Core Python Course

**Module 1: Introduction to Python**

**Slide 1: Overview**

* **Title:** Introduction to Python
* **Duration:** 2 hours
* **Objective:** To introduce Python programming language and its fundamentals.



**Slide 2: What is Python?**

* **Definition:**

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. Developed in the late 1980s by Guido van Rossum and first released in 1991, Python emphasizes code readability and expressive syntax, making it ideal for both beginners and experienced programmers alike. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming styles. Python's extensive standard library and large ecosystem of third-party packages contribute to its widespread adoption in various fields such as web development, data science, artificial intelligence, data engineering, scientific computing, data analyst and more.

* **Key Points:**

**Versatility**: Python is a versatile programming language widely used in various domains such as web development, data analysis, artificial intelligence, scientific computing, and more. Its versatility makes it a preferred choice for diverse projects and applications.

**Ease of Learning**: Python's simple and straightforward syntax makes it easy for beginners to grasp programming concepts. Its readability and simplicity contribute to a shorter learning curve compared to other programming languages, allowing newcomers to quickly start writing useful code.

**Example Script**: Here's a simple Python script that prints "Hello, World!":

print("Hello, World!")

**Slide 3: Why Python?**

1. **Ease of Learning**: Python's simple and readable syntax makes it an ideal choice for beginners. Its straightforward syntax resembles English-like language, reducing the learning curve and allowing developers to focus more on problem-solving rather than syntax intricacies.
2. **Versatility**: Python is incredibly versatile, with applications spanning across various domains such as web development, data analysis, artificial intelligence, scientific computing, automation, and more. Its versatility makes it a go-to language for tackling a wide range of projects and tasks.
3. **Large Standard Library**: Python comes with an extensive standard library that provides ready-to-use modules and functions for performing tasks such as file I/O, networking, data manipulation, and more. This vast collection of modules accelerates development by eliminating the need to build everything from scratch.
4. **Rich Ecosystem of Libraries and Frameworks**: Python boasts a thriving ecosystem of third-party libraries and frameworks contributed by a vast community of developers. These libraries and frameworks cater to specific needs and domains, enabling developers to leverage pre-built solutions for rapid development.
5. **Community Support**: Python enjoys robust community support with active forums, mailing lists, and online communities. Developers can seek help, share knowledge, and collaborate with peers, making problem-solving and learning more accessible and efficient.
6. **Cross-Platform Compatibility**: Python is platform-independent, meaning code written in Python can run seamlessly on various operating systems such as Windows, macOS, Linux, and others without requiring modifications. This cross-platform compatibility simplifies deployment and ensures consistent behavior across different environments.
7. **Scalability**: Python's scalability allows it to accommodate projects of all sizes, from small scripts to large-scale enterprise applications. Its flexibility and modularity make it adaptable to evolving requirements and growth, ensuring that Python remains a viable choice as projects expand.
8. **Strong Industry Adoption**: Python's popularity has surged in recent years, with many tech giants, startups, and organizations embracing it for their projects. This widespread adoption translates into ample job opportunities, career growth, and community resources for Python developers.

**Slide 4: Installation**

1. **Official Python Website**: The official Python website (python.org) provides installation packages for various operating systems. Users can download the installer suitable for their platform (Windows, macOS, Linux) and follow the installation instructions provided.
2. **Integrated Development Environments (IDEs)**: Many integrated development environments offer built-in options to install Python:
   * **PyCharm**: PyCharm, a popular Python IDE, offers an option to download and install Python during the IDE installation process.
   * **Visual Studio Code (VS Code)**: VS Code provides extensions like Python for VS Code, which offers built-in support for Python and provides options to install Python directly from within the IDE.
   * **Anaconda**: Anaconda is a Python distribution that comes pre-packaged with popular data science libraries and tools. Users can download and install Anaconda, which includes Python along with other scientific computing packages.
3. **Operating System Package Managers**:
   * **Windows Store (Windows)**: On Windows 10, users can install Python directly from the Microsoft Store by searching for "Python" and clicking the "Install" button.
4. **Cloud Services and Platforms**: Some cloud service providers offer pre-configured environments with Python installed:
   * **Google Colab**: Google Colab provides free Jupyter notebooks with Python support. Users can run Python code in the browser without installing anything locally.
   * **AWS, Azure, and Google Cloud Platform**: Cloud platforms like AWS, Azure, and Google Cloud Platform offer virtual machines and container services where users can deploy Python applications.

**Slide 5: Good to Know**

* **Tips for Beginners:**
  + Google Colab provides free Jupyter notebooks with Python support.
  + Utilize online resources and communities for learning.
  + Learn GitHub along with Python to store all of your coding on shared repository

1. **Create a GitHub Account**: If you don't already have one, sign up for a GitHub account at <https://github.com/>.
2. **Create a New Repository**:
   * Click on the "+" icon in the top-right corner of the GitHub website and select "New repository".
   * Enter a name for your repository, choose whether it's public or private, add a description if desired, and then click "Create repository".
3. **Clone the Repository to Your Local Machine**:
   * Once your repository is created, you'll see its URL. Copy the URL.
   * Open your terminal or command prompt on your local machine.
   * Navigate to the directory where you want to store your code.
   * Run the following command, replacing **<repository\_url>** with the URL you copied:

git clone < **repository\_url** >

1. **Write Your Python Code**:
   * Create a new Python file or use an existing one to write your code.
2. **Add Your Code to the Repository**:
   * Once you've written your Python code, navigate to the directory where your repository was cloned.
   * Copy your Python file into this directory.
   * In your terminal or command prompt, navigate to the repository directory.
   * Run the following command to add your file to the staging area:

git add <file\_name.py>

Replace **<file\_name.py>** with the name of your Python file.

1. **Commit Your Changes**:
   * After adding your file to the staging area, commit your changes with a meaningful message describing what you've added or modified:

git commit -m “commit msg”ython script <file\_name.py>"

1. **Push Your Changes to GitHub**:
   * Once you've committed your changes, push them to your GitHub repository using the following command:

git push origin master

This command will push your committed changes to the **master** branch of your GitHub repository.

1. **Verify on GitHub**:
   * Go back to your GitHub repository's page in your web browser. You should see your Python file listed among the repository's contents.

**Slide 6: References**

* Official Python Documentation: [python.org](https://www.python.org/)
* Online Python Tutorials: W3Schools Python Tutorial, [Real Python](https://realpython.com/)

**Module 2: Python Basics**

1. **Variables in Python:**
   * In Python, a variable is a named location in memory used to store data.
   * Variables are created when a value is assigned to them using the assignment operator **=**.
   * Python is dynamically typed, meaning you don't need to declare the type of a variable explicitly.
2. **Variable Naming Rules:**
   * Variable names can contain letters (a-z, A-Z), digits (0-9), and underscores (\_).
   * They must start with a letter or an underscore.
   * Variable names are case-sensitive.
   * Python keywords cannot be used as variable names.
3. **Variable Assignment and Reassignment:**
   * You can assign a value to a variable using the = operator.
   * Variables can be reassigned with new values, even with different data types.
4. **Dynamic Typing:**
   * Python is dynamically typed, meaning you don't need to declare the type of a variable explicitly.
   * The type of a variable is inferred from the value assigned to it.
5. **The print statement:**
   * The **print()** function in Python is used to display output to the console.
   * It can accept multiple arguments separated by commas and prints them with spaces in between.
   * You can also use formatted strings (f-strings) with the print statement for more complex output.
6. **Comments:**
   * Comments in Python are used to explain the code and make it more understandable.
   * Single-line comments start with a **#** symbol, while multi-line comments are enclosed within triple quotes (**"""**).
   * Comments are ignored by the Python interpreter during execution.
7. **Python Data Structures & Data Types:**
   * Python supports various built-in data structures like lists, dictionaries, tuples, sets, etc.
   * Lists are ordered collections of items, dictionaries are key-value pairs, and tuples are immutable sequences.
   * Each data type has its own set of methods and operations that can be performed on it.
8. **String Operations in Python:**
   * Strings are sequences of characters enclosed within quotes (single, double, or triple).
   * Python offers various operations on strings like concatenation, slicing, formatting, etc.
   * String methods allow manipulation and transformation of string data.
9. **Simple input & output:**
   * **input()** function is used to take user input from the console.
   * User input is returned as a string by default, but can be converted to other data types using type casting.
   * **print()** function is used to display output to the console.
10. **Simple Output Formatting:**
    * Output formatting in Python allows you to control how data is displayed.
    * **format()** method and f-strings (formatted string literals) are commonly used for formatting output.
    * Formatting options include padding, aligning, truncating, and specifying precision for floating-point numbers.
11. **Operators in Python:**
    * Python supports various types of operators like arithmetic, comparison, logical, bitwise, etc.
    * Arithmetic operators (**+**, **-**, **\***, **/**, **//**, **%**, **\*\***) are used for basic mathematical operations.
    * Comparison operators (**<**, **>**, **<=**, **>=**, **==**, **!=**) are used to compare values.
    * Logical operators (**and**, **or**, **not**) are used for logical operations.

**Data Types**

1. **Numeric Types:**
   * Python supports various numeric types, including integers (**int**), floating-point numbers (**float**), and complex numbers (**complex**).
   * Integers represent whole numbers, while floating-point numbers represent decimal numbers.
2. **Sequence Types:**
   * Python supports several sequence data types, including lists (**list**), tuples (**tuple**), and strings (**str**).
   * Lists are mutable sequences of elements, while tuples are immutable sequences.
   * Strings represent sequences of characters and are also immutable.
3. **Mapping Type:**
   * Python's mapping type is the dictionary, which stores key-value pairs.
   * Keys within a dictionary must be unique, and they are typically immutable types like strings or numbers.
4. **Boolean Type:**
   * The boolean type (**bool**) represents the truth values **True** and **False**.
   * Booleans are often used in conditional statements and logical operations.

**Module 3: Python Program flow**

Python Program Flow refers to the sequence in which statements and instructions are executed in a Python program. It encompasses the order of execution, including conditional statements, loops, and function calls, which determine how the program behaves and processes data. In essence, it defines how the program moves from one statement to another, influencing its logic and behavior.

**Indentation**:

In Python, indentation is used to define blocks of code. Proper indentation is crucial for code readability and to determine the scope of functions, loops, and conditional statements.

**If statement and its related statement:**

The if statement is used to execute a block of code if a specified condition is true. Related statements include elif (short for "else if") and else.

**While loop**:

In Python, indentation is used to define blocks of code. Proper indentation is crucial for code readability and to determine the scope of functions, loops, and conditional statements.

**For loop**:

In Python, indentation is used to define blocks of code. Proper indentation is crucial for code readability and to determine the scope of functions, loops, and conditional statements.

**Range**:

In Python, indentation is used to define blocks of code. Proper indentation is crucial for code readability and to determine the scope of functions, loops, and conditional statements.

**Break and Continue**:

In Python, indentation is used to define blocks of code. Proper indentation is crucial for code readability and to determine the scope of functions, loops, and conditional statements.

**Module 4: Data Structure**

In Python, data structures are used to store and organize data effectively. Each data structure has its own characteristics, making it suitable for different tasks. Here are definitions and examples of some common data structures:

**List**:

A list in Python is a collection of elements that are ordered and mutable, meaning you can change its content after creation. Lists are versatile and can hold elements of different data types, such as integers, strings, or even other lists.

Ordered: Elements in a list are stored in a specific sequence and can be accessed using their index.

Mutable: Lists can be modified after creation. You can add, remove, or change elements in a list.

**Tuple:**

A tuple is similar to a list but immutable, meaning you cannot change its content after creation. Tuples are commonly used for grouping related data.

Ordered: Like lists, elements in a tuple are ordered and can be accessed using their index.

Immutable: Once a tuple is created, you cannot change its content (add, remove, or modify elements).

**Dict**:

A dictionary is a collection of key-value pairs. It's unordered, meaning the order of items is not guaranteed, and mutable, allowing for dynamic changes to its content.

Unordered: Unlike lists and tuples, dictionaries do not have a specific order for their elements.

Mutable: You can add, remove, or modify key-value pairs in a dictionary.

Key-Value Pair: Each element in a dictionary consists of a key and its associated value.

**Set**:

A set is an unordered collection of unique elements. It's mutable, meaning you can add or remove items, but sets themselves are immutable (you cannot change the elements once they're added).

Unordered: Like dictionaries, sets do not have a specific order for their elements.

Mutable: You can add or remove elements from a set.

Unique Elements: Sets do not allow duplicate elements; each element in a set is unique.

**List comprehension**:

List comprehension is a concise way to create lists in Python. It allows you to generate lists based on existing lists, iterables, or any iterable expression, with a compact and readable syntax. It's a powerful feature that can replace loops and conditional statements in many cases, making your code more expressive and succinct.