PYTHON PROGRAMMING

TASK-2
ASSIGNMENT 1
(WEEK-1)

-S ROHITH ADITHYA

INTRODUCTION TO PYTHON

- Python is a high-level, versatile programming language known for its simplicity and readability. It was created by Guido van Rossum in the late 1980s.
- Python is an interpreted language, which means that code is executed line by line. This makes it easier for developers to test and debug their programs.
- Python supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural programming, providing flexibility to developers.
- Python has a clean and easy –to-understand syntax, making it a great choice for beginners and experienced programmers.
- Python has an extensive standard library that provides a wide range of built-in functions and modules for various tasks, such as file I/O, networking, and data manipulation.
- Python has a large and active community of developers who contribute to its ecosystem by creating and maintaining libraries and frameworks for different application.
- Python is widely used in web development, data analysis, artificial intelligence, scientific computing, automation, and more, showcasing its versatility.
- With its powerful features, readability, and comprehensive documentation, python is considered an excellent choice for a wide range of programming tasks, making it one of the most popular languages in the world.

HISTORY OF PYTHON

- Python was created by Guido van Rossum in the late 1980s while working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands. The first version, Python 0.9.0, was released in February 1991.
- The name "Python" was inspired by the British comedy show "Monty Python's Flying Circus," of which Guido van Rossum was a fan. This is why Python has quirky language features and references to Monty Python throughout its documentation.
- Python was designed to be a simple and readable language, with a focus on code readability and ease of use. This philosophy is captured in the Zen of Python, a collection of aphorisms that describe the guiding principles of Python development.
- Python 2.0, released in 2000, introduced new features such as list comprehensions and garbage collection, solidifying Python's position as a powerful and versatile language for development.
- In 2008, Python 3.0 (also known as Python 3000 or Py3k) was released, introducing backward-incompatible changes to improve the language's consistency and fix design flaws. This led to a gradual adoption of Python 3 within the community.

FUNCTIONS IN PYTHON

- Defining a Function: In Python, you can define a function using the def keyword followed by the function name and parentheses containing any parameters.
- Function Arguments: Functions in Python can have zero or more arguments. Arguments are specified within the parentheses when defining the function.
- Return Statement: Functions can return a value using the return statement. The returned value can be used in other parts of the program.
- Default Arguments: You can provide default values for function parameters. If an argument is not provided when calling the function, it takes the default value.
- Keyword Arguments: When calling a function, you can specify arguments by parameter name, which allows you to provide the arguments in any order.
- Arbitrary Arguments: You can pass a variable number of arguments to a function by using the *args or **kwargs notation in the function definition. *args collects positional arguments into a tuple, while **kwargs collects keyword arguments into a dictionary.
- Lambda Functions: In Python, you can create anonymous functions using the lambda keyword. Lambda functions are useful for creating quick, short functions without explicitly defining a function name.
- Global vs. Local Variables: Variables defined inside a function are considered local to that function, while variables defined outside the function are considered global. To modify a global variable inside a function, you need to use the global keyword.

MODULES IN PYTHON

- In Python, a module is a file containing Python code. It can define functions, classes, and variables, and can be imported and used in other Python files to reuse code.
- Importing Modules: To use a module in Python, you need to import it using the import statement.
- Using Functions from Modules: Once a module is imported, you can access its functions, classes, and variables using dot notation.
- Aliasing Modules: Modules can be imported with an alias to make their names shorter and more convenient to use.
- Importing Specific Items: You can import specific items (functions, classes, or variables) from a module rather than the whole module.
- You can create your own modules by defining functions, classes, and variables in a Python file. To use these custom modules in other files, you just need to import them.
- The __name__ Attribute: Each module in Python has a built-in attribute called __name__. When a Python script is executed as the main program, __name__ is set to "__main__". This allows you to check whether the module is being imported or run directly.
- Standard Library Modules: Python's standard library is a collection of modules that come with Python installation. These modules provide a wide range of functionalities that can be used without any additional installations, covering areas like file I/O, networking, mathematics, and more.

THANK YOU 😊