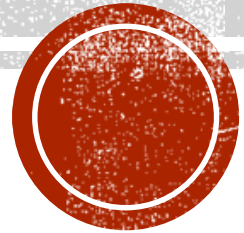
PYTHON

Day 1 NTI Sohag



OUTLINES

- Python pros.
- Interpreted vs compiled programming languages.
- IDE vs Editor vs compiler vs interpreter.
- Kaggle.
- Google Colab.
- Tasks.

WHY

EASE OF USE

- **Clear Syntax:**
  - Python uses indentation instead of braces or semicolons, making the structure of the code visually apparent.
- **Simple Commands:**
  - Many tasks can be accomplished with fewer lines of code compared to other languages.
  - For example, file handling or string manipulation can be done with straightforward commands.
- **Learning Curve:**
  - Python is often recommended as a first language for new programmers because it allows them to focus on learning programming concepts without getting bogged down by complex syntax.

Syntax comparison

Python	Java	CPP

`print("Hello, World!")`	`public class HelloWorld {`
	`  public static void main(String[] args) {`
	`  System.out.println("Hello, World!");`
	`  }`
	`}`
		`#include <iostream>`
		
		`int main() {`
		`  std::cout << "Hello, World!" << std::endl;`
		`  return 0;`
		`}`

VERSATILITY

- **Web Development**
  - Django and Flask framworks.
- **Data Science and Analytics**
  - Pandas, NumPy, and Matplotlib for data manipulation and visualization.
- **Machine Learning and AI**
  - Python's extensive libraries, such as TensorFlow, PyTorch, and scikit-learn, make it an ideal choice for developing machine learning models and AI applications.
- **Automation**
- Web Scraping (selenium) , Interacting with an API (requests), control in mouse and keyboard (Pyautogui)…etc.

RICH ECOSYSTEM

- Python's ecosystem is one of its greatest strengths, providing tools and libraries for almost any task:
- **Libraries**: Python's extensive library collection, like NumPy for numerical computations, Pandas for data analysis, and Matplotlib for plotting, makes it highly efficient for specialized tasks.
- **Frameworks**: Frameworks such as Django (for web development) and TensorFlow for machine learning, allowing developers to focus on application logic rather than boilerplate code.
- **Third-Party Packages**: The Python Package Index (PyPI) hosts over 300,000 packages, allowing developers to extend Python's capabilities effortlessly.

PYPI

- the central repository for Python software packages.
- It's where you can find, install, and share Python libraries.
- Think of it as a vast library of tools and building blocks for Python developers.
- pip install library_name

COMMUNITY SUPPORT

- Python's community is one of the largest and most active in the programming world, offering immense support and resources:
- **Documentation**: Python's official documentation is comprehensive and beginner-friendly, covering everything from basic syntax to advanced features.
- **Online Resources**: The Python community provides an abundance of tutorials, guides, forums (like Stack Overflow), and blogs where developers can learn and troubleshoot issues.
- **Open Source Contributions**: Python's community continuously contributes to the development of new libraries, tools, and improvements to the language itself, ensuring that it evolves to meet current needs.

STATISTICS

		CRITERIA 1	CRITERIA 2	CRITERIA 3	CRITERIA 4	CRITERIA 5	
Ranking	Language	# of Jobs Available	Avg Salary (USD)	Devs Love It	Difficulty to Learn	Future Proof	TOTAL
1	Python	12	9	10	5	5	41
2	SQL	11	3	9	8	3	34
3	TypeScript	3	11	11	3	3	31
4	Rust	6	4	12	3	3	28
5	JavaScript	8	2	5	5	5	25
6	C++	9	8	4	1	3	25
7	Java	10	5	2	2	5	24
8	Golang	7	6	7	3	3	26
9	C#	5	7	8	2	3	25
10	Ruby	2	12	3	5	2	24
11	Kotlin	1	10	6	2	2	21
12	PHP	4	1	1	5	1	12

WHAT

COMPILED LANGUAGES

- In a compiled language, the source code is first translated into machine code by a compiler. This machine code is then executed directly by the computer's CPU.
- **Compilation Step:** The source code is compiled into an executable file (often with a .exe or similar extension) before it can be run. This step creates a binary file that the computer can execute directly.
- **Faster Execution:** Since the machine code is already generated, compiled programs typically run faster than interpreted programs.
- **Platform Dependence:** Compiled code is often specific to the platform it was compiled for.
  - For example, a program compiled on Windows may not run on Linux without recompilation.
- **Error Checking:** Compilers perform a thorough check of the code during compilation, catching many errors before the program is run.
- **Examples**: C, C++, Go, Rust, Fortran

INTERPRETED LANGUAGES

- In an interpreted language, the code is executed line-by-line by an interpreter at runtime. The interpreter reads the source code, translates it into machine code on the fly, and then executes it immediately.
- **Execution:** The interpreter processes the code one instruction at a time, which allows for dynamic and interactive programming.
- **No Compilation Step:** There is no need to compile the code before running it. This makes the development process faster since you can immediately see the results of your code.
- **Platform Independence:** Since the source code is interpreted by an interpreter specific to each platform, the same Python script, for example, can run on different operating systems without modification.
- **Slower Execution:** Because the interpreter translates code line-by-line during execution, interpreted languages tend to be slower compared to compiled languages.
- **Examples**: Python, JavaScript, Ruby, PHP, Perl.

DIFFERENCES

- **Translation Method:**
  - **Interpreted:** Translates code line-by-line at runtime.
  - **Compiled:** Translates the entire code into machine code before
- **Execution Speed:**
  - **Interpreted:** Generally slower due to the on-the-fly translation.
  - **Compiled:** Faster, since the code is pre-translated into machine code.
- **Platform Dependency:**
  - **Interpreted:** More platform-independent; the interpreter handles platform-specific details.
  - **Compiled:** Often platform-dependent; code needs to be recompiled for different platforms.
- **Development Cycle:**
  - **Interpreted:** A shorter cycle, as changes can be tested immediately.
  - **Compiled:** Longer cycle, as code must be recompiled after changes.
- **Error Detection:**
  - **Interpreted:** Errors are caught at runtime, which can be beneficial for debugging but might also lead to runtime failures.
  - **Compiled:** Many errors are caught at compile-time, reducing the likelihood of runtime issues.

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

- An Integrated Development Environment (IDE) is a comprehensive software suite that provides all the tools a developer needs to write, compile, run, and debug code.
- **Code Editor:** A text editor specifically designed for writing code, often with syntax highlighting, auto-completion, and error detection.
- **Compiler/Interpreter Integration:** An IDE typically includes or integrates with a compiler or interpreter, allowing you to compile or run your code directly within the environment.
- **Debugger:** A tool that helps you find and fix bugs by allowing you to run code step-by-step, inspect variables, and analyze the program's flow.
- **Build Automation:** Many IDEs include tools to automate the process of building (compiling and linking) your application.
- **Version Control Integration:** IDEs often integrate with version control systems like Git, allowing you to manage code changes and collaborate with others.
- **EX:**
  - Visual Studio (for C#, C++, etc.)
  - Eclipse (for Java)
  - PyCharm (for Python)
  - IntelliJ IDEA (for Java and other languages)

EDITOR

- A code editor is a text editor specifically designed for writing and editing code. While it lacks the full suite of tools found in an IDE, it is often lighter and faster, making it ideal for quick edits or for developers who prefer a more minimalist environment.
- **Syntax Highlighting:** Different parts of the code (keywords, variables, etc.) are displayed in different colors, making it easier to read.
- **Auto-Completion:** The editor can suggest completions for keywords, function names, etc., as you type.
- **Extensions/Plugins:** Many editors allow you to add functionality through extensions or plugins, such as linters, debuggers, and version control tools.
- **Lightweight:** Editors are generally faster and use fewer resources than full IDEs, making them ideal for quick tasks or working on smaller projects.
- Sublime Text, Notepad++

COMPILER

- A compiler is a program that translates source code written in a high-level programming language into machine code (binary code) that a computer's CPU can execute directly.
- **Translation Process:** The compiler reads the entire source code, translates it into machine code, and creates an executable file.
- **Error Checking:** Compilers check the source code for syntax errors and other issues during the compilation process, providing error messages if the code is incorrect.
- **Optimization:** Compilers often optimize the code during the translation process to make it run faster or use less memory.
- **Platform Dependence:** The generated machine code is often specific to the operating system and hardware it was compiled on, though some compilers can produce cross-platform binaries.
- **Examples of Compilers:** GCC (GNU Compiler Collection) for C, C++, and other languages.javac (Java Compiler) for Java.Clang (for C, C++, Objective-C).MSVC (Microsoft Visual C++) for C/C++.

INTERPRETER

- An interpreter directly executes the instructions in the source code line-by-line, without translating it into machine code beforehand.
- **Line-by-Line Execution**: The interpreter reads and executes each line of code sequentially, which allows for more dynamic behavior and easier debugging.
- **No Executable File**: Unlike a compiler, an interpreter does not produce a separate executable file. The source code must be re-interpreted each time it runs.
- **Slower Execution**: Because the code is translated on-the-fly during execution, interpreted programs generally run slower than compiled ones.
- **Platform Independence**: Since the interpreter handles the platform-specific details, the same code can often be run on different platforms without modification.
- **Examples of Interpreters:**
  - **Python Interpreter** (for Python code).
  - **Node.js** (for JavaScript code).
  - **Ruby Interpreter** (for Ruby code).
  - **PHP Interpreter** (for PHP code).

ONLINE COMPILERS

- https://www.programiz.com/python-programming/online-compiler/
- And so on

GOOGLE COLAB

- https://colab.research.google.com/

KAGGLE

- https://www.kaggle.com/

ANACONDA INSTALLATION

- https://www.anaconda.com/download/success
- Spyder
- Jupyter notebook
- Virtual environments
- Anaconda prompt

HOW TO CREATE ENVIRONMENT

- conda create -n reviews python=3.10
- # then activate the environment to install libraries in it
- conda activate reviews
- #then install ipykernel
- pip install ipykernel
- #then
- python -m ipykernel install --user --name reviews --display-name "reviews"
- #then install jupyter notebook
- conda install jupyter
- #to install spyder
- conda install spyder
- #to open jupyter notebook
- jupyter notebook
- #to open spyder
- spyder

TASKS

STUDENT GRADES MANAGEMENT

- **Problem Statement:**
  - You are tasked with developing a program to manage student grades. You need to store students' names, along with their grades for different subjects.
- **Requirements:**
  - Store each student's information as a tuple of the form (name, {'subject1': grade1, 'subject2': grade2, ...}).
  - Create a list of students, each represented by the above tuple.
  - Implement a function to add a new student to the list.
  - Implement a function to update the grades for a particular student.
  - Implement a function to calculate the average grade of a student.
  - Implement a function to find the student with the highest average grade.

SHOPPING CART SYSTEM

- **Problem Statement:**
  - You need to create a shopping cart system for an e-commerce application. The cart should keep track of items, their prices, and quantities.
- **Requirements:**
  - Represent each item as a tuple (item_name, price).
  - Use a list to store items added to the cart.
  - Use a dictionary to keep track of the quantity of each item in the cart, with the item name as the key and the quantity as the value.
  - Implement functions to add items to the cart, update the quantity of an item, and remove an item from the cart.
  - Implement a function to calculate the total price of items in the cart.

EMPLOYEE MANAGEMENT SYSTEM

- **Problem Statement:**
  - Develop a system to manage employee data, including their ID, name, position, and salary.
- **Requirements:**
  - Store each employee's data in a tuple (ID, name, position, salary).
  - Use a list to maintain a collection of employees.
  - Implement a function to add a new employee to the list.
  - Implement a function to find an employee by their ID and return their details.
  - Implement a function to update an employee’s salary based on their ID.
  - Implement a function to find all employees with a salary above a certain threshold and return their names.

ANACONDA

- Please install anaconda