Python Notes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is python?

* Python is a high-level programming language created by Guido Van Rossum - fondly known as Benevolent Dictator for Life.
* Python was first released in 1991. Today Python interpreters are available for many Operating Systems including Windows and Linux.
* Python programmers are often called Pythonists or Pythonistas.

Python Features: -

* Python is a interpreted language, so it doesn’t need to be compiled before execution.
* Python follows an object-oriented programming.
* Python is general purpose language. i.e. python find it is in various domains
* **Free and Open Source**

### **Easy to code**

**Python Latest & Upcoming Features**

Python recently release Python 3.12 in October 2023 and here in this section we have mentioned all the features that Python 3.12 offer. Along with this we have also mentioned the lasted trends.

* **Security Fix:** A critical security patch addressing potential vulnerabilities (details not publicly disclosed).
* **SBOM (Software Bill of Materials) Documents:** Availability of SBOM documents for C Python, improving transparency in the software supply chain.

**Expected Upcoming Features of Python 3.13**

* **Pattern Matching (PEP 635):** A powerful new syntax for pattern matching, potentially similar to features found in languages like Ruby. This could significantly improve code readability and maintainability.
* **Union Typing Enhancements (PEP 647):** Extending type annotations for unions, allowing for more precise type definitions and improved static type checking.

**Python history and versions:**

* Python was initially designed by Guido van Rossum in 1991.
* Guido started working on Python as a hobby project during Christmas in 1989 at Centrum Wiskunde & Informatica (CWI) in the Netherlands.
* The inspiration for the name “Python” came from the BBC TV show called “Monty Python’s Flying Circus,” which Guido was a fan of. [He wanted a short, unique, and slightly mysterious name for his invention, and thus Python was born](https://www.geeksforgeeks.org/history-of-python/" \t "_blank)
* **Evolution of Python:**

Python **1.0** was released in **January 1994**, laying the foundation for the language’s syntax and structure.

Subsequent versions brought significant changes:

**Python 2.0** (released in **2000**) introduced features like list comprehensions, garbage collection, and Unicode support.

**Python 3.0** (released in **2008**) brought further improvements, including a reimagined print statement and better Unicode handling.

[Python’s design philosophy emphasizes code readability and developer productivity](https://www.tutorialsfreak.com/python-tutorial/python-history" \t "_blank)[2](https://www.tutorialsfreak.com/python-tutorial/python-history" \t "_blank).

* **Current State:**

Python continues to evolve, with the latest stable version being **Python 3.12.1**.

[It is widely used by top technology organizations like Dropbox, Google, Quora, Mozilla, and more due to its elegance, simplicity, and versatility](https://www.geeksforgeeks.org/history-of-python/" \t "_blank).

**Variables:** -

A variable is a name given to a memory location in a program

Ex.name=” XYZ” //variables

Age=23

Price=25.99

Identifiers and Keywords:

* Python is a case sensitive language.
* Python identifier is a name used to identify a variable, function, class, module, or other object.

Rules for creating identifiers: -

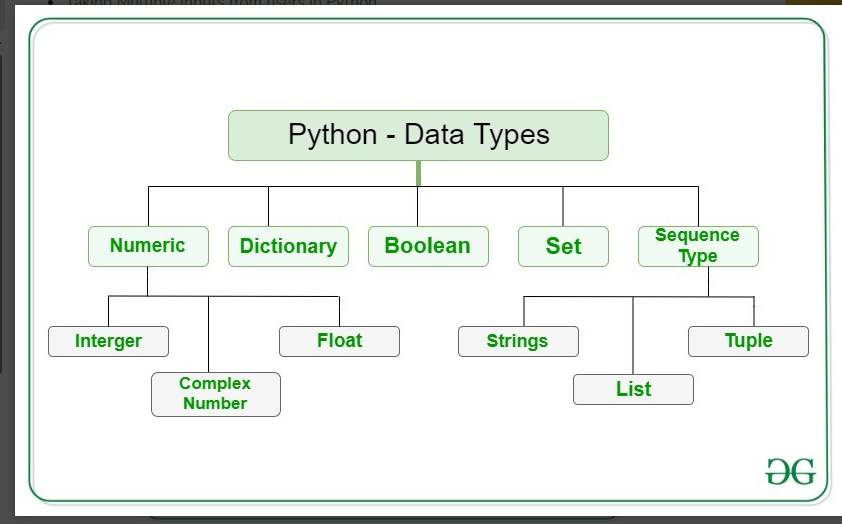
- Starts with alphabet or an underscore.

- Followed by zero or more letters, and digits.

- keyword cannot be used as identifier.

All keywords are in lowercase

Data types:



Python supports 3 categories of data types:

Basic data types: -

* Integer: -

int can be expressed in binary, decimal, octal, hexadecimal

binary starts with 0b/0B, octal with 0o/0O, hex with 0x/0X 0b10111, 156, 0o432, 0x4A3

* String: -

string is an immutable collection of Unicode characters enclosed # within ' ', " " or """ """. 'Razzmatazz', "Razzmatazz", """Razzmatazz"""

* Float: -

Float can be expressed in fractional or exponential form - 314.1528, 3.141528e2, 3.141528E2

* Boolean: -

bool can take any of the two Boolean values both starting in caps True, False

* None: -

**Container types –**

Container types typically refer to multiple values stored together. Examples of different basic types are given below:

* **List:-**

Container is an entity which contains multiple data items. It is also known as a collection or a compound data type.

list is a indexed collection of similar/dissimilar entities [10, 20, 30, 40, 50, 20, 30, 10], ['She', 'sold', 10, 'shells'’]

**Stack Data Structure:-**

* Adding an element to a stack is called push operation and removing an element from it is called pop operation. Both these operations are carried are carried out at the rear end of the list.
* Push and pop operations can be carried out using the append( ) and pop( ) methods of list object.
* **Tuple:-**
* Though a list can store dissimilar data, it is commonly used for storing similar data.
* Though a tuple can store similar data it is commonly used for storing dissimilar data . The tuple data is enclosed within ( )
* By default, sorted( ) sorts by first item in list/tuple, i.e. name. x If we wish to sort by salary, we need to use the itemgetter( ) function of operator module. x The key parameter of sorted( ) requires a key function (to be applied to objects to be sorted) rather than a single key value
* tuple is an immutable collection ('Sanjay', 34, 4500.55), ('Let Us Python', 350, 195.00)
* **set:-**
* Sets are like lists, with an exception that they do not contain duplicate entries.
* While storing an element in a set, its hash value is computed using a hashing technique to determine where it should be stored in the set.
* Since hash value of an element will always be same, no matter in which order we insert the elements in a set, they get stored in the same order.

set is a collection of unique values {10, 20, 30, 40}, {'Sanjay', 34, 45000}

* **Dict:-**
* Dictionary is a collection of key-value pairs.
* Dictionaries are also known as maps or associative arrays.
* A dictionary contains comma separated key : value pairs enclosed within { }.
* dict is a collection of key-value pairs, with unique key enclosed in ' ' {'ME101' : 'Strength of materials', 'EE101' : 'Electronics'}

**User-defined types –**

* class

Classes and Objects: -

* In Python every type is a class. So int, float, complex, bool, str, list, tuple, set, dict are all classes. These are ready-made classes. Python also permits us to create user-defined classes.
* An object is created from a class. A class describes two things—the form an object created from it will take and the methods (functions) that can be used to access and manipulate the object.
* From one class multiple objects can be created. When an object is created from a class, it is said that an instance of the class is being created.
* A class has a name, whereas objects are nameless. Since objects do not have names, they are referred using their addresses in memory.
* All the above statements can be verified through the following program.

**Operator:** -

1)Arithmetic Operators: -

Arithmetic operators perform the common mathematical operation on the numeric operands.

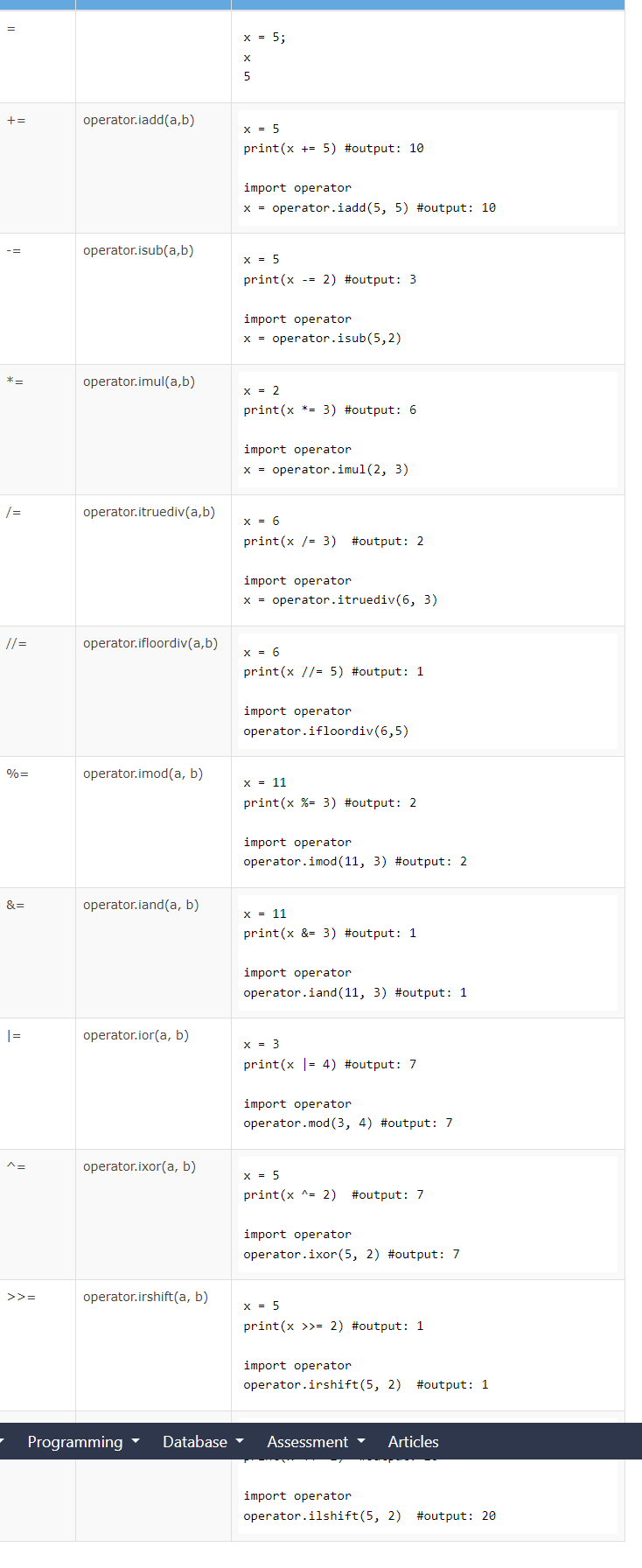
The arithmetic operators return the type of result depends on the type of operands, as below.

1. If either operand is a complex number, the result is converted to complex;
2. If either operand is a floating-point number, the result is converted to floating point
3. If both operands are integers, then the result is an integer and no conversion is needed.



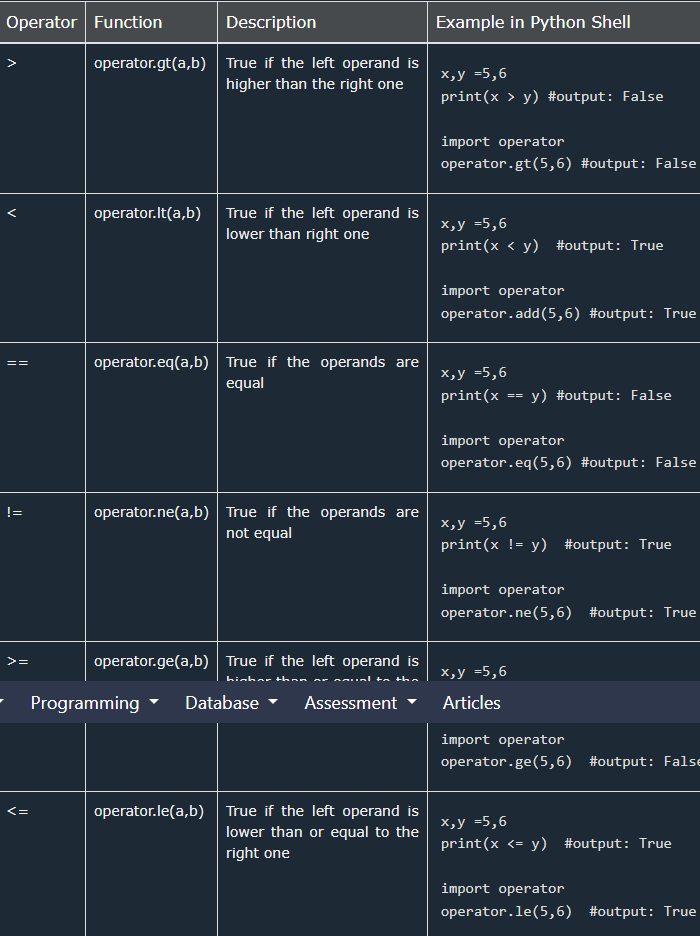
2) Assignment Operators

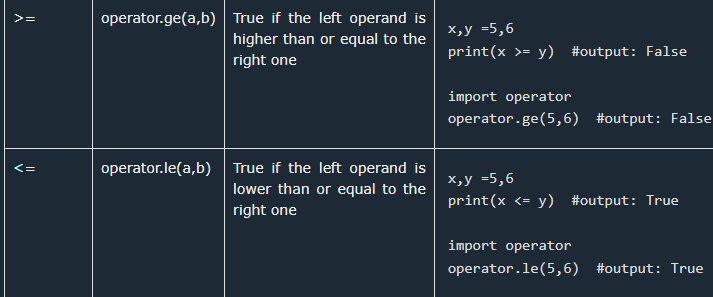
The assignment operators are used to assign values to variables. The following table lists all the arithmetic operators in Python:



3)Comparison Operators

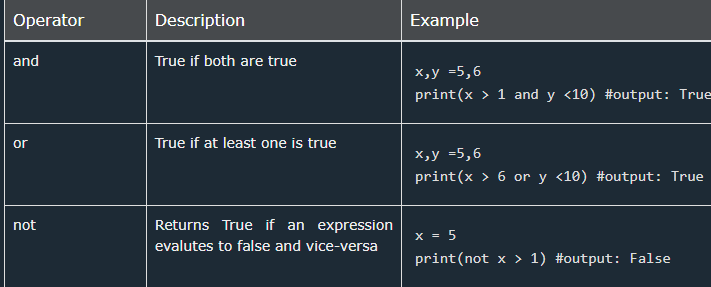
The comparison operators compare two operands and return a boolean either True or False. The following table lists comparison operators in Python.





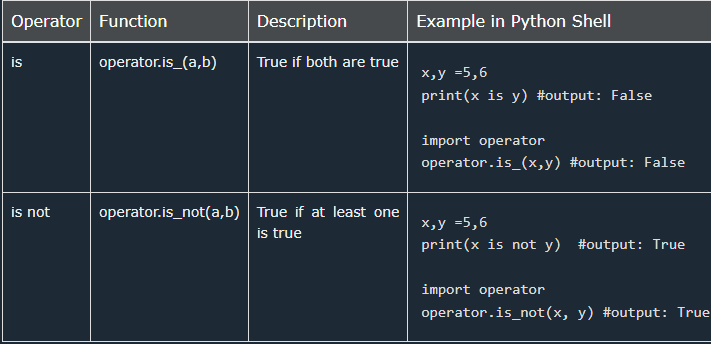
4)Logical Operators

The logical operators are used to combine two boolean expressions. The logical operations are generally applicable to all objects, and support truth tests, identity tests, and boolean operations.

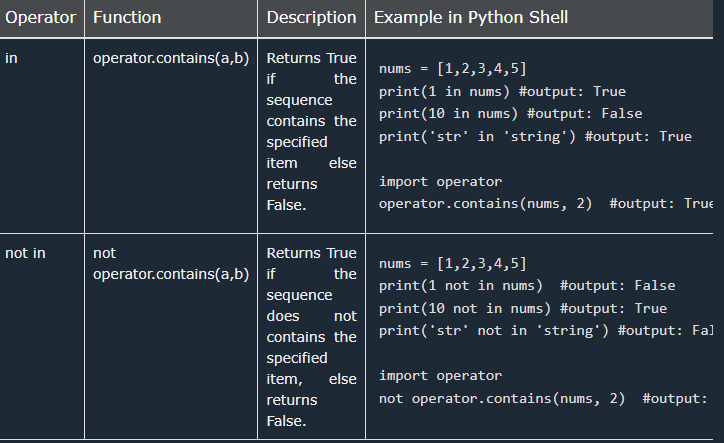


**5)Identity Operators**

The identity operators check whether the two objects have the same id value i.e. both the objects point to the same memory location

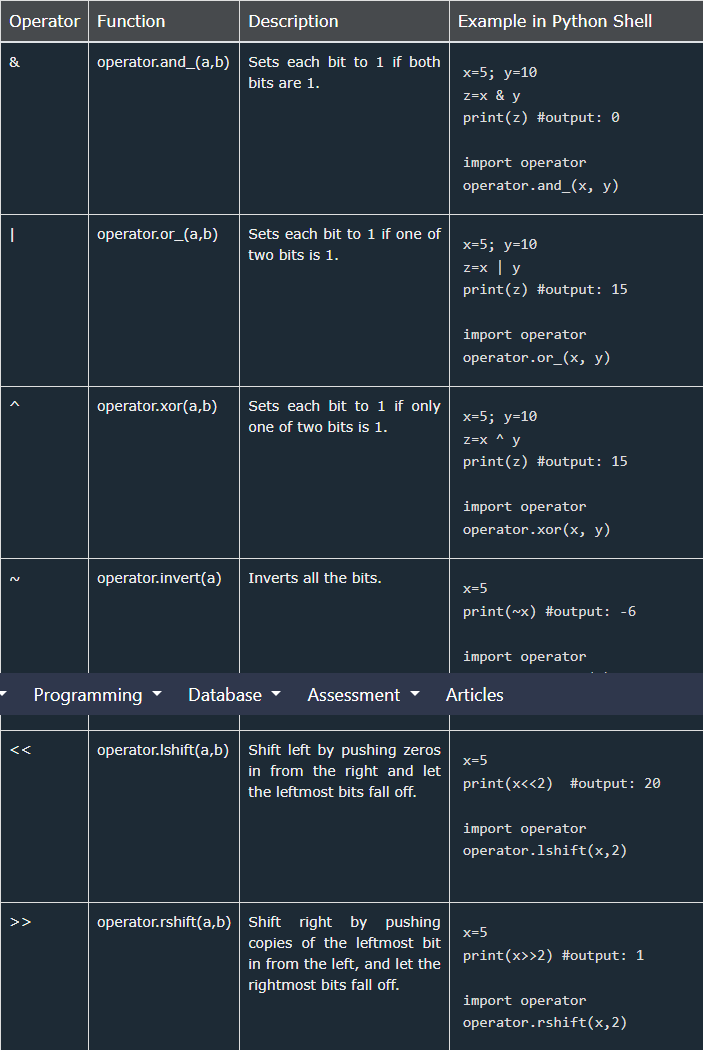
6) Membership Test Operators

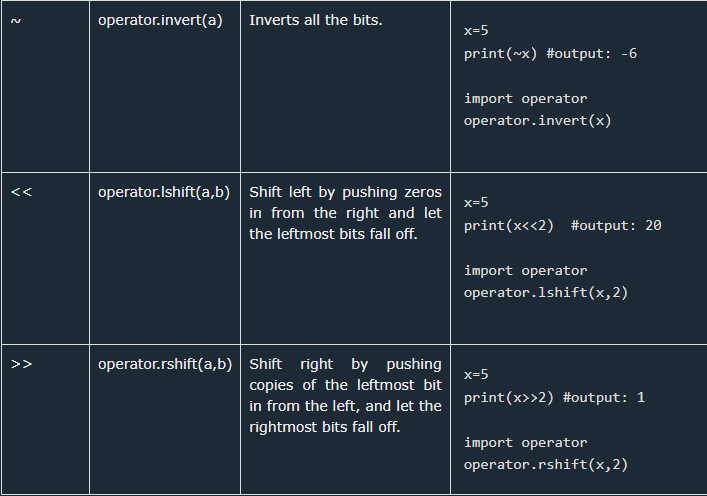
The membership test operators in and not in test whether the sequence has a given item or not. For the string and bytes types x in y is True if and only if x is a substring of y .



**7)Bitwise Operators**

Bitwise operators perform operations on binary operands.





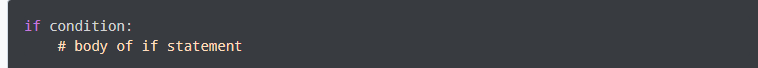
**Control statement:** -

**If statement:-**

if the condition of the if statement is:

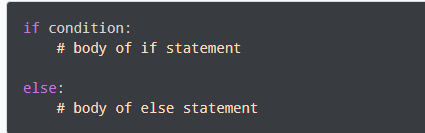
* **True** - the body of the if statement executes.
* **False** - the body of the if statement is skipped from execution.

Syntax:



If else:

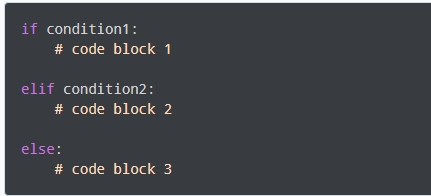
An if statement can have an optional else clause. The else statement executes if the condition in the if statement evaluates to false.



**If elif else:** -

The if else statement is used to execute a block of code among two alternatives.

However, if we need to make a choice between more than two alternatives, we use the if elif else statement

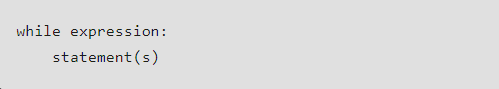


**Loop Control statement**: -

**While Loop:-**

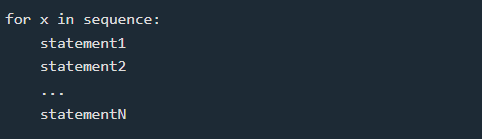
Python uses the while and for keywords to constitute a conditional loop, by which repeated execution of a block of statements is done until the specified boolean expression is true.

Syntax:



**For Loop: -**

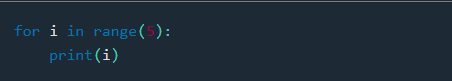
The body of the for loop is executed for each member element in the sequence. Hence, it doesn't require explicit verification of a boolean expression controlling the loop (as in the while loop).



For Loop with the range () Function: -

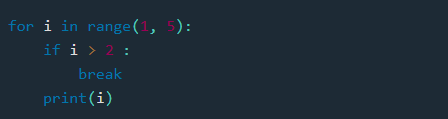
The range class is an immutable sequence type. The range () returns the range object that can be used with the for loop

Ex:



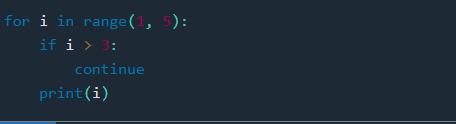
**Exit the For Loop:** -

The execution of the for loop can be stop and exit using the break keyword on some condition.



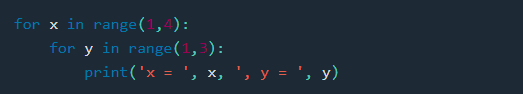
**Continue:**

Use the continue keyword to skip the current execution and continue on the next iteration using the continue keyword on some condition



**Nested for Loop:** -

If a loop (for loop or while loop) contains another loop in its body block, we say that the two loops are nested. If the outer loop is designed to perform m iterations and the inner loop is designed to perform n repetitions, the body block of the inner loop will get executed m X n times.



**Pass Statement**

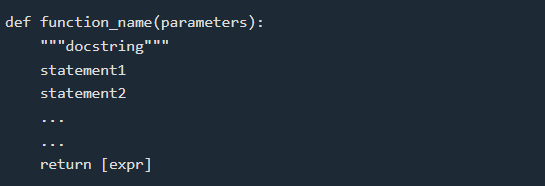
* pass statement is intended to do nothing on execution. Hence it is often called a no-op instruction.
* If we wish that on execution of a statement nothing should happen, we can achieve this using a pass statement.
* It is often used as a placeholder for unimplemented code in an if, loop, function or class. This is not a good use of pass. Instead you should use ... in its place. If you use pass it might make one believe that you actually do not intend to do anything in the if/loop/function/class.

**Python Function:-**

**Defining a Function**

A function is a reusable block of programming statements designed to perform a certain task. To define a function, Python provides the def keyword.

Syntax:



**Built-in Functions: -**

Python has many built-in functions that are always available in any part of the program. The print () function that we have been using to send output to screen is a built-in function.

Help about any built-in function is available using help(function).

Built-in functions that are commonly used with numbers are given below:

abs(x) # returns absolute value of x

pow (x, y) # returns value of x raised to y

min (x1, x2...) # returns smallest argument

max (x1, x2,…) # returns largest argument

divmod (x, y) # returns a pair (x // y, x % y)

round (x [, n]) # returns x rounded to n digits after.

bin(x) # returns binary equivalent of x

oct(x) # returns octal equivalent of x

hex(x) # returns hexadecimal equivalent of x

**Repetitions**

* There are two ways to repeat a set of statements in a function: - By using while or for loop - By calling the function from within itself
* The first method is known as iteration, whereas the second is known as recursion.
* The functions that use iteration are called iterative functions and those that use recursion are called recursive functions.

**Recursion function: -**

* A Python function can be called from within its body. When we do so it is called a recursive function.
* Recursive call keeps calling the function again and again, leading to an infinite loop.
* A provision must be made to get outside this infinite recursive loop. This is done by making the recursive call either in if block or in else block

**When to use Recursion?**

* Recursion is useful in 2 scenarios:

- When a problem can be solved by breaking it down into similar sub-problems.

- When a problem requires an unknown number of loops.

* Examples of problem as similar sub-problems:

- Finding factorial value of a number

- Finding sum of digits of an integer

- Finding binary equivalent of a number

* Examples of unknown number of nested loops:

- Finding all combinations of 1 to n, where n is received as input

- Traversing a binary tree data structure

- Traversing a graph data structure

**Anonymous Function:-**

A lambda function is a small anonymous function.A lambda function can take any number of arguments, but can only have one expression.

Example:

x = lambda a: a + 10

print(x(5))