**Introduction to Python Theory:**

**Q1) Introduction to Python and its Features (simple, high-level, interpreted language).**

ANS) **Introduction:**

* Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
* Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development.
* Python supports modules and packages, which encourages program modularity and code reuse.
* The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Features of Python:**

* Clean syntax plus high-level data types:
* Leads to fast coding (First language in many universities abroad!)
* Uses white-space to delimitblocks:
* Humans generally do, so why not the language?
* Try it, you will end up liking it
* Uses white-space to delimit blocks:
* Variables do not need declaration.
* Although not a type-less language.

**Q2) History and evolution of Python.**

**ANS)History:**

* In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) which is situated in the Netherlands. It was started as a hobby project because he was looking for an interesting project to keep him occupied during Christmas.

**Evolution:**

* The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with [Java](https://www.geeksforgeeks.org/java), [C++](https://www.geeksforgeeks.org/c-plus-plus)& [C](https://www.geeksforgeeks.org/c-programming-language). Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released, it had more than enough capability to provide classes with inheritance, several core data types of exception handling and functions.

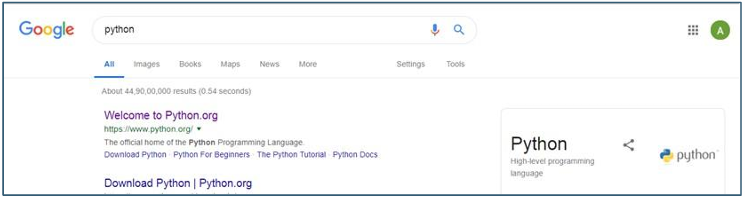
**Q3)** **Advantages of using Python over other programming languages.**

**ANS)Advantages:**

* **Versatile, easy to read, learn, and write:** Python is known for its simplicity and readability, making it an excellent choice for both beginners and experienced programmers.
* **Open source and large active community base:** Python is open source, and it has a large and active community that contributes to its development and provides support.
* **High-level language:** Python is a high-level language that abstracts low-level details, making it more user-friendly.
* **Extensive support libraries:** Python boasts extensive support libraries like NumPy for numerical calculations and Pandas for data analytics, making it suitable for scientific and data-related applications.
* **Object-Oriented and Procedural programming language:** Python supports both object-oriented and procedural programming, providing versatility in coding styles.
* **Dynamically typed language:** Python is dynamically typed, meaning you don't need to declare data types explicitly, making it flexible but still reliable.
* **Interpreted language:** Python is interpreted, which allows for easier debugging and code development.

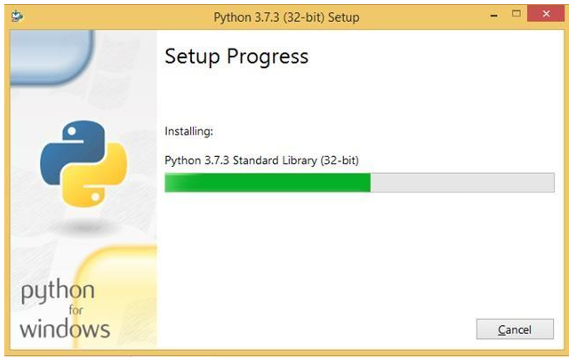
**Q4)** **Installing Python and setting up the development environment (Anaconda, PyCharm, or VS Code)**.

**ANS) Step 1:**

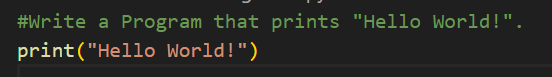
****

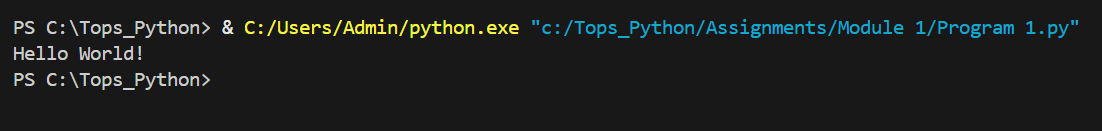
**STEP 2:**

****

**STEP 3: **

**Q5)** **Writing and executing your first Python program.**

****

****

**Q6)** **Understanding Python’s PEP 8 guidelines.**

**ANS) PEP 8 Guidelines:**

1. **Use docstrings :**There are both single and multi-line docstrings that can be used in Python. However, the single line comment fits in one line, triple quotes are used in both cases.
2. **Use of regular and updated comments are valuable to both the coders and users** : There are also various types and conditions that if followed can be of great help from programs and users point of view. Comments should form complete sentences.
3. **Use of trailing commas :** This is not mandatory except while making a tuple.
4. **Use spaces around operators and after commas, but not directly inside bracketing constructs.**
5. **Naming Conventions :** There are few naming conventions that should be followed in order to make the program less complex and more readable.
6. **Characters that should not be used for identifiers :**'l' (lowercase letter el), 'O' (uppercase letter oh), or 'I' (uppercase letter eye) as single character variable names as these are similar to the numerals one and zero.
7. **Name your classes and functions consistently:**The convention is to use **CamelCase** for classes and **lower\_case\_with\_underscores** for functions and methods.
8. **While naming of function of methods always use self for the first argument.**

**Q7)** **Indentation, comments, and naming conventions in Python.**

**ANS) Indentation:**

* Whitespace is used for **indentation in Python**. Unlike many other programming languages which only serve to make the code easier to read, **Python indentation** is mandatory.

**Comments:**

* Python comments start with the hash symbol # and continue to the end of the line. [Comments in Python](https://www.geeksforgeeks.org/python-comments/) are useful information that the developers provide to make the reader understand the source code.
* Types of comments in Python:

1. Single Line Comments
2. Multiple Line Comments

**Naming Conventions**:

* Naming conventions in Python refer to rules and guidelines for naming variables, functions, classes, and other entities in your code. Adhering to these conventions ensures consistency, readability, and better collaboration among developers.
* Naming Conventions in python are:

1. Modules
2. Variables
3. Classes
4. Exceptions

**Q8)** **Writing readable and maintainable code.**

**ANS) Guidelines:**

1. Consistent Naming Conventions
2. Avoid Long Functions
3. Comment Wisely
4. Use Proper Indentation
5. Modular Design

**Q9)** **Understanding data types: integers, floats, strings, lists, tuples, dictionaries, sets.**

**ANS) Integers:**

* This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.

**Float:**

* This value is represented by the float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.

**String:**

* Python Strings are arrays of bytes representing Unicode characters. In Python, there is no character data type Python, a character is a string of length one. It is represented by str class.
* Strings in Python can be created using single quotes, double quotes or even triple quotes.
* We can access individual characters of a String using index.

**Lists:**

* Lists are just like arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

**Tuples**:

* Just like a list, a tuple is also an ordered collection of Python objects. The only difference between a tuple and a list is that tuples are immutable. Tuples cannot be modified after it is created.

**Dictionaries:**

* A dictionary in Python is a collection of data values, used to store data values like a map, unlike other Python Data Types that hold only a single value as an element, a Dictionary holds a key: value pair. Key-value is provided in the dictionary to make it more optimized.
* Each key-value pair in a Dictionary is separated by a colon : whereas each key is separated by a ‘comma’.

**Sets:**

* In Python Data Types, Set is an unordered collection of data types that is iterable, mutable, and has no duplicate elements.

**Q10)** **Python variables and memory allocation.**

**ANS) There are two parts of memory:**

1. Stack Memory
2. Heap Memory

* The methods/method calls and the references are stored in stack memory and all the values objects are stored in a private heap.

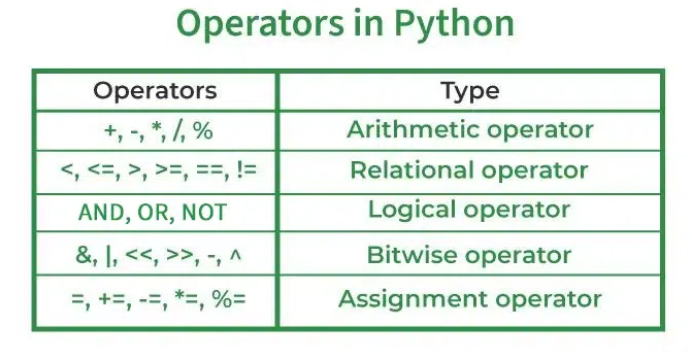
**Variables:**

Python variables are simply containers for storing data values. Unlike other languages, such as Java, Python has no command for declaring a variable, so you create one the moment you first assign a value to it. Python variables are simply containers for storing data values.

**Q11) Python operators: arithmetic, comparison, logical, bitwise.**

**ANS)** In Python programming, Operators in general are used to perform operations on values and variables.

* **OPERATORS:**These are the special symbols. E.g. - + , \* , /, etc.



**Q12)** **Introduction to conditional statements: if, else, elif.**

**ANS) IF:**

* If the simple code of block is to be performed if the condition holds true then if statement is used**.**
* Syntax:

*If condition:*

*# Statements to execute if condition is true*

**IF-ELSE:**

* In conditional if Statement the additional block of code is merged as else statement which is performed when if condition is false.
* Syntax:

*If (condition):*

*# executes this block if condition is true*

*Else:*

*# executes this block if condition is false*

**IF-ELIF:**

* The if-elif statement is shortcut of if..else chain. While using if-elif statement at the end else block is added which is performed if none of the above if-elif statement is true.
* Syntax:

*If (condition):*

*Statement*

*Elif (condition):*

*Statement*

*Else:*

*Statement*

**Q13) Nested if-else conditions**.

**ANS)** if statement can also be checked inside other if statement. This conditional statement is called a nested if statement. This means that inner if condition will be checked only if outer if condition is true and by this, we can see multiple conditions to be satisfied.

* Syntax:

*If (condition1):*

*# executes when condition1 is true*

*If (condition2):*

*# executes when condition2 is true*

**Q14) Introduction to for and while loops.**

**ANS) While Loop:**

* In Python, a while loop is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.
* Syntax:

While expression:

Statement(s)

**For Loop:**

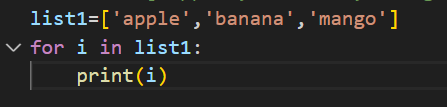
* For loops are used for sequential traversal. For example: traversing a list or string or array,etc.
* Syntax :

for iterator\_var in sequence:

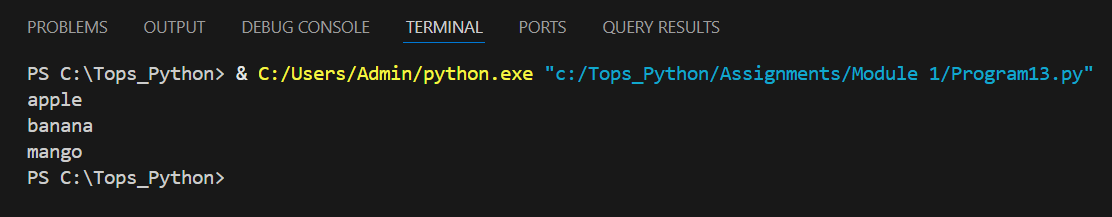
statements(s)

**Q15) How loops work in Python.**

**ANS)**

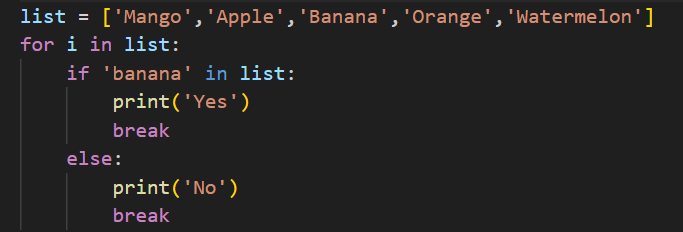


**Output:**

****

**Q16) Using loops with collections (lists, tuples, etc.).**

**ANS)**

****

**OUTPUT:**

****

**Q17) Understanding how generators work in Python.**

**ANS)** Instead of using return to send back a single value, generator functions use yield to produce a series of results over time. This allows the function to generate values and pause its execution after each yield, maintaining its state between iterations.

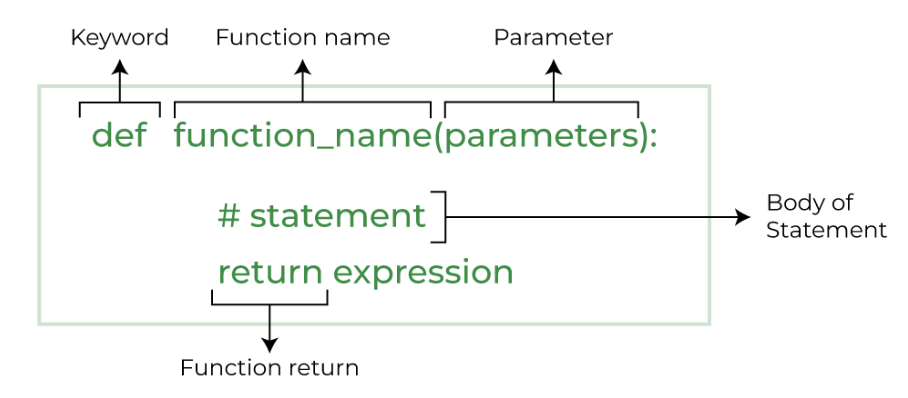
**Q18) Difference between yield and return.**

**ANS)**

**Q19) Understanding iterators and creating custom iterators.**

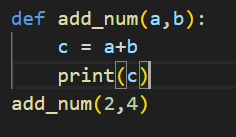
**Q20) Defining and calling functions in Python.**

**ANS)** Python Functions is a block of statements that does a specific task. The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

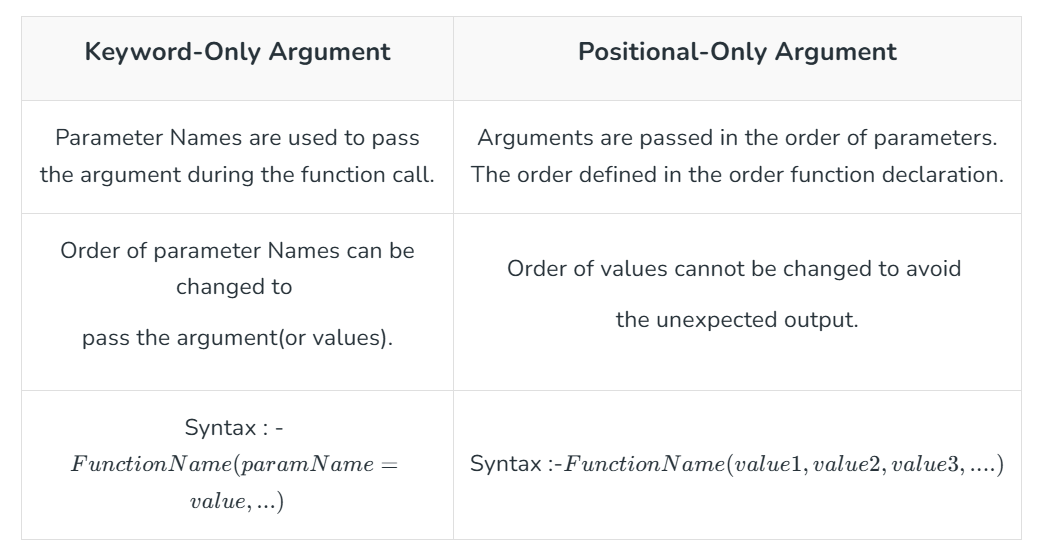


There are 2 types of functions in python :

1. Built-In Functions: These are Standard functions in Python that are available to use.
2. User-Defined Funcions : We can create our own functions based on our requirements.



**Q21) Function arguments (positional, keyword, default).**

**ANS)** 

**Default Arguments**:

Default values indicate that the function argument will take that value if no argument value is passed during the function call. The default value is assigned by using the assignment(=) operator of the form keyword name = value.

Syntax:

def function\_name(param1, param2=default\_value2, param3=default\_value3)

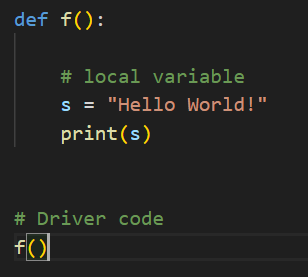
**Q22) Scope of variables in Python.**

**ANS**) The scope of a variable determines where it can be accessed.

The location where we can find a variable and also access it if required is called the scope of a variable.

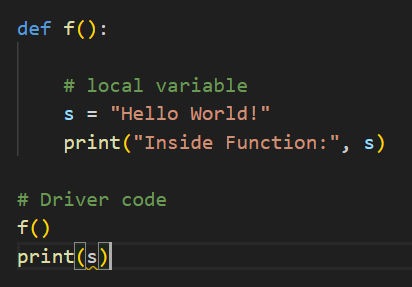
**Local Variables:**

* Local variables are those that are initialized within a function and are unique to that function. It cannot be accessed outside of the function.



**Global Variables:**

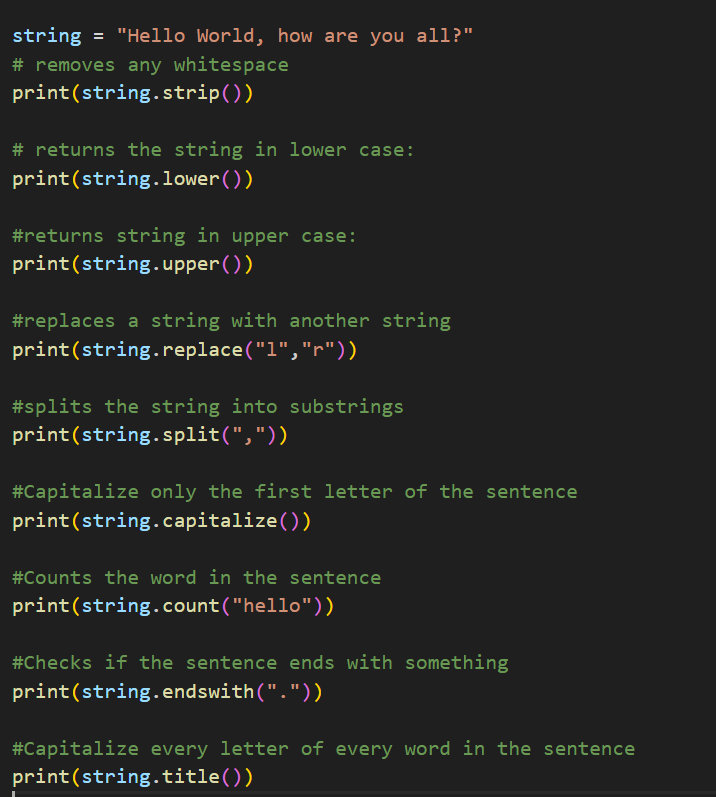
* Global variables are the ones that are defined and declared outside any function and are not specified to any function. They can be used by any part of the program.



**Q23) Built-in methods for strings, lists, etc.**

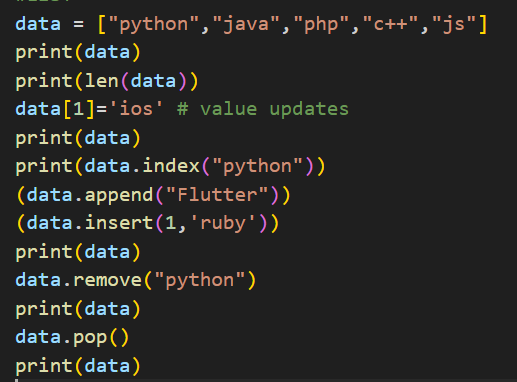
**ANS) String Methods are :**

1. Lower()
2. Upper()
3. Capitalize()
4. Title()
5. Strip()
6. Replace()
7. Endswith()
8. Split()
9. Isalpha()
10. Isdigit()
11. Isalnum()



**List Methods are:**

1. Append(x)
2. Extend(iterable)
3. Insert(i,x)
4. Remove(x)
5. Pop([i])
6. Clear()
7. Index(x)
8. Reverse()



**Q24) Understanding the role of break, continue, and pass in Python loops.**

**ANS) Break Statement:**

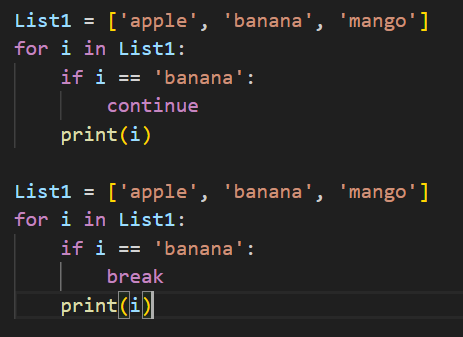
* The break statement in Python is used to exit or “break” out of a loop (either a for or while loop) prematurely, before the loop has iterated through all its items or reached its condition. When the break statement is executed, the program immediately exits the loop, and the control moves to the next line of code after the loop.

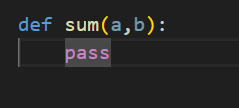
**Continue Statement:**

* Python Continue statement is a loop control statement that forces to execute the next iteration of the loop while skipping the rest of the code inside the loop for the current iteration only, i.e. when the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped for the current iteration and the next iteration of the loop will begin.

**Pass Statement:**

* Pass statement in Python is a null operation or a placeholder. It is used when a statement is syntactically required but we don’t want to execute any code. It does nothing but allows us to maintain the structure of our program.

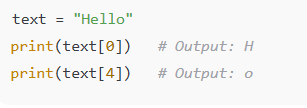




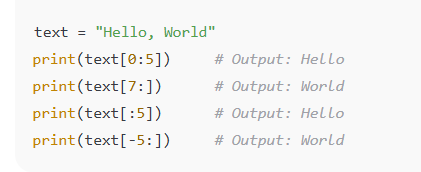
**Q25)** **Understanding how to access and manipulate strings.**

**ANS)**

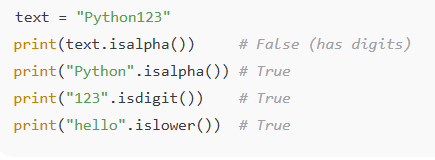
1. Strings are sequences of characters, and you can access each character using indexing:



1. Slicing Strings: You can get a substring using slicing:



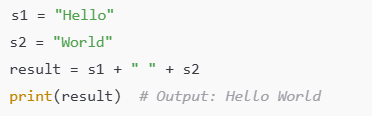
1. Common String Methods(Manipulations):



**Q26) Basic operations: concatenation, repetition, string methods (upper(), lower(), etc.).**

**ANS) Basic Operations:**

1. **Concatenation(+):**Joining two or more strings together.



1. **Repitition(\*):**Repeating a string multiple times.



1. **String Methods:**

* **Upper():**Converts all characters to UPPERCASE

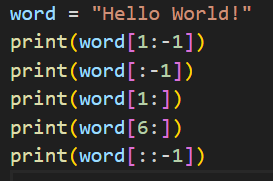


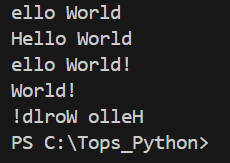
* **Lower():**Converts all characters to lowercase**.**



**Q27) String slicing.**

**ANS)** String slicing means extracting a part (or slice) of a string using indexes.





**Q28) How functional programming works in Python.**

**ANS)** Functional programming is a programming paradigm in which we try to bind everything in a pure mathematical functions style. It is a declarative type of programming style. Its main focus is on “what to solve" in contrast to an imperative style where the main focus is "how to solve".

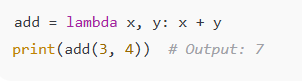
* **First-Class Functions:**

In Python, functions can be assigned to variables, passed as arguments, or returned from other functions.



* **Lambda Functions:**

Useful for short, one-line functions and also known as Anonymous Function.



**Q29) Using map(), reduce(), and filter() functions for processing data.**

**ANS)**

**Q30)** **Introduction to closures and decorators.**

**ANS)**