**uses of python** web development**/** gaming**/** cyber security/hacking**/** tool designing**/** software**/** iot**/** cryptocurrency**/** app development**/** ML, AI, DL**/** animation **what is python** python is a high-level, general-purpose and a very popular programming language**/** It support interpreted and object-oriented scripting language**/** Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it**/** Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects**/** Python is a Beginner's Language − Python is a great language for the beginner-level programmers **Programming** A program is executed (the code is first compiled and generate the output) **/** A program is general is a sequence of instructions written so that a computer can perform certain task **Scripting** A script is interpreted(A script is a code written in Scripting language ) **/** It is a type of programming language in which you can write code to control another software application **history** Python was developed by Guido Van Rossum, a Dutch programmer in the late 80’s. Guido Van Rossum had been working on the development of ABC language in a computer science research institute named Centrum Wiskunde & Informatica (CWI) in the Netherlands. In 1991, Van Rossum conceived and published Python as a successor of ABC language**/** For many uninitiated people, the word Python is related to a species of snake. Rossum though attributes the choice of the name Python to a popula comedy series "Monty Python's Flying Circus" on BBC**/** Open sourced from the beginning **Features** Easy-to-learn**/** Easy-to-read**/** Easy-to-maintain**/** Portable**/** Databases**/** Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code**/** GUI Programming − Python supports GUI applications to create Software**/** Scalable − Python provides a better structure and support for large programs than shell scripting **Escape sequence** An Escape character is denoted by (\)**/** Use:- To insert the character which is not defined syntax**/** List of escape character (mostly use..) **/** To print quote: - \’, \’’ **/** To print next line: - \n**/** To print space(tab): - \t**/** To print backslash: - \\ (for single), \\\\ (for double) **example** to print quote: - \’, \’’**/** To print next line: - \n**/** to print tab: - \t**/** to print next line: - \\ (for single \), \\\\ (for double \\). **Comments** is a statement which is used for to explain the code, Or we can say that it help to code readable and prevent form execution**/** It represent with #**/** Compiler or interpreter can not run comments**/** Types of comment**/** Single Line comment**/** Multi line comment **Variable** are data values which stored anything or we can say it is types of container to store**/** It help to assign the values example x = 5**/** y = "John"**/** print(x)**/** print(y) **rule for variables** A variable can be start with a letter or the underscore character/ A variable name cannot start with a number**/** digit(1,2,,467…..87..7.87)**/** A variable name can be start alpha-numeric characters and underscores (A-z, 0-9, and \_ )**/** Variable names are case-sensitive **data types** **string** is a sequence of characters enclosed within either single quotes (' ') or double quotes (" ") ex: my\_string = 'python'**/** print(my\_string) **numeric** data type is used to hold numeric values**/** int: holds signed integers of non-limited length**/** float: holds floating decimal points and it's accurate up to **15** decimal places ex: num1 = 5**/** print(num1)**/** num1 = 5.0**/** print(num1) **List** is a collection data type. It allows multiple values to be stored within the same field ex: x = ["apple", "banana", "cherry"]**/** print(x) [**Tuples**](https://www.geeksforgeeks.org/python-tuples/) is an immutable collection of that are more like lists. Python Provides a couple of methods to work with tuples two method of tuple; count() method**/** index()methodlist example ma [] hata ka () lagana ha **Dictionary** is a collection of keys values, used to store data values like a map, which, unlike other data types which hold only a single value as an element ex: Dict **=** {1: 'Geeks', 2: 'For', 3: 'Geeks'}**/** print(Dict) **Set** are represented by { } (values enclosed in curly braces)ex: var = {"Geeks", "for", "Geeks"}**/** type(var) **Input** is a function which help to take value or input from user and tore in the variables Syntax:- input word + ()= input() ex: Name=input(“enter your name”)**/** print(name) **statement** Instructions that a Python interpreter can execute are called statements **indentation** To define the block code for understanding in some language we use curly braces{} like c, c++ ,java. The enforcement of indentation in Python makes the code look neat and clean. **operators** [Arithmetic operators](https://www.geeksforgeeks.org/python-arithmetic-operators/) are used to perform basic mathematical operations like**addition, subtraction, multiplication,** and**division** [**Comparison**](https://www.geeksforgeeks.org/python-object-comparison-is-vs/) operatorsof[Relational operators](https://www.geeksforgeeks.org/relational-operators-in-python/) compares the values. It either returns **True** or **False** according to the condition(<,>,=,==,<=,>=).

[**Logical operators**](https://www.geeksforgeeks.org/python-logical-operators-with-examples-improvement-needed/) perform **Logical AND**, **Logical OR**, and**Logical NOT** operations. It is used to combine conditional statements [**Bitwise operators**](https://www.geeksforgeeks.org/python-bitwise-operators/) act on bits and perform bit-by-bit operations. These are used to operate on binary numbers (&, |,~,^,>>,<<)  [**Assignment operators**](https://www.geeksforgeeks.org/assignment-operators-in-python/) are used to assign values to the variables (=,+=,-=,\*=,/=) [**identity operators**](https://www.geeksforgeeks.org/python-membership-identity-operators-not-not/) **is** and **is not** are the [identity operators](https://www.geeksforgeeks.org/python-membership-identity-operators-not-not/) both are used to check if two values are located on the same part of the memory **Membership Operators** **in** and **not in** are the[membership operators](https://www.geeksforgeeks.org/python-membership-identity-operators-not-not/) that are used to test whether a value or variable is in a sequence **list** is data type which help to store more than one values .or it help to store various type of data(integer, float, string…)**/** List is only in python ,in other language we use array**/** To make list use []and separate the item by (,) comma**/** It is mutable ,ordered , can access by index, allow Duplicates**/** Var =[“India”,”Poland”,”China”,”Japan”,”Germany”]**/** print(Var) **Indexing** is a way to traversing or access the elements or values in programming ex: print(country[0])**/** print(country[1])**/** print(country[1/0]) **negative indexing** ex: print(country[-1])**/** print(country[-2])**/** print(country[-3]) **get element from list** It means we can access the every element in the list or we can change the element as per need**/** With the help if index number we can do this ,the index number start with 0(zero) ex: country[2]=”bali”**/** print(country) **add new element** append:- this method add elements in the last of the list**/** insert:- this method help to add elements in the middle of list**/** extend:- it help to merge the 2 list or elements ex: country.append(“Russia”) **remove new element** Remove :- particularly we can remove any elements**/** pop:- it help to remove the elements by index number**/** del:- it can remove the elements by the index number as well as delete all elements from the list**/** clear:- this method help to empty the list ex: country.remove("japan")**/** country.pop(2)**/** del country[4]**/** country.clear() **Tuples** is a collection of values separated by comma()/ It is ordered and immutable/ It use parentheses() ,it allows duplicate value/ \*\* it is the collection of string (more then one elements) ex: food\_categories=("Vegetable", "Fruits", "Dairy", "Grains", "herbs", "seafood")**/** print (food\_categories)

**method** min():- it gives you minimum value from the tuple/ max();- it gives you maximum value from the tuple**/** len():- it gives you length of the tuples**/** index():- it start with zero(0),and you can traverse**/** count():- it show number of that elements present in the list **Sets** are the unordered , unchangeable , collection of data**/** Set elements are unique. Duplicate elements are not allowed**/** A set itself may be modified, but the elements contained in the set must be of an immutable type**/** It represent with {} curly braces**/**bracket Add: - This function help to add element in set**/** Update: - This function help to update the from other list or elements ex: country.add(“jk”)**/** country.update(3,”china”) **Dictionary** is the unordered collection of data values**/** Is used (:)colon to separate the columns**/** It Is mutable , dynamic and do not allow duplicates**/** It contains key : value and separated by commas ex: var **=** {car: 'bmw', bike: 'tvs', bus: 'tata'}**/** print(var) **String** are the collection of words and character**/** String use single and double quotes to represent itself **method** Upper:- This method used to change the alphabet from lower to upper**/** Lower:-This method used to change the alphabet from upper to lower**/** Replace :- Help to replace the string from another**/** Split:- Help to break the string or replace**/** Capitalize:- help to you capital the first letter**/** Count:- help to count the value in string**/** Find:- to find the value in the string**/** Join:- To join the two or more string ex: txt = "Hello my friends"/ x = txt.upper()/ print(x) **Conditions** are one of the most important elements that must be available in any programming language**/** There are 3 way to apply the condition in programming**/** If :- it is used to decide whether statement is execute or not by condition**/** else:-it is used to decide whether statement is execute or not by condition, but it cannot use without if else**/** elif:- it is also help to execute the condition in between if and else **if** syntax:-**/** If condition**/** If it is true then execute**/** If it is not then exit from condition statement ex: a = 33**/** b = 200**/** if b > a:**/** print("b is greater than a") **else** keyword catches anything which isn't caught by the preceding conditions syntax:-**/** If condition**/** If it is true then execute**/** If it is not then exit from condition statement ex: a = 200**/** b = 33**/** if b > a:**/** print("b is greater than a")**/** elif a == b:**/** print("a and b are equal")**/** else:**/** print("a is greater than b") **elif**keyword is Python's way of saying "if the previous conditions were not true, then try this condition" syntax:-**/**If condition**/** If it is true then execute**/** If it is not then exit from condition statement ex: a = 33**/** b = 33**/** if b > a:**/** print("b is greater than a")**/** elif a == b:**/**print("a and b are equal") **nested if** You can have if statements inside if statements, this is called nested if statements ex: x = 41**/** if x > 10:**/** print("Above ten,")**/** if x > 20:**/** print("and also above 20!")**/** else:**/** print("but not above 20.") **Looping** are the constructs that repeatedly execute a piece of code based on the conditions. When we want to execute or print one statement for more than one time with a single specification then we use the concept of looping **While loop** Syntax:-**/** While statement :**/**Statements () ex: count = 0**/** while (count < 3):**/** count = count + 1**/** print("Hello Geek") **For loop/** Syntax:-**/** for statement :**/** Statements() ex: n **=** 4**/** **for** i **in** range(0, n):**/** print(i) **While loop with** **break** statement we can stop the loop even if the while condition is true ex: i = 1**/** while i < 6:**/** print(i)**/** if i == 3:**/** break**/** i += 1 **function it also known as method** is a piece code that performs a specific task**/** It help to make a program organized and manageable**/** It help to reusable code that is used to perform a single, related action**/** In python it is very easy to define the function. Use def () Keywords**/** SYNTAX:-**/** def function\_name () :**/** ...statement. **/** Return or print () **Arguments** are the values passed inside the parentheses of the functions calling**/** A function can have any number of arguments separated by commas**/** Types of Arguments:**/** Default Argument**/** Keyword Argument**/** Variable Length Argument **Parameter** is separated from the next by comma**/** It pass the argument to the function**/** Parameters allow a function to perform tasks without knowing the specific input values ahead of time**/** Parameters are indispensable components of functions, which programmers use to divide their code into logical blocks. **loop with break or pass with examples** the break and pass statements are used within loops to control their flow, but they have different purposes**/** The break statement is used to exit (break out of) the current loop prematurely when a certain condition is met**/** It is commonly used in for and while loops to terminate the loop early if a specific condition is satisfied/ example:-**/** num = 0**/** for i in range(10):**/** num += 1**/** if num == 8:**/** break**/** print(num)**/** print("break")**/** The pass statement is a no-operation statement. It does nothing and is often used as a placeholder to maintain syntactic correctness when no action is required**/** It is commonly used when you're writing code structure and intend to implement the logic later**/** example:-**/** for i in range (50) :**/** if i % 2 == 0:**/** pass**/** else:**/** print (f"Odd number: {i}”) bulb\_on = False**/** fan\_on = False**/** while True:**/** print ("Choose an option:")**/** print ("1. Turn the bulb on")**/** print ("2. Turn the bulb off")**/** print ("3. Turn the fan on")**/** print ("4. Turn the fan off")**/** print ("5. Quit")**/** choice = input("Enter your choice (1/2/3/4/5): ")**/** if choice == '1':**/** if not bulb\_on:**/** bulb\_on = True**/** print("Bulb is now ON.")**/** else:**/** print("Bulb is already ON.")**/** elif choice == '2':**/** if bulb\_on:**/** bulb\_on = False**/** print("Bulb is now OFF.")**/** else:**/** print("Bulb is already OFF.")**/** elif choice == '3':**/** if not fan\_on:**/** fan\_on = True**/** print("Fan is now ON.")**/** else:**/** print("Fan is already ON.")**/** elif choice == '4':**/** if fan\_on:**/** fan\_on = False**/** print("Fan is now OFF.")**/** else**:/** print("Fan is already OFF.")**/** elif choice == '5':**/** print("Exiting the program.")**/** break**/** else:**/** print("Invalid choice. Please enter 1, 2, 3, 4, or 5.") **advantages of using loops over conditional statements**: Repetition and Automation: Loops are designed for repetitive tasks. They allow you to automate tasks that need to be performed multiple times without duplicating code**/** Handling Collections: Loops are particularly useful when working with collections of data (e.g., lists, arrays)**/** Dynamic Data: Loops are flexible and can handle dynamic data sets. You can iterate through data structures of varying lengths without needing to know the exact number of iterations in advance Complex Logic: When you need to execute complex logic repeatedly, loops are well-suited. You can encapsulate the logic within the loop, making it easier to read and maintain**/** Modularity: Loops encourage modular code design. You can place a set of instructions inside a loop and call that loop whenever you need to perform the same operation, enhancing code organization**/** Generalization: Loops can be used for a wide range of tasks, including iterating over data, implementing numerical algorithms, handling user input, and more.