.**



The left shift and right shift operator shifts the bits in the bit representation by the number of positions specified

The Right shift operation on bit representation of 5 by 1 place

a = 5 *#0101*  
a>>1 *#0010*

2

The Left shift operation on bit representation of 5 by 1 place

a = 5 *#0101*  
a<<1 *#1010*

10



**Home Work**

#### 1) create a basic calculator

* Take 2 numbers from the user (using input statement)
* Print out the remainder of 2 numbers
* Print out the quotient with decimal places.
* and the bitwise and of the numbers

#### 2) Guess the output of the below operation

#### 1. 3<10 and 4<5 or False

#### 2. not (15>10 and 17<8)

#### 3. 3<10 or 4<9 or 5>=9 and False

#### 4) Print the length of every item in the below list

my\_list = ["ON", "My", "own", "technology"]

#### 5) Remove decimal places from the given integers

x = 12.14  
y = 2000.188888

#### 6) Find if the number 24563 is divisible by 16 or not.

#### 7) Find the remainder if 2022 is divided by 4.

### 8)Create a tuple and a slice object. Start the slice object at position 3, and slice to position 5.

#### 9)Create a tuple and a slice object. Use the step parameter to return every third item

#### 10)Assume a=10 b=20 and find which variable holds the greater value using the comparison operator.