**Assignment 1**

* **Introduction to python and its features**
* Python is a widely used high-level, interpreted programming language. It was created by Guido van Rossum in 1991.
* Further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code
* [Python](https://www.geeksforgeeks.org/python-programming-language-tutorial/) is a programming language that lets you work quickly and integrate systems more efficiently.
* **Why It is named python:-**

Python is named after the British comedy series **Monty Python's Flying Circus**, rather than the snake.

* **Usage of Python**
* **Web Development:** Frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/), [Flask](https://www.geeksforgeeks.org/flask-tutorial/).
* [**Data Science**](https://www.geeksforgeeks.org/data-science-with-python-tutorial/)**and Analysis:** Libraries like [Pandas](https://www.geeksforgeeks.org/pandas-tutorial/), [NumPy](https://www.geeksforgeeks.org/python-numpy/" \t "_blank), [Matplotlib](https://www.geeksforgeeks.org/matplotlib-tutorial/" \t "_blank).
* [**Machine Learning**](https://www.geeksforgeeks.org/machine-learning/)**and AI:** TensorFlow, PyTorch, Scikit-learn.
* **Automation and Scripting:** Automate repetitive tasks.
* **Game Development:** Libraries like Pygame.
* [**Web Scraping**](https://www.geeksforgeeks.org/python-web-scraping-tutorial/)**:** Tools like [BeautifulSoup](https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/" \t "_blank), Scrapy.
* **Desktop Applications:** GUI frameworks like [Tkinter](https://www.geeksforgeeks.org/python-gui-tkinter/" \t "_blank), PyQt.
* **Scientific Computing:** SciPy, SymPy.
* **Internet of Things (IoT):** MicroPython, Raspberry Pi.
* [**DevOps**](https://www.geeksforgeeks.org/devops-tutorial/)**and Cloud:** Automation scripts and APIs.
* **Cybersecurity:** Penetration testing and ethical hacking tools.

**Key features of python**

* **Python is Easy to Learn and Use:** There is no prerequisite to start Python, since it is Ideal programming language for beginners.
* **High Level Language:** Python don’t let you worry about low-level details, like memory management, hardware-level operations etc.
* **Python is Interpreted:**Code is executed line-by-line directly by interpreter, and no need for separate compilation. Which means –

You can run the same code across different platforms.

You can make the changes in code without restarting the program.

* **Dynamic Typed:**Python is a dynamic language, meaning there are no need to explicitly declare the data type of a variable. Type is checked during runtime, not at compile time.
* **Object Oriented:**Python supports object-oriented concepts like classes, inheritance, and polymorphism etc. OOPs empowers Python with modularity, reusability and easy to maintain code.
* **Extensive Library are Available:**Python has huge set of library and modules, which can make development lot easier and faster.
* **Open-Source with Huge community Support:**Along with opensource, Python is blessed with very large community contributing to its further development.
* **Cross Platform:**Same Python code can run on Windows, macOS and Linux, without any modification in code.
* **Good Career Opportunities:** Python is in high demand across industries like Software development, AI, finance, and cloud computing etc.

**History and evolution of python**

* **1980s**: Python’s creator, **Guido van Rossum**, began working on Python in the late 1980s at **Centrum Wiskunde & Informatica (CWI)** in the Netherlands
* **December 1989**: Guido van Rossum began the project that would become Python.
* **February 1991**: **Python 1.0** was released
* **1994**: **Python 1.2** was released, adding more features like the sys module and better exception handling.
* **October 1995**: **Python 1.4** introduced **lambda functions** and **list comprehensions**, making the language more powerful and concise.
* **October 2000**: **Python 2.0** was released, marking a major milestone in the language’s development
* **2001**: **Python 2.1** introduced nested scopes and new style classes, which were crucial for object-oriented programming.
* **2003**: **Python 2.3** added **iterators**, **generators**, and **set literals**.
* **2005**: **Python 2.4** introduced **decorators**, a powerful tool in Python for modifying functions or methods.
* **2006**: **Python 2.5** added **with** statement (context managers) and **function decorators**.
* **2008**: **Python 2.6** provided compatibility improvements for Python 3.0
* **July 2010**: **Python 2.7** was released, marking the final version of the Python 2.x series **December 2008**: **Python 3.0** was released as a major upgrade with **backward compatibility breaks** from Python 2.x.
* **June 2009**: **Python 3.1** added **performance improvements**, such as **better memory management** and **faster dictionaries**
* **2010**: **Python 3.2** introduced **@staticmethod** and better **multithreading** support. It also added new built-in modules, like concurrent.futures.
* **2012**: **Python 3.3** introduced:
* **Virtual environments**.
* **yield from** for working with iterators.
* Improved **Unicode** handling
* **2014**: **Python 3.4** brought several important features such as asynio,Pathlib,type hinting
* **2015**: **Python 3.5** introduced
* **Type hints** with **PEP 484**.
* **async/await** syntax for asynchronous programming.
* **2017**: **Python 3.6** introduced:
* **f-strings** for easier string formatting.
* **Type annotations** for more structured and readable code.
* **Underscores in numeric literals** (e.g., 1\_000\_000).
* **2018**: **Python 3.7** brought:
* **Data classes** (@dataclass) for simpler class definitions.
* **Postponed evaluation of type annotations** (PEP 563).
* **Built-in breakpoint()** function for debugging.
* **2020**: **Python 3.8** added:
* The **walrus operator** (:=) for assignment expressions.
* Positional-only parameters (/ in function signatures).
* Improvements in typing, such as **TypedDict**.
* **2021**: **Python 3.9** introduced:
* **Dictionary merge (|)** and **update (|=)** operators.
* Type hinting improvements, such as **TypeGuard**.
* Removal of deprecated features from previous versions.
* **2022**: **Python 3.10** was released, featuring:
* **Structural Pattern Matching** (similar to switch/case).
* **Parenthesized Context Managers** for more readable with statements.
* Improved error messages for easier debugging.
* **2023**: **Python 3.11** focused on **performance improvements**, making Python faster and more efficient:
* 2024 : Python 3.13

**Advantages of using python over other programming languages**

* **Readability and Simplicity**: Python's syntax is clear and easy to read, which makes it an excellent choice for beginners. The language emphasizes readability, allowing developers to express concepts in fewer lines of code compared to languages like C++ or Java.
* **Versatility**: Python is a general-purpose programming language that can be used for a wide range of applications, including web development, data analysis, artificial intelligence, machine learning, automation, scientific computing, and more.
* **Rich Ecosystem and Libraries**: Python has a vast collection of libraries and frameworks (such as NumPy, Pandas, TensorFlow, Flask, Django, etc.) that simplify complex tasks and speed up development. This extensive ecosystem allows developers to leverage existing tools rather than building from scratch.
* **Community Support**: Python has a large and active community, which means that developers can easily find resources, tutorials, and forums for help. This community support fosters collaboration and knowledge sharing.
* **Cross-Platform Compatibility**: Python is cross-platform, meaning that code written on one operating system can often run on others with little or no modification. This flexibility is beneficial for developers working in diverse environments.
* **Integration Capabilities**: Python can easily integrate with other languages and technologies, such as C, C++, and Java. This makes it a good choice for projects that require interaction with other systems or legacy code.
* **Rapid Development:** Python's simplicity and the availability of powerful libraries allow for rapid prototyping and development. This is particularly advantageous in startups and research environments where time-to-market is critical.
* **Strong Support for Data Science and Machine Learning:** Python has become the de facto language for data science and machine learning, thanks to libraries like Pandas, NumPy, SciPy, and scikit-learn. Its ease of use and powerful data manipulation capabilities make it ideal for these fields.
* **Dynamic Typing:** Python uses dynamic typing, which allows for more flexibility in coding. Developers can write code without explicitly declaring variable types, which can speed up development and reduce boilerplate code.
* **Extensive Documentation:** Python has comprehensive and well-organized documentation, making it easier for developers to learn and reference the language's features and libraries.
* **Support for Multiple Programming Paradigms**: Python supports various programming paradigms, including procedural, object-oriented, and functional programming. This flexibility allows developers to choose the best approach for their specific problem.
* **Strong Emphasis on Testing:** Python has built-in support for unit testing and test-driven development (TDD), which encourages developers to write tests and maintain code quality.

**Installing and setup for python**

* **Step 1: Install Python**
* **Download Python**:
  + Go to the official [Python website](https://www.python.org/downloads/).
  + Download the latest stable version of Python for your operating system (Windows, macOS, or Linux).
* **Install Python**:
  + During installation, ensure you check the box **Add Python to PATH**. This step ensures that you can run Python from the command line easily.
  + Follow the installation prompts to complete the setup.
* **Verify Installation**:
  + Open your terminal or command prompt and type the following command:
    - bash
    - Copy code
    - python --version
    - or
    - bash
    - Copy code
    - python3 --version
  + You should see the installed version of Python. If not, the installation might not have been successful, and you might need to check your environment variables.
* **Step 2: Install Visual Studio Code (VS Code)**
* **Download VS Code**:
  + Visit the official [VS Code website](https://code.visualstudio.com/) and download the version for your operating system.
* **Install VS Code**:
  + Run the installer and follow the installation prompts. Once installed, you can launch VS Code.
* **Step 3: Install Python Extension for VS Code**
* **Open VS Code**: Launch Visual Studio Code.
* **Install Python Extension**:
  + On the left sidebar of VS Code, click the **Extensions** view icon (or press Ctrl+Shift+X).
  + Search for **Python** in the Extensions Marketplace.
  + Find the **Python extension** by Microsoft and click **Install**.
  + This extension provides features like IntelliSense (auto-completion), linting (error checking), debugging, and more, making it a powerful tool for Python development.
* **Step 4: Configure Python Interpreter in VS Code**
* **Open Command Palette**:
  + Press Ctrl+Shift+P (or Cmd+Shift+P on macOS) to open the Command Palette.
* **Select Python Interpreter**:
  + Type **Python: Select Interpreter** in the Command Palette and hit Enter.
  + A list of available Python interpreters will appear. Select the appropriate interpreter (usually the one installed on your system or any virtual environment you are using).
* **Step 5: Set Up a Python Virtual Environment (Optional but Recommended)**
* Creating a virtual environment is a good practice as it isolates your project’s dependencies. Here's how to set it up:
* **Navigate to Your Project Folder**: Open your terminal (or use the integrated terminal in VS Code) and navigate to your project folder using the cd command:
  + bash
  + Copy code
  + cd path/to/your/project
* **Create a Virtual Environment**: Run the following command to create a virtual environment (replace env with your desired environment name):
  + bash
  + Copy code
  + python -m venv env
* **Activate the Virtual Environment**:
  + On **Windows**:
    - bash
    - Copy code
    - .\env\Scripts\activate
  + On **macOS/Linux**:
    - bash
    - Copy code
    - source env/bin/activate
* **Install Dependencies**: Once the virtual environment is activated, you can install any dependencies (like libraries) using pip:
  + bash
  + Copy code
  + pip install <package-name>
* **Deactivate the Virtual Environment**: When you're done working in your virtual environment, you can deactivate it with:
  + bash
  + Copy code
  + deactivate
* **Step 6: Install Additional Python Packages in VS Code**
* You can install Python packages in the integrated terminal within VS Code:
* Open the integrated terminal in VS Code by going to **View > Terminal** or pressing Ctrl+ (backtick).
* Install Python packages using pip. For example:
  + bash
  + Copy code
  + pip install requests
* **Step 7: Running Python Code in VS Code**
* **Create a Python File**:
  + In VS Code, open your project folder and create a new file with the .py extension (e.g., hello.py).
* **Write Your Python Code**: For example, write the following code in your .py file:
  + python
  + Copy code
  + print("Hello, world!")
* **Run Python Code**:
  + You can run your Python code directly in VS Code by opening the integrated terminal (Ctrl+), and typing:
    - bash
    - Copy code
    - python hello.py
  + Alternatively, if you have the Python extension installed, you can press the green **Run** button in the top-right corner of the editor window.
* **Step 8: Debugging in VS Code**
* VS Code comes with an integrated debugger that helps you run and debug Python code efficiently.
* **Set a Breakpoint**:
  + In your Python code, click on the left margin next to the line number where you want to pause the execution. This will set a breakpoint.
* **Start Debugging**:
  + Press F5 or click the green **Run and Debug** button to start the debugger.
  + The debugger will pause execution at the breakpoint, and you can step through the code using the debug toolbar.