Python Notes

:- Deepraj Vadhwane

Python Notes

What is Python?

Python is a popular and versatile programming language that is easy to read and write, making it suitable for beginners and experts and it is a high-level, interpreted and object oriented language.

Why we should use python for data science?

Python is chosen for data science due to its simplicity, extensive libraries, and strong support for machine learning and visualization

Features of Python :-

- Simple and easy to learn
- Open source
- Interpreted Language
- Platform Independent
- Dynamic Type
- Rich Libraries
- Function Oriented and Object Oriented

What are Python modules?

Python modules are files containing Python code that define functions, classes, and variables. They allow code reuse and organization, making it easier to manage and maintain larger projects.

Comments in Python:-

Comments in Python are denoted by the # character and also denoted by (''' ''') .

Identifier:

Identifier is a variable and is used to identify the variable, function ,class, module or any user-defined object.

Rules of Identifier

- It can start with a letter (a-z,A-Z) or an underscore(_).
- It cannot start with number.
- Python is case sensitive
- It cannot be a reseved keyword (e.g:-if,while,for)

Operators:

Operators are the symbols or special characters that allow you to perform operations on variables, values, or expressions.

Types of Operators:

- Arithmetic Operator
- Logical Operator
- Bitwise Operator
- Assignment Operator
- Relational Operator
- Membership Operator
- Special Operator

Arithmetic Operators: It is used for basic mathematical operations.

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Modulus (%)
- Exponentiation (**)
- Floor Division (//)

Comparison Operators: Used to compare values.

- == (Equal to)
- != (Not equal to)
- (Greater than)
- < (Less than)
- = (Greater than or equal to)
- <= (Less than or equal to)

Logical Operators: Used to combine conditions.

- and (Logical AND)
- or (Logical OR)
- not (Logical NOT)

Bitwise Operators: Used to perform bitwise operations on integers.

- & (Bitwise AND)
- | (Bitwise OR)
- ^ (Bitwise XOR)
- ~ (Bitwise NOT)
- << (Left shift)
- >> (Right shift)

Membership Operators: Used to test if a value is a member of a sequence (e.g., list, tuple, string).

- in (True if value is present in the sequence)
- not in (True if value is not present in the sequence)

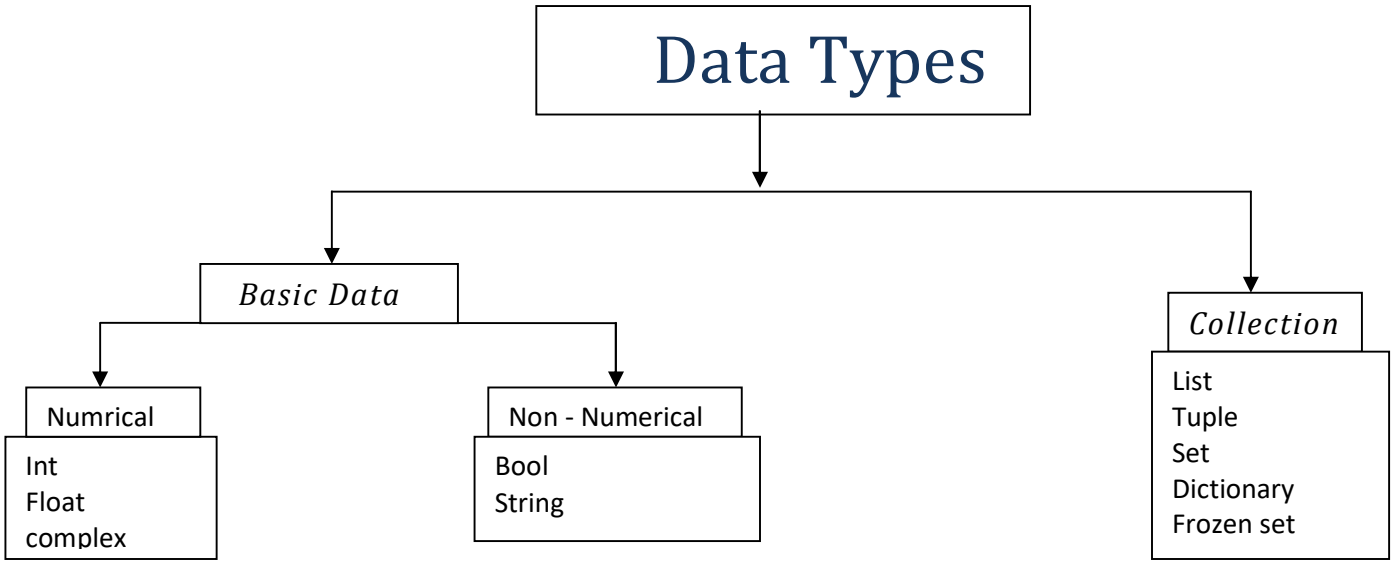
Identity Operators: Used to compare the memory location of two objects.

- is (True if both variables point to the same object)
- is not (True if both variables point to different objects)

Basic Funtions:

- print() :-To display the output on console
- input():- Taking the input from the user
- len():- No of values present in the variable
- id():- to check the memory address
- help():- Description and usecase
- type():- Type of data is

Data types:- Data types in Python represent the type or category of data stored in variables. Python is a dynamically typed language, meaning you don't need to explicitly declare the data type when defining a variable; the interpreter determines it automatically based on the value assigned to the variable



Basic Data Types:

Numeric Data Types:-

- Integer: Used to represent whole numbers without a fractional component.
  Example: age = 25
- Float: Used to represent numbers with a fractional component.
  Example: pi = 3.14

Non-Numeric Data Types:-

- Boolean: Used to represent True or False values.
- Example: is_student = True
- String: Used to represent a sequence of characters, enclosed within single or double quotes. Example: name = "Deepraj"

String Manipulation or Methods:

1. len(): This function returns the length of the string, i.e., the number of characters in the string.

2. lower(): It converts all characters in the string to lowercase. If there are any non-alphabetic characters, they remain unchanged.

3. upper(): It converts all characters in the string to uppercase. If there are any non-alphabetic characters, they remain unchanged.

4. strip(): This method removes leading and trailing whitespace characters (spaces, tabs, and newline) from the string. It does not modify the original string but returns a new string with the whitespace removed.

5. split(): This method splits the string into a list of substrings based on a specified separator. By default, if no separator is provided, it splits the string at whitespace characters.

6. join(): It joins elements of a list into a single string using a specified separator. The method is called on the separator string and takes the list as an argument.

7. replace(): This method replaces occurrences of a substring with another substring in the original string. It does not modify the original string but returns a new string with replacements.

8. startswith() and endswith(): These methods check if a string starts or ends with a specific substring. They return True if the string starts or ends with the specified substring; otherwise, they return False.

Indexing: Accessing single Value from the variable from the variable is called indexing and it starts from Zero index

Ex: a="Deepraj"

a[0] => "D"

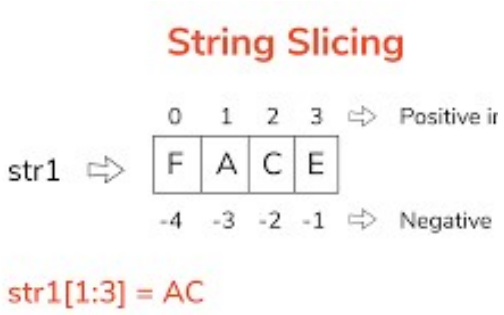
a[1]=> "e"

Slicing :-

slice means a piece [ ] operator is called slice operator,which can be used to retrieve parts of String. In Python Strings follows zero based index. The index can be either +ve or -ve. +ve index means forward direction from Left to Right -ve index means backward direction from Right to Left

slice() can take three parameters:

- start (optional) - Starting integer where the slicing of the object starts. Default to None if not provided.
- stop - Integer until which the slicing takes place. The slicing stops at index stop -1 (last element).
- step (optional) - Integer value which determines the increment between each index for slicing. Defaults to None if not provided.



Difference Between Indexing and Slicing:-

Indexing	Slicing
Indexing is used to obtain individual elements.	Slicing is used to obt of elements.
Attempting to use an index that is too large will result in an IndexError.	Out of range indexes gracefully when usec
Indexing returns one item	Slicing returns new l

Collection Data Types:-

List: It is a type of sequential data type which hold multiple heterogeneous values separated by comma and enclosed in square brackets.
Example: grades = [85, 92, 78, 95], a=[1,2,3,5,2.0,65.0,'lst']

Attributes of List:

- Mutable:- we can modify /change the original data
- Ordered:- Index based
- Duplicates are allowed
- Index Base

CRUD → Create, Read, Update, Delete,

- Create :- we can create a empty [ ], list() ,list with values
- Read:- we can read the data using For loop, Indexing, Slicing,
- Update : We can update the valued using append and extend method
- Delete: we can delete the values using remove,pop,del and clear method.

List Manipulation or Methods:-

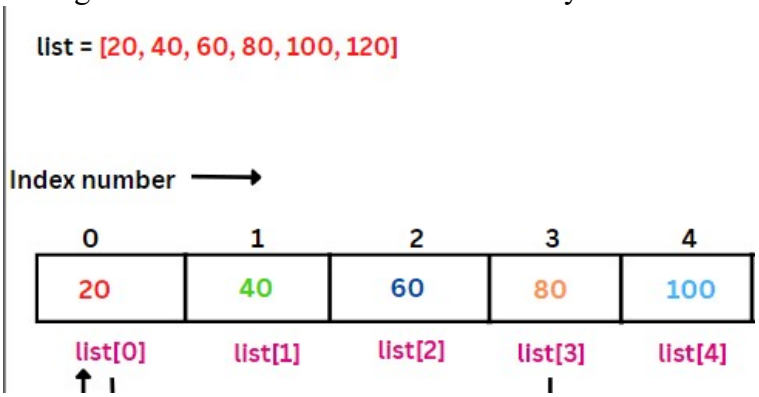
1. append(): This method adds an element to the end of the list. It modifies the original list and does not return a new list.
2. extend(): It extends the list by appending elements from another iterable (e.g., another list, tuple, string, etc.) to the end. It modifies the original list and does not return a new list.
3. insert(): This method inserts an element at a specific index in the list, shifting the other elements to the right. It modifies the original list and does not return a new list.
4. remove(): It removes the first occurrence of a specified value from the list. If the value is not found, it raises a ValueError.
5. pop(): This method removes and returns the element at a specified index in the list. If no index is provided, it removes and returns the last element.
6. index(): It returns the index of the first occurrence of a specified value in the list. If the value is not found, it raises a ValueError.
7. count(): This method returns the number of occurrences of a specified element in the list.

8. sort(): It sorts the elements of the list in ascending order. The original list is modified, and the method does not return a new list. Optionally, you can use the reverse=True argument to sort in descending order.

9. reverse(): This method reverses the order of elements in the list. The original list is modified, and the method does not return a new list.

10. clear(): It removes all elements from the list, making it empty. This method modifies the original list and does not return a new list.

List Slicing: Slicing the list it starts from zero and ends by n-1



Tuple: It is a sequential data type which can hold multiple heterogeneous values separated by comma, enclosed in parenthesis. Tuple parenthesis are optional. (100, 200, 21.33, "INDIA") // 100, 200, 21.33, "INDIA"

Attributes of List:

- Immutable:- we cannot modify /change the original data
- Ordered:- Index based
- Duplicates are allowed
- Index Base

CRUD → Create, Read, Update, Delete,

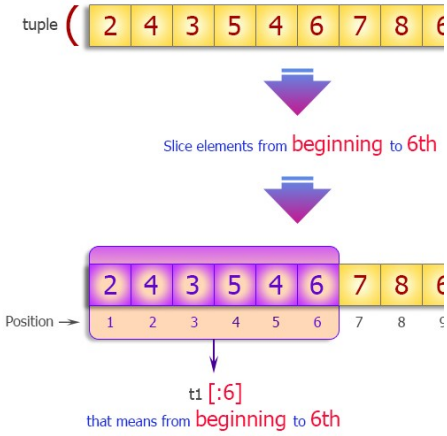
- Create :- we can create a empty tuple (), tuple() ,tuple with values
- Read:- we can read the data using For loop, Indexing, Slicing,
- Update : We can't perform any insertion, replacement, element deletion with tuple
- Delete :- we can delete entire tuple.

Tuple Manipulation or Methods:-

1. index(): This method returns the index of the first occurrence of a specified value in the tuple. If the value is not found, it raises a ValueError.

2. count(): It returns the number of occurrences of a specified element in the tuple.

Tuple Slicing: It is similar like list and string slicing.



Set: Used to store a collection of unique elements (no duplicates).

Example: unique_numbers = {1, 2, 3, 4, 5}

Attributes of List:

- Mutable:- we can modify /change the original data
- Not Ordered:- Not Index based
- No Duplicates are allowed
- No Index Base

CRUD → Create, Read, Update, Delete.

- Create :- we can create a empty set().
- Read:- we can read the data using For loop, Indexing, Slicing,
- Update : We can update the data using add and update method
- Delete :- we can delete the data by using Remove,pop clear and del

Set Manipulation or Methods:-

1. add(): This method returns the addition of two numbers in a set.
2. clear():It removes all elements from the set, making it empty
3. difference():To get the difference between two sets, you can use the difference() method.

4. difference_update():This method check the difference of two numbers and update the set.
5. discard(): It removes the elements from the set.
6. remove(): It removes the elements and return the set.
7. intersection(): To get the intersection of two sets, you can use the intersection() method.
8. isdisjoin(): This method is used to check if two sets are disjoint. Two sets are considered disjoint if they have no elements in common, meaning their intersection is an empty set {}.
9. issubset(): This method is used to check whether a set is a subset of another set. A set A is considered a subset of another set B if all the elements of set A are present in set B.
10. isupper(): This method is used to check whether a set character is uppercase or not.
11. islower(): This method is used to check whether a set character is lowercase or not.
12. union():This method returns a new set containing all the elements that are present in either the original set or the set provided as an argument.
13. pop():This method remove the last element of an set and return a new set containing all the elements except last element.

Dictionary:- Used to store key-value pairs, providing a way to access values based on their corresponding keys.
Example: student = {"name": "Alice", "age": 21, "major": "Computer Science"}

CRUD → Create, Read, Update, Delete.

- Create :- we can create a empty dictionary {},dict() .
- Read:- we can read the data using For loop, keys, get, method
- Update : We can update the data using update, setdefault method
- Delete :- we can delete the data by using del, pop , popitem, clear.

Dictionary Manipulation or Methods:-

1. dict.keys(): Returns a view object that displays a list of all the keys in the dictionary.
2. dict.values(): Returns a view object that displays a list of all the values in the dictionary.
3. dict.items(): Returns a view object that displays a list of key-value pairs in the dictionary as tuples.

4. dict.get(key, default=None): Returns the value associated with the given key. If the key is not found, it returns the default value (None by default).

5. dict.pop(key, default=None): Removes the key from the dictionary and returns its value. If the key is not found, it returns the default value (None by default).

6. dict.popitem(): Removes and returns the last key-value pair inserted into the dictionary as a tuple.

7. dict.clear(): Removes all the key-value pairs from the dictionary, making it an empty dictionary.

8. dict.update(iterable): Updates the dictionary with the key-value pairs from the iterable.

9. dict.fromkeys(iterable, value=None): Creates a new dictionary with keys from the iterable and values set to the provided value (None by default).

10. dict.setdefault(key, default=None): Returns the value associated with the given key. If the key is not found, it inserts the key with the default value into the dictionary.

11. len(dict): Returns the number of key-value pairs in the dictionary.

Difference Between List, Tuple, Dictionary and Set

LIST	TUPLE	DICTIONARY	SET
Allows duplicate members	Allows duplicate members	No duplicate members	No duplicate members
Changeble	Not changeable	Changeable indexed	Cannot be changed, but can be added, non -indexed
Ordered	Ordered	Unordered	Unordered
Square bracket [ ]	Round brackets ( )	Curly brackets{ }	Curly brackets{ }

Flow Control

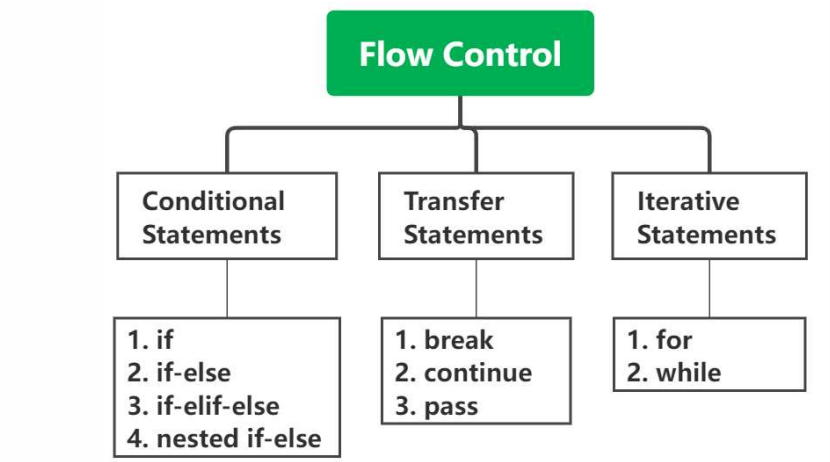
Flow control in Python allows you to control the execution flow of a program based on certain conditions. Python provides various constructs for flow control, including conditional statements

and loops. These control structures enable you to make decisions, perform repetitive tasks, and execute specific code blocks under different conditions.

Control Flow Statements

The flow control statements are divided into three categories

1. Conditional statements
2. Iterative statements.
3. Transfer statements



Conditional statements

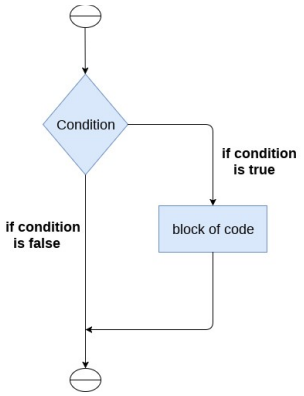
In Python, condition statements act depending on whether a given condition is true or false. You can execute different blocks of codes depending on the outcome of a condition. Condition statements always evaluate to either True or False.

There are three types of conditional statements.

1. if statement
2. if-else
3. if-elif-else
4. nested if-else

1. if statement:-

- If condition is evaluated to True, the code inside the body of if is executed.
- If condition is evaluated to False, the code inside the body of if is skipped.if-else



2. if-else :-

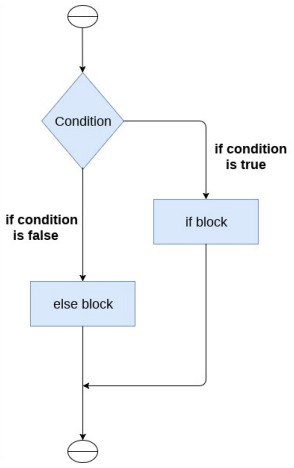
The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition.

If the condition evaluates to True,

- the code inside if is executed
- the code inside else is skipped

If the condition evaluates to False,

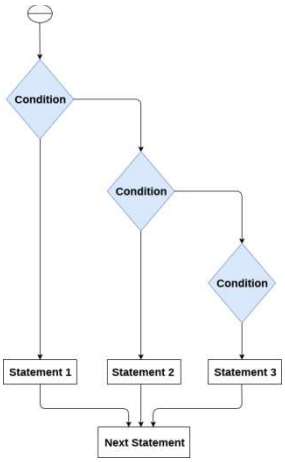
- the code inside else is executed
- the code inside if is skipped



4. if-elif-else:-

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

The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them. We can have any number of elif statements in our program depending upon our need.



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

In Python, iterative statements allow us to execute a block of code repeatedly as long as the condition is True. We also call it a loop statements.

Python provides us the following two loop statement to perform some actions repeatedly

Iterative Statement:

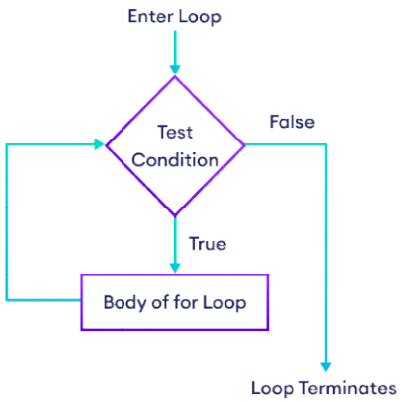
1. for loop
2. while loop

1. for loop:

For loop is used to iterate over sequence such as list, tupes, string, range ,array…

for iterator_var in sequence:

    statements(s)



Using else statement with for loop in Python

We can also combine else statement with for loop like in while loop. But as there is no condition in for loop based on which the execution will terminate so the else block will be executed immediately after for block finishes execution.

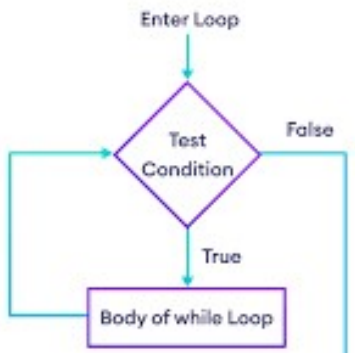
2. While loop:-While loop is used to execute a block of statements repeatedly until a given condition is satisfied. And when the condition becomes false, the line immediately after the loop in the program is executed.

while condition:

    # execute these statements

else:

    # execute these statements



Transfer statements

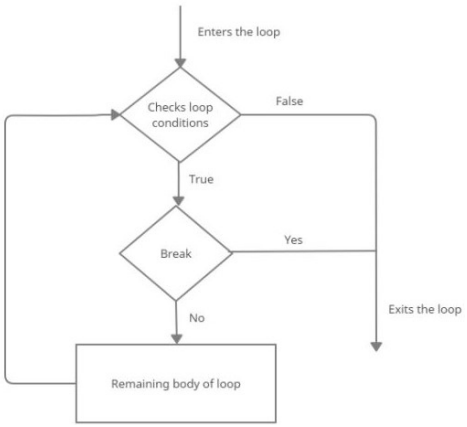
In Python, Transfer statements are used to alter the program's way of execution in a certain manner. For this purpose, we use three types of transfer statements.

1. break statement
2. continue statement
3. pass statement

1. break statement:- In programming, the "break" statement is a control flow statement used to exit or terminate a loop prematurely. It is commonly found in loops (such as "for" loops, "while" loops, or "do-while" loops) and switch statements. When the "break" statement is encountered

inside a loop or switch, the execution of the loop or switch is immediately halted, and the program continues with the next statement after the loop or switch.

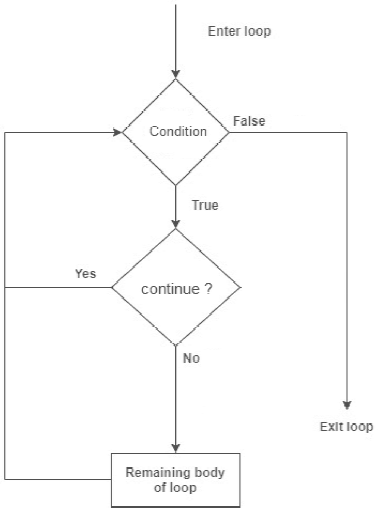
The primary purpose of using the "break" statement is to exit the loop early based on a certain condition. It allows you to stop the loop's execution once a specific condition is meet, rather than continuing to iterate through the rest of the loop's statements.



2.Continue Statement:- In programming, the "continue" statement is another control flow statement that is used within loops to skip the current iteration and move on to the next iteration without executing the remaining statements within the loop body for the current iteration.

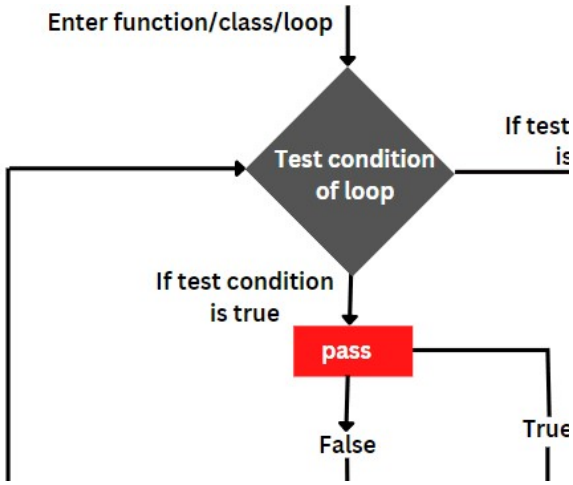
The primary purpose of using the "continue" statement is to control the flow of a loop and selectively skip some iterations based on specific conditions.

When the "continue" statement is encountered, it immediately stops the current iteration and jumps to the beginning of the loop for the next iteration, bypassing any code that follows the "continue" statement for the current iteration.



3.pass statement:- In programming, the "pass" statement is a no-operation (NOOP) statement. It is used when a statement is syntactically required but you don't want to execute any code or perform any action at that point in the program. The "pass" statement essentially does nothing and serves as a placeholder to avoid syntax errors when a block of code is expected but you don't want to put any code in that block.

The "pass" statement is particularly useful in situations where you are writing code placeholders or stubs for future implementation. It allows you to create the structure of a function, class, or conditional block without adding the actual logic yet.



Range() : It is a function which generates the values accepting three parameters

Starting value : Starting value

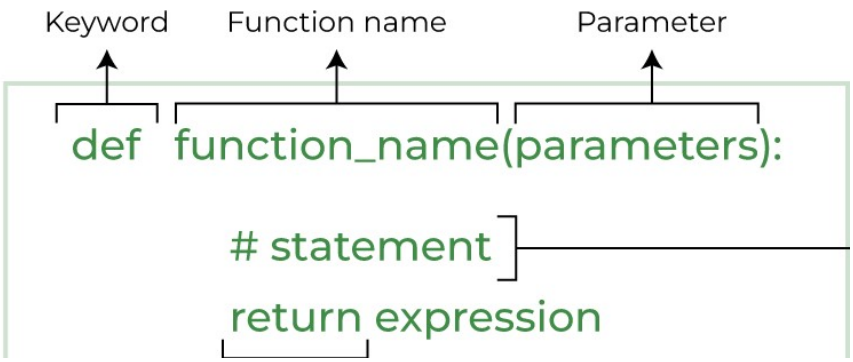
Ending value : Ending value

Step : it defines how to display generated values.

Syntax : range(start , end , step)

Functions :

It is a repeated block of code which consists of business logic. This block can be called "n" number of times based on the requirement.



Functions consists of two parts :

1. Function Definition
2. Function Calling

1. Function Defination:

- Function declaration:- It is first line of the definition with "def" keyword
- Function Implementation:- Block of code under function definition is called function implementation.
- Function Signature : No: of argument which we pass to function is called function signature.

Function definition should start with "def" keyword followed by which identifies the function definition uniquely. Function definition should end with parenthesis Function definition can exist without calling.

Function definition should be defined before function calling and We can define multiple function definition within the same program. Function definition is executed when it is called.

2. Function Calling:- Function definition is executed when it is called.

- Function calling should be always mentioned after definition of the function.
- Function calling can be done n number of time.
- Function calling cannot exist without function definition. It triggers error.

Types of Variables

1. Local Variables
2. Global Variable

1. Local Variable:- Variables which are defined inside the function is called local variable which can be accessed within that function only.

2. Global Variables: Variables which are declared with "global" keyword are called global variables which can be accessed anywhere with the program. and Variables which are declared outside the functions are also considered as global variables.

Types of Arguments

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Varaible Length Argument

1. Positional Arguments : When arguments are assigned with respective values based on position is called positional argument.

Positional Arguments should be ahead of all other arguments

2. Keyword Arguments : Arguments which are initialized with their respective values at function calling place is called keyword argument

3. Default Arguments :- Arguments which are initialized with default values at function definition is called Default Argument.

3. Variable Length Argument: It is special type of argument which can accept "n" Number of positional values.

These values are stored in the form of tuple

Variable length argument are recommended with variable "*args"

4. Keyword Variable Length Argument: It is special type of argument which can accept "n" number of keyword argument

These arguments are stored in the form of dictionary

Keyword variable length argument is recommended with ** kwargs.

Note:- In this Notes I have cover theory part so that if any want to learn python in a esay way or recap concepts then you can learn it