MD. ARSHAD </p>

ALISHARMEEN02@GMAIL.COM </p>

1. Who developed Python Programming Language? ANSWER :- Python programming language was created by Guido van Rossum.

```
He started working on Python in the late 1980s,
and the first version, Python 0.9.0, was released in February 1991.
Guido van Rossum remained the project's lead developer and
"Benevolent Dictator For Life" (BDFL) until he stepped down from the role in July 2018.
 The language has since been developed and maintained by a community of volunteers.
```

1. Which type of Programming does Python support?

ANSWER :- Python supports a variety of programming paradigms, making it a versatile and flexible programming language. Some of the main programming paradigms that Python supports are:

1. Imperative Programming: Python allows you to write code that explicitly defines a sequence of steps to be executed. This includes statements and control structures like loops and conditionals.
2. Procedural Programming: Python supports procedural programming, where you can organize your code into reusable functions and procedures.
3. Object-Oriented Programming (OOP): Python is an object-oriented language, which means you can define classes and objects, encapsulate data and behavior within objects, and use inheritance and polymorphism.
4. Functional Programming: Python supports functional programming concepts, such as first-class functions, higher-order functions, and lambda expressions. You can treat functions as objects, pass them as arguments, and return them as results.
5. Dynamic Typing: Python is dynamically typed, meaning you don't need to declare the data type of a variable explicitly. The type of a variable is determined at runtime.
6. Dynamic Binding: Python supports dynamic binding, allowing you to change the behavior of functions or methods at runtime.
7. Scripting: Python is often used as a scripting language for automating tasks and writing quick, concise scripts.
8. Interpreted Language: Python code is executed by an interpreter, rather than being compiled into machine code. This enables a faster development cycle and easier debugging.
9. Modular Programming: Python supports modular programming through the use of modules and packages, allowing you to organize your code into reusable and maintainable components.
10. Extensible: Python can be extended with C/C++ code, enabling you to use existing libraries or create performance-critical modules.

    " The ability to support multiple programming paradigms makes Python suitable for a wide range of applications, from web development to scientific computing, data analysis, machine learning, and more."

1. Is Python case sensitive when dealing with identifiers?

ANSWER :- Yes, Python is case-sensitive when dealing with identifiers. This means that Python treats uppercase and lowercase letters as distinct characters. For example, variables or function names with different capitalization are considered to be different identifiers. Here's an example to illustrate this:

In [1]:
```
Arshad = 26
arshad = 39
print(Arshad)
print(arshad)
```

```
26
39
```

In the above code, arshad and Arshad are treated as two separate identifiers because Python is case-sensitive. It's important to consistently use the correct capitalization when referring to identifiers in your Python code.

1. What is the correct extension of the Python file?

ANSWER :- The correct extension for a Python source code file is .py. When you save your Python code in a file, you should give it a name and use the .py extension. For example, if you have a Python program named "my_program", you would save it as "my_program.py". This convention helps indicate that the file contains Python code and allows tools and systems to recognize and treat the file as Python source code.

1. Is Python code compiled or interpreted?

ANSWER :- Python code is both compiled and interpreted, but the process is a bit more nuanced than in some other programming languages.

1. Compilation: When you write Python code, it is first compiled into an intermediate form known as bytecode. This compilation is performed by the Python interpreter. The bytecode is a lower-level representation of your code that is not directly executed by the CPU. It's a platform-independent format that can be executed on any system that has a compatible Python interpreter.
2. Interpretation: After the compilation step, the Python interpreter reads and executes the bytecode line by line. This is the interpretation phase. The interpreter translates the bytecode into machine code that can be executed by the computer's CPU. The interpretation process happens in real-time as the code is being executed.

This combination of compilation and interpretation provides Python with a number of advantages, such as portability (since bytecode can be executed on different platforms) and ease of development (since you can run and test your code without a separate compilation step).

In summary, Python code is compiled into bytecode and then interpreted by the Python interpreter during runtime. This is often referred to as a "compiled" language with an "interpreted" runtime.

1. Name a few blocks of code used to define in Python language?]

ANSWER :- In Python, code blocks are defined by their indentation levels rather than explicit braces or keywords. Here are a few key blocks of code that are commonly used in Python:

1. Function Definitions: Functions in Python are defined using the def keyword, followed by the function name, parameters, and a colon. The indented block of code beneath the function definition is the function body.
2. Conditional Statements: Conditional statements such as if, elif, and else are used for controlling the flow of execution based on conditions. The indented blocks under each condition define the code to be executed.
3. Loops: Loops like for and while are used to repeatedly execute code blocks. The indented block under the loop defines the code to be repeated.
4. Class Definitions: Classes in Python are defined using the class keyword, followed by the class name and a colon. The indented block of code beneath the class definition defines the class's methods and attributes.
5. Exception Handling: Exception handling with try, except, finally, and raise allows you to handle errors gracefully. The indented block under try and except defines the code to handle exceptions. These are just a few examples of code blocks in Python. Proper indentation is crucial in Python to define and structure these code blocks correctly.

1. State a character used to give single-line comments in Python? ANSWER :- In Python, the # character is used to indicate a single-line comment. Any text following the # symbol on the same line is treated as a comment and is ignored by the Python interpreter. Single-line comments are used to provide explanations, notes, or annotations within the code.

For example:

In [2]:
```
# This is a single-line comment
print("Hello, World!")  # This is another comment
```

```
Hello, World!
```

In the above code, both lines that start with '#' are comments and will not be executed by the interpreter.

Thank You ,That's All </p>