EdYoda Digital University

Python-21 March 2022

Batch-DS250322

Sagar Sarkar

Day 4-7 April Variable Assignment, Object Storage,Operators and Immutability

1. Variable Assignment
2. Object Storage
3. Immutability
4. Operators

Variable Assignment

1. When programming, it is useful to be able to store information in variables.
2. A **variable** is a string of characters and numbers associated with a piece of information. The **assignment operator**, denoted by the "=" symbol, is the operator that is used to assign values to variables in Python.
3. The line x=1 takes the known value, 1, and **assigns** that value to the variable with name "x".
4. After executing this line, this number will be stored into this variable. Until the value is changed or the variable deleted, the character x behaves like the value 1.

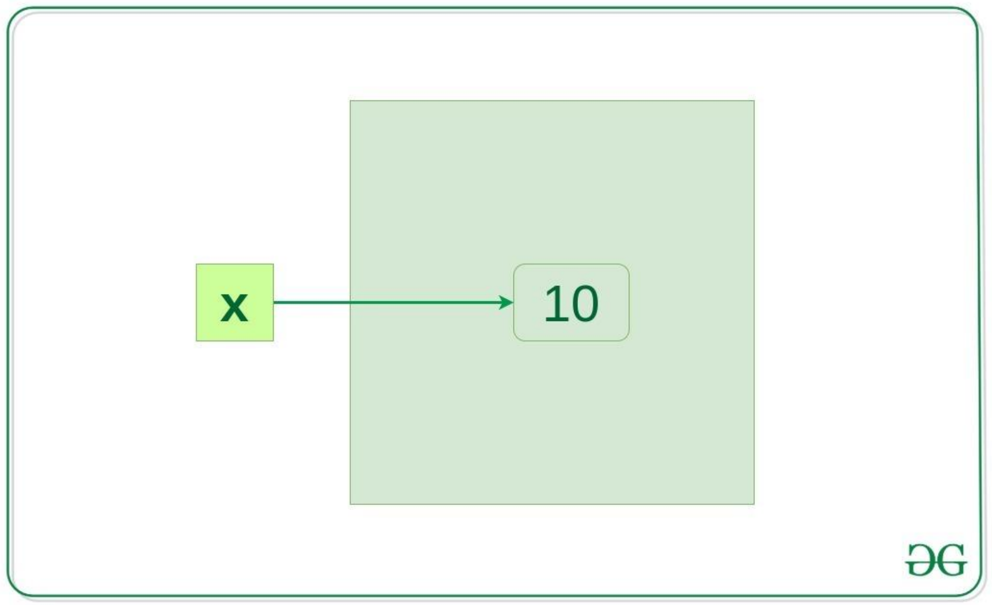
Memory Storage

- Understanding Memory allocation is important to any software developer as writing efficient code means writing a memory-efficient code.
- Memory allocation can be defined as allocating a block of space in the computer memory to a program.
- In Python memory allocation and deallocation method is automatic as the Python developers created a garbage collector for Python so that the user does not have to do manual garbage collection.

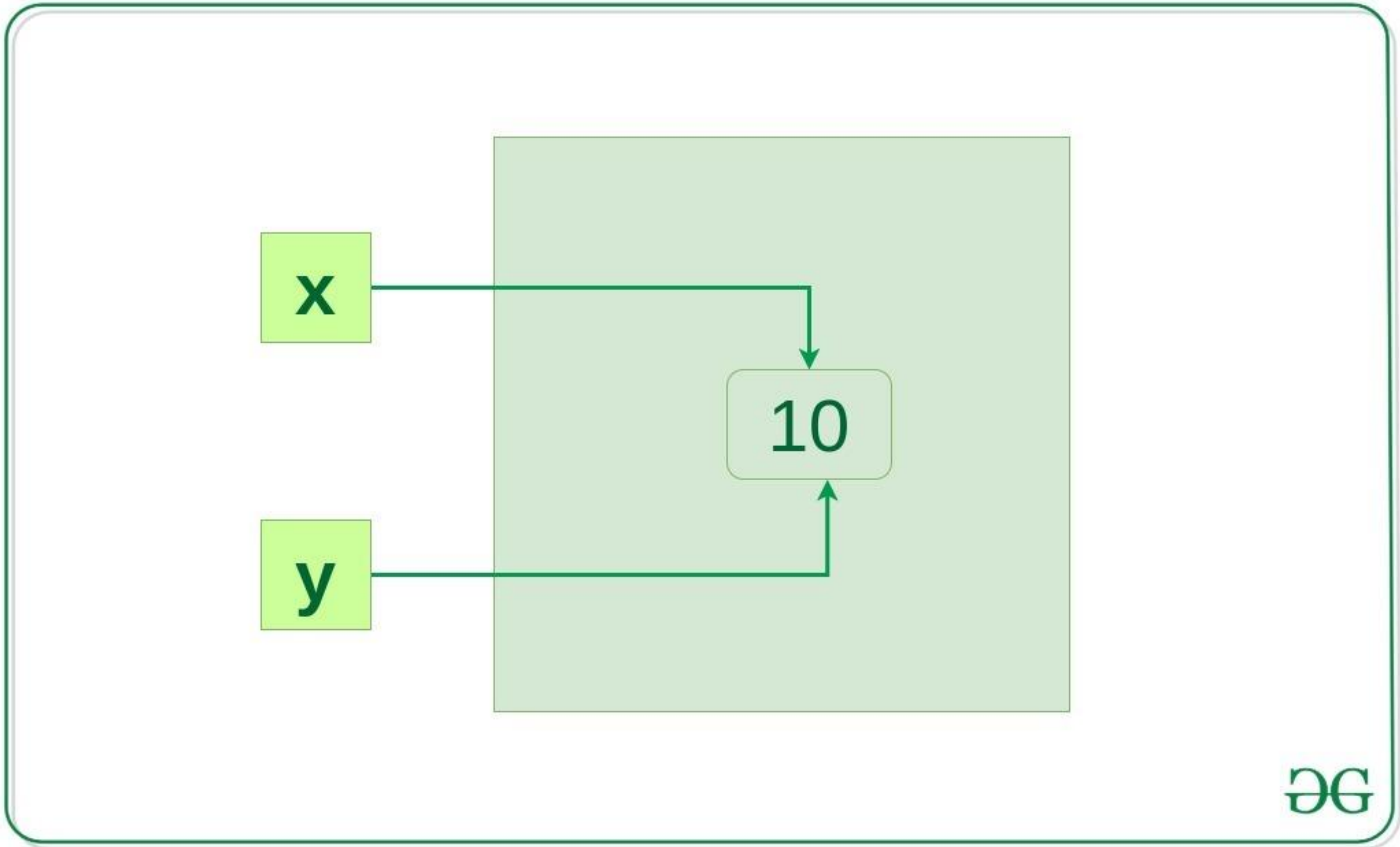
- Garbage collection is a process in which the interpreter frees up the memory when not in use to make it available for other objects.
- Assume a case where no reference is pointing to an object in memory i.e. it is not in use so, the virtual machine has a garbage collector that automatically deletes that object from the heap memory

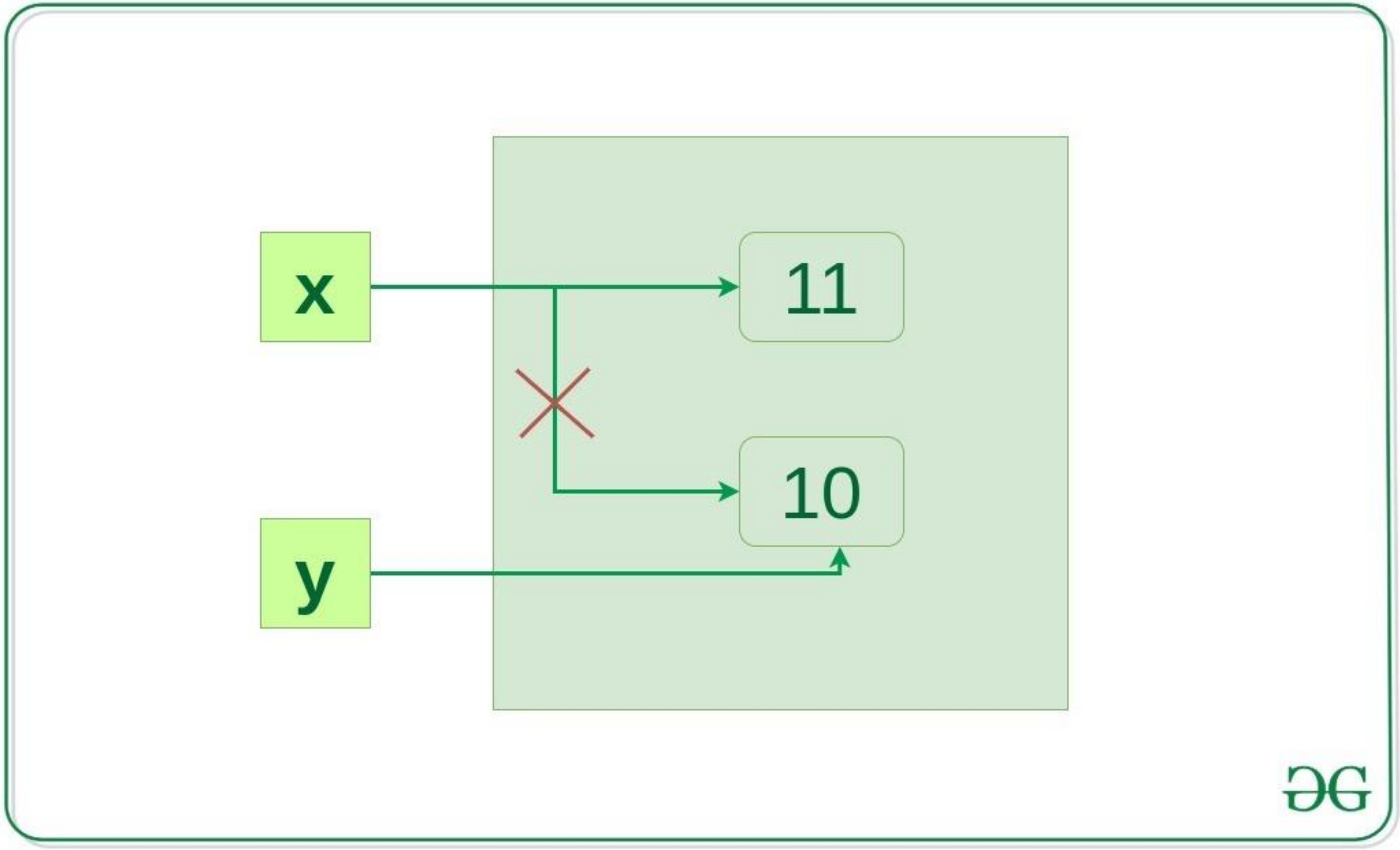
Reference Counting

- Reference counting works by counting the number of times an object is referenced by other objects in the system.
- When references to an object are removed, the reference count for an object is decremented. When the reference count becomes zero, the object is deallocated.
- For example, Let’s suppose there are two or more variables that have the same value, so, what Python virtual machine does is, rather than creating another object of the same value in the private heap, it actually makes the second variable point to that originally existing value in the private heap.
- Therefore, in the case of classes, having a number of references may occupy a large amount of space in the memory, in such a case referencing counting is highly beneficial to preserve the memory to be available for other objects



- When `x = 10` is executed an integer object 10 is created in memory and its reference is assigned to variable x, this is because everything is object in Python.
-


x
y
10


x
y
11
10

Immutability

In Python, everything is an object. An object has its own internal state. Some objects allow you to change their internal state and others don’t.

An object whose internal state can be changed is called a mutable object, while an object whose internal state cannot be changed is called an immutable object.

The following are examples of immutable objects:

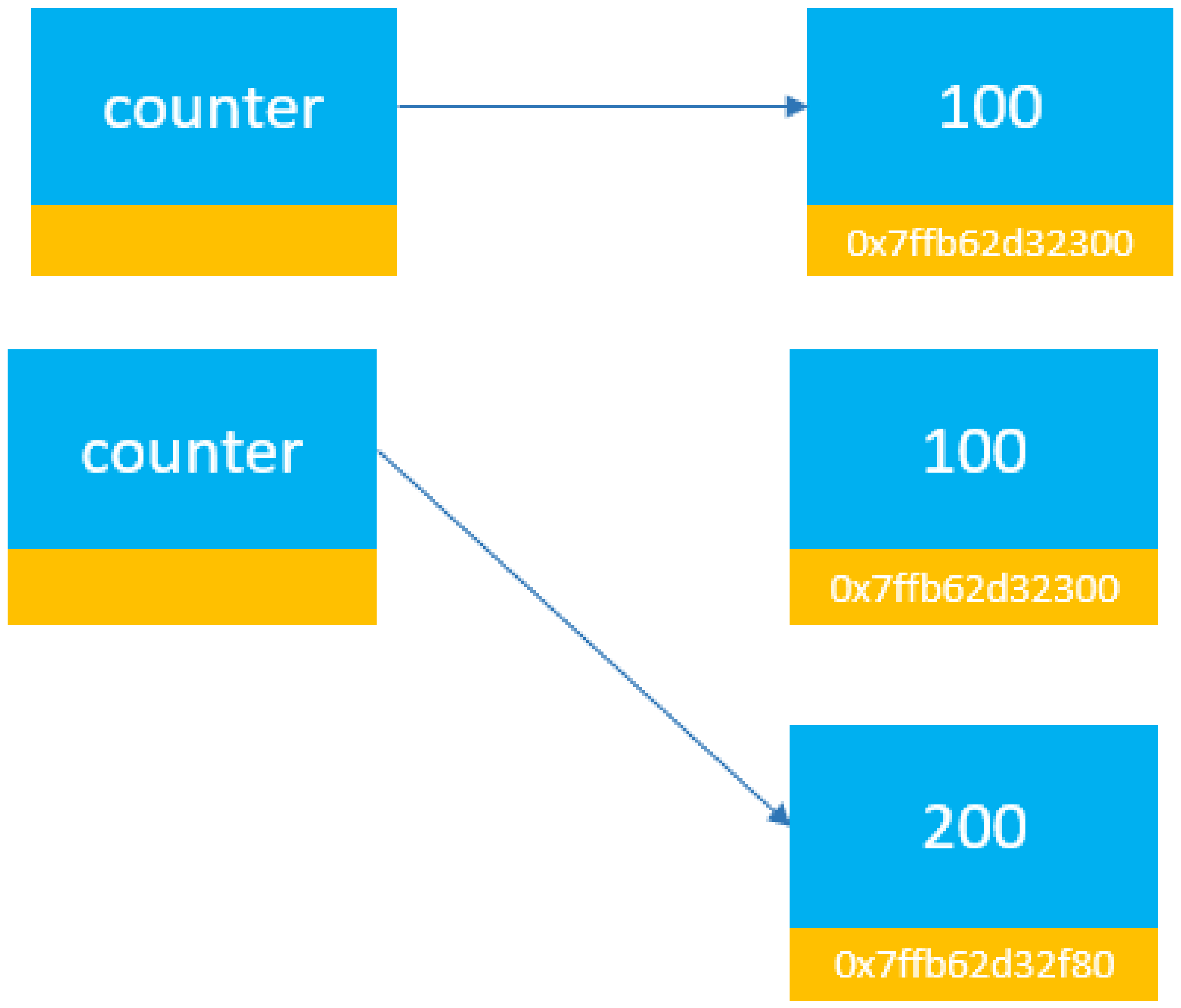
- Numbers (int, float, bool,…)
- Strings
- Tuples
- Frozen sets

And the following are examples of mutable objects:

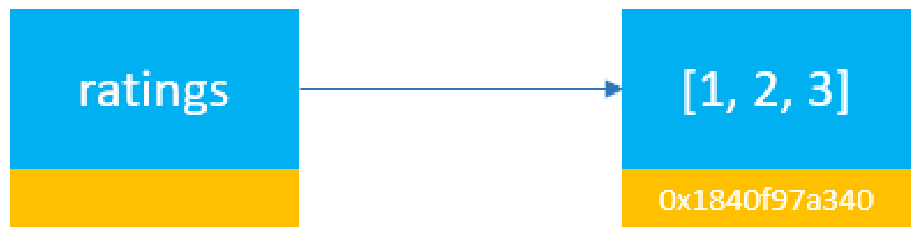
- Lists
- Sets
- Dictionaries

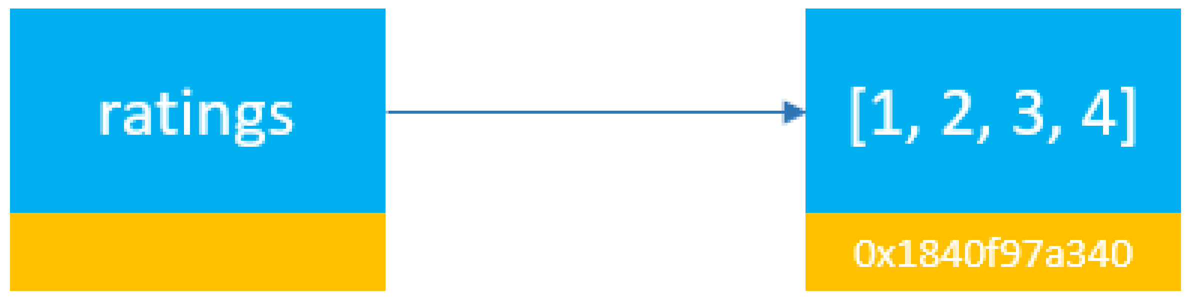
User-defined classes can be mutable or immutable, depending on whether their internal state can be changed or not.

Immutable Data Type Illustration



Mutable Data Type Illustration





Operators

- **Python Operators** in general are used to perform operations on values and variables. These are standard symbols used for the purpose of logical and arithmetic operations. In this article, we will look into different types of Python operators.

Arithmetic Operators

- Arithmetic operators are used to performing mathematical operations like addition, subtraction, multiplication, and division.

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

Comparison **Operators**

- Comparison of Relational operators compares the values. It either returns **True** or **False** according to the condition.

Operator	Description	Syntax
>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the 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 the left operand is greater than or equal to the right	x >= y
<=	Less than or equal to True if the left operand is less than or equal to the right	x <= y

Logical Operators

- Logical operators perform **Logical AND**, **Logical OR**, and **Logical NOT** operations. It is used to combine conditional statements.

Operator	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

Assignment Operators

- Assignment operators are used to assigning values to the variables.

Operator	Description	Syntax
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add AND: Add right-side operand with left side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign the result to left operand	a%=b a=a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b a=a**b

Identity Operators

- **is** and **is not** are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.
- `**Is**-- True if the operands are identical`
- `**is not**--True if the operands are not identical`

Membership Operators

- **in** and **not in** are the membership operators; used to test whether a value or variable is in a sequence.
- **`in`**`--True if value is found in the sequence`
- **`not in`**`--True if value is not found in the sequence`

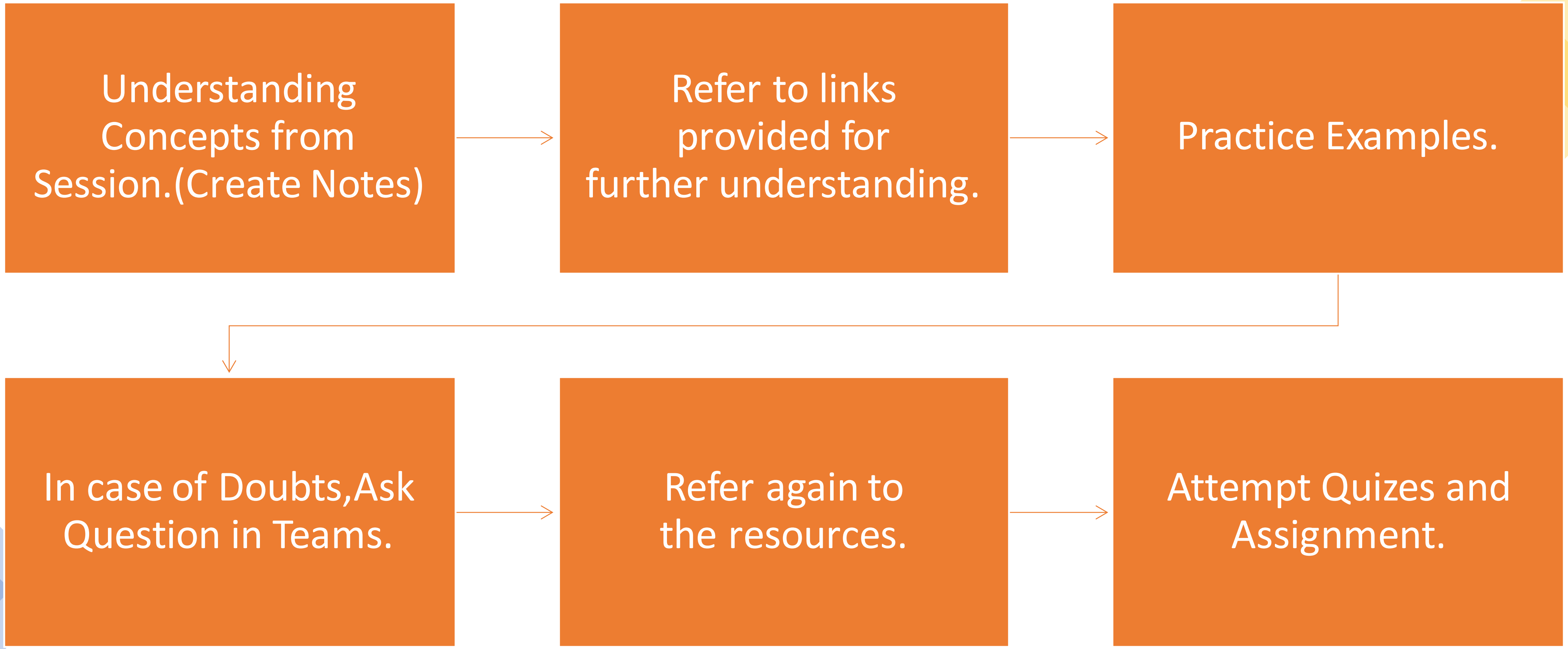
Ternary Operator in Python

- Ternary operators are also known as conditional expressions are operators that evaluate something based on a condition being true or false. It was added to Python in version 2.5.
- It simply allows testing a condition in a **single line** replacing the multiline if-else making the code compact.

Difference between == and is operator in Python

- The Equality operator (==) compares the values of both the operands and checks for value equality. Whereas the ‘**is**’ operator checks whether both the operands refer to the same object or not (present in the same memory location).

Approach to learning Python


Understanding Concepts from Session.(Create Notes)
Refer to links provided for further understanding.
Practice Examples.
In case of Doubts,Ask Question in Teams.
Refer again to the resources.
Attempt Quizes and Assignment.

Anyone ??