MODULE – 6

(PYTHON FUNDAMENTALS)

Name : Megha Patel

1>>> INTRODUCTION TO PYTHON :

1> INTRODUCTION TO PYTHON AND ITS FEATURES.

→ Python is a popular, high-level, open-source programming language with many features, including:

- **Easy to use**: Python uses English-like words and indentation instead of curly brackets.
- **Dynamically typed**: Python determines variable types at runtime, so programmers don't need to declare them.
- **Interpreted**: Python runs code line by line, so errors are easy to find.
- **Cross-platform**: Python works on Windows, macOS, Linux, and Raspberry Pi.
- **Versatile**: Python supports multiple programming paradigms, including object-oriented, structured, and functional programming.
- **Rich libraries**: Python has extensive libraries for web development, data science, and AI.
- **Scalable**: Python supports large programs and can be used as a scripting language.

- **Rapid prototyping**: Python's quick syntax allows for fast idea implementation.
- **Debugging**: Python's edit-test-debug cycle is fast because there's no compilation step.
- **Modules and packages**: Python supports modules and packages, which encourages code reuse.

2> HISTORY AND EVOLUTION OF PYTHON.

→ Python was created by Guido van Rossum, a Dutch programmer working for the Centrum Wiskunde & Informatica (CWI) in the Netherlands, as a hobby project to keep himself busy during the Christmas holidays of 1989. His aim was to develop a language that emphasized code readability and simplicity.

3> ADVANTAGES OF USING PYTHON OVER OTHER PROGRAMMING LANGUAGES.

→ Python has several advantages over other programming languages, including:

- **Easy to learn**: Python has a simple syntax and is easy to read, making it accessible to beginners.
- **Large standard library**: Python's standard library includes many common programming tasks, which reduces the amount of code that needs to be written.
- **Open source**: Python is free to use and distribute, and is developed by a community that contributes code through mailing lists and conferences.

- **Integrates with other languages**: Python can integrate with other languages, such as Java, C, and C++, through libraries like Jython and Cython.
- **Extensive libraries**: Python has a rich ecosystem of libraries for many tasks, including data analysis, scientific computing, and machine learning.
- **Platform-independent**: Python is platform-independent.
- **Strong typing**: Python's strong typing can lead to more maintainable codebases.
- **Code readability**: Python emphasizes code readability and allows you to use English keywords without punctuation.
- **Community support**: Python has a large and supportive community.

2>>> PROGRAMMING STYLE :

- **Indentation**: Use four spaces for each indentation level, and avoid using tabs.
- **Line length**: Limit lines to 79 characters for code, and 72 characters for comments and docstrings.
- **Imports**: Import one module per line, and order them alphabetically.
- **Whitespace**: Use spaces around operators and operands to improve clarity. Also use whitespace around parentheses, brackets, and braces.

- **Block comments**: Indent block comments to the same level as the code they describe. Start each line with a # followed by a single space. Separate paragraphs with a line containing a single #.
- **Naming conventions**: Use lowercase letters and underscores to separate words for variable and function names. This is known as snake case. For class names, use CamelCase, which means capitalizing each new word without an underscore. For constants, use all capital letters and underscores to separate words.
- **Blank lines**: Use blank lines to separate functions, classes, and other logical blocks.

2> INDENTATION,COMMENTSAND NAMING CONVENTION IN PYTHON.

→ In Python, indentation, comments, and naming conventions are important for writing clean and readable code:

- **Indentation**: Use spaces or tabs to indent each line of code within a block by the same number of spaces. The standard is four spaces, but you can choose your own width. However, you should be consistent throughout your codebase.
    - **Comments**: Use comments to explain the code. You can write comments on a single line, next to the corresponding line of code, or in a block of multiple lines.
    - **Naming conventions**: Use different naming conventions for different types of variables, classes, methods, and constants:

    3>WRITING READABLE AND MAINTAINABLE CODE.

3>>>> CORE PYTHON CONCEPT .

1> UNDERSTANDING DATATYPES : INTEGERS ,FLOATS,STRINGS,LISTS,TUPLES,DICTIONARIES ,SETS.

→ The data types in Python that we will look at in this tutorial are integers, floats, Boolean, and strings. If you are not comfortable with using variables in Python, our article on how to declare python variables can change that.

Python Lists

•

In Python, a **list** is a built-in dynamic sized array (automatically grows and shrinks) that is used to store an ordered collection of items. We can store all types of items (including another list) in a list. A list may contain mixed type of items, this is possible because a list mainly stores references at contiguous locations and actual items maybe stored at different locations.

In Python, a tuple is a built-in data type that stores a collection of elements in an ordered and immutable way:

- **Ordered**: The elements in a tuple are ordered, with the first item having an index of 0, the second item having an index of 1, and so on.
- **Immutable**: Once created, the elements in a tuple cannot be modified.
- **Can contain different data types**: Tuples can contain elements of different data types, such as integers, strings, floats, or even other tuples.

- **Defined by parentheses**: Tuples are defined by having values between parentheses ().

The statistics. mean() method calculates the mean (average) of the given data set. Tip: Mean = add up all the given values, then divide by how many values there are.

2> PYTHON VARIABLE AND MEMORY ALLOCATION .

→ 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.

3> EXPLAIN PYTHON OPRATORS :

ARITHMATIC
COMPARISON
LOGICAL
BITWISE

→ An arithmetic operator is a symbol that performs a mathematical operation on one or more operands. The basic arithmetic operators in Python include addition (+), subtraction (-), multiplication (*), division (/), floor division (//), modulus (%), and exponentiation (**).

Comparison operators, also known as relational operators, in Python compare two values and return either True or False. They

are used to determine the relationships between variables and create conditional statements.
Here are some comparison operators in Python:

- **== or equal to**: Checks if two values are equal
- **!= or not equal to**: Checks if two values are not equal
- **> or greater than**: Checks if the value on the left side of the operator is greater than the value on the right side
- **< or less than**: Checks if the value on the left side of the operator is less than the value on the right side
- **>= or greater than or equal to**: Checks if the left operand is greater than or equal to the right
- **<= or less than or equal to**: Checks if the left operand is less than or equal to the right

Python's logical operators are used to evaluate conditions in conditional statements and return Boolean values:

- **AND**: Returns True if both operands are True, and False otherwise
- **OR**: Returns True if at least one operand is True,
  and False otherwise
- **NOT**: Returns the inverse Boolean value of the operand

Python bitwise operators are used to perform bitwise calculations on integers. The integers are first converted into binary and then operations are performed on each bit or corresponding pair of bits, hence the name bitwise operators. The result is then returned in decimal format.

4>>> CONDITIONAL STATEMENTS :

1> INTRO OF CONDITIONAL STATEMENTS :

- IF
- ELSE
- ELIF

IF :

An if statement is a condition statement used to check a condition, and execute it if the condition holds true. It is also a control flow statement, which utilizes decision-making to control the flow of execution.

ELSE:

In Python, the for-else loop is a construct that combines a for loop with an else clause. The loop body typically checks for a condition. If the condition is True , the control breaks out of the loop. The else block will execute only when the for loop completes normally without encountering a break statement.

ELIF :

The “elif” keyword in Python, stands for “else if”. It can be used in conditional statements to check for multiple conditions. For example, if the first condition is false, it moves on to the next “elif” statement to check if that condition is true.

2>> EXPLAIN NESTED IF CONDITION .

→ A nested if condition in Python is an if statement that is placed inside another if statement. This allows for the evaluation of multiple conditions in a hierarchical manner.

Here are some things to know about nested if conditions in Python:

- They can be used to make complex decisions in a script.
- They can be nested inside each other in any order.
- The depth of nesting can be determined by using indentation.
- The inner if condition will only be checked if the outer if condition is true.
- They can be perplexing, so it is recommended to avoid them unless absolutely required.

5>>> LOOPING :

1> INTRODUCTION TO FOR AND WHILE LOOP .

→ For and while loops are both control flow statements that repeat a block of code, but they are used in different situations:

- **For loop**

Used when you know how many times you want to iterate over a sequence, like a list, tuple, or string.

- **While loop**

Used when you don't know how many times you want to iterate, or when you want to repeat a block of code until a condition is true.

2> HOW LOOPS WORK ON P0YTHON.

→ A for loop in Python is used to iterate over a sequence (list, tuple, set, dictionary, and string). Example: The preceding code executes as follows: The variable i is a placeholder for every item in your iterable object. The loop iterates as many times as the number of elements and prints the elements serially.

3> USING LOOP WITH COLLECTIONS.

→ Looping with for. A for-loop in Python executes a block of code, once for each element of an iterable data type: one which can be accessed one element at a time, in order. As it turns out, both strings and lists are such iterable types in Python, though for now we'll explore only iterating over lists with for-loops.

6>>> GENERATORS AND ITERATORS :

1> UNDERSTANDING HOW GENERATORS WORK IN PYTHYON.

→ Python generator functions allow you to declare a function that behaves like an iterator, making it a faster, cleaner and easier way to create an iterator. An iterator is an object that can be iterated or looped upon. It is used to abstract a container of data to make it behave like an iterable object.

2> DIFFERENCE BETWEEN YIELD AND RETURN.

→ Yield measures the income generated by an investment relative to its price, typically expressed as a percentage and focusing on regular payments like interest or dividends. Return encompasses the total gain or loss from an investment, including both income and changes in its price.

3> UNDERSTANDING ITERATORS AND CREATING CUSTOM ITERATORS.

→ Iterators are objects that allow users to access the elements in a container or sequence in a sequential manner. They are important for Python developers, especially when working with large-scale or complex data. Custom iterators can be created to provide scalable and efficient data processing, and to increase the capabilities of Python's built-in data structures.

- **Custom iterators**

To use custom iterators, create an Apex class that implements the Iterator interface. All methods in the Iterator interface must be declared as global or public.

7>>>> FUNCTIONS AND METHODS :

1> DEFINING AND CALLING FUNCTIONS IN PYTHON.

→ Creating a Function in Python. When declaring a function in Python, the 'def' keyword must come first, then the function name, any parameters in parenthesis, and then a colon. The code that needs to be run is indented in the function body. The 'return' statement is optional for a function to return a value.

2>>FUNCTION ARGUMENTS (POSITIONAL , KEYWORD ,DEFALUT)

→ n Python, a function argument is a value passed to a function during a function call. Arguments are inputs that tell functions what to output, and they allow functions to act in different ways and use different data.

Here are some types of function arguments in Python:

- **Default arguments**

A parameter that assumes a default value if a value is not provided in the function call for that argument.

- **Keyword arguments**

Values that are identifiable by specific parameter names. Keyword arguments are preceded by a parameter and the assignment operator, =.

- **Positional arguments**

Arguments that need to be included in the proper position or order. The first positional argument always needs to be listed first when the function is called.

2> SCOPE OF VARIABLE IN PYTHON.

→ Scope of a variable is the region in the code where the variable is available/accessible. A variable declared outside a function (i.e. the main region of the code) is called a global variable and a variable declared inside a function is called a local variable of that function

3> BUILT-IN METHODS FOR STRING ,LISTS ETC.

→Capitalize

Casefold()

Count

Encode()

Expandtabs()

Find

Format()

Format_map()

Index

Isalnum

Isalpha

Isdecimal()

Isidentifier()

Islower()

Isnumeric()

Isprintable()

8>>> CONTROL STATEMENTS. :

1> UDERSTANDING THE ROLE OF BREAK ,COUNTINUE , AND PASS.

- In Python, the keywords "break" and "continue" are used to escape a loop early, "pass" is a placeholder statement that does

nothing, and these words are frequently used when a statement is syntactically necessary but no action is required.

9>>> STRING MANIPULATION :

1> UNDERSTANDING HOW TO ACCESS AND MANIPUKATE STRINGS.

- → Python list elements are ordered by index, a number referring to their placement in the list. List indices start at 0 and increment by one. To access a list element by index, square bracket notation is used: list[index] .

2> BASIC OPERATIONS :
CONCATENATION
REPETITION
STRING METHODS
STRING SLICING

Concatenating means obtaining a new string that contains both of the original strings. In Python, there are a few ways to concatenate or combine strings. The new string that is created is referred to as a string object

This technique is known as string repetition or string replication. The multiplication operator (*) is used to repeat a string a specified number of times. string = "hello " result = string * 3 print(result) # output: hello hello hello # string is assigned the value `hello`.

- **len()**: Returns the total number of characters in a string
- **upper()**: Converts all characters in a string to uppercase
- **lower()**: Converts all characters in a string to lowercase
- **strip()**: Removes leading and trailing whitespace from a string
- **replace(old, new)**: Replaces all occurrences of a specified substring with another
- **capitalize()**: Returns a new string where only the first character is in uppercase and all the remaining characters are in lowercase
- **join()**: Returns a string by joining all items in an iterable together
- **rstrip()**: Deletes all the trailing characters mentioned in its argument
- **split()**: Splits a string into a list of substrings from the right end of the string based on a specified delimiter
- **startswith()**: Returns true if the string starts with the specified value
- **swapcase()**: Swaps cases, lower case becomes upper case and vice versa
- **title()**: Converts the first character of each word to upper case
- **find()**: Searches for the position of one string within another

10>>> ADVANCED PYTHON :

1> HOW TO FUNCTIONAL PROGRAMMIBNG WORKS IN PYTHON .

→ Functional programming in Python is a coding style that uses functions to structure code and avoid changing state:

- **Functions are first-class citizens**

  Functions can be passed as arguments to other functions, returned by other functions, and stored and manipulated.

- **Avoids state changes**

  Functional programming minimizes side effects by returning new copies of data instead of modifying existing data.

- **Data transformations**

  Instead of moving through steps, data is transformed with the desired result as the end state.

- **Immutability**

  Tuples, frozensets, and strings are examples of immutable types in Python, which means they cannot be changed once created.

- **Recursion**

  Iteration is implemented through recursion, where recursive functions repeatedly call themselves until they reach a base case.

2> USING MAP ,REDUUCE AND FILTER FUNCTION FOR PROCESSING DATA.

3> INTRODUCTION TO CLOSURES AND DECORATORS.

- **Closures**

A closure is a function object that remembers the values from its enclosing scope, even after the enclosing function has finished executing. Closures are created by defining a function inside another function and returning the inner function.

- **Decorators**
  A decorator is a higher-order function that takes another function as an argument, adds functionality to it, and returns a new function. Decorators can be used to modify or extend the behavior of functions and methods.