Prepared By - PRIYANKA NEOGI

Topics Completed :

1. Python as Calculator:

- Python can be used as a calculator for performing various mathematical calculations.
- You can use it to perform basic arithmetic operations like addition, subtraction, multiplication, and division, as well as more complex mathematical operations.
- You can use Python's interactive shell or create a script to perform more complex calculations and manipulate data as needed.
- Python's extensive standard library provides many additional mathematical functions and modules for specialized calculations

```
1+2

3

3453+345

3798
```

API :

```
API in simple terms using a common real-world analogy:


You're at a restaurant. You -->(the customer) want to order food from
the
kitchen --> (the chef).
```

```
However, you can't just walk into the kitchen and start cooking
yourself.
There's a waiter --> (the API) who takes your order and delivers it to
the kitchen.
The waiter knows how to communicate your request to the chef and how
to bring the cooked food back to you.

You (the customer): You represent a software program or app that wants
to do something, like
requesting information or performing a task.


The Kitchen (the chef): This is like a computer system or service that
has the information or functionality
you need. It could be a weather service, a social media platform, or
any other piece of software.

The Waiter (the API): The waiter is like an API, which stands for
"Application Programming Interface."
Just as the waiter takes your order and conveys it to the chef, an API
takes your request and sends it to
the computer system. It also brings back the system's response to you.
The waiter doesn't have to know the cooking.


Your Order (the request): This is the specific thing you want the
kitchen to do, like "I'd like a
cheeseburger" or "Tell me the weather in New York." In the digital
world, your order is a set of
instructions or data sent to the API.

The Food (the response): This is what you get back from the kitchen
after your order has been processed. In the digital realm, it could be
information, like the weather forecast, or it could be an action
performed, like posting a tweet on social media.
```

An API is like a waiter that helps different software programs communicate and work together. It makes it possible for your app to request services or data from other apps or systems without having to know how those other apps are built or work internally. It's a way for different pieces of software to "talk" to each other and share information, just like a waiter helps you get the food you want from the kitchen.

API in Technical Terms:

- An API, or `Application Programming Interface`, is a set of rules and protocols that allows different software applications to communicate with each other.
- It defines the methods and data formats that developers can use to request and exchange information between different software systems, whether they are running on the same computer or distributed across a network.

- APIs serve as intermediaries that enable different software components to interact and share data without requiring the developers to understand the internal workings of the systems they are communicating with.
- This abstraction makes it easier to build complex applications by leveraging the functionality provided by other software services or platforms.

Use Cases: APIs are used for a wide range of purposes, including accessing web services (e.g., weather data, social media platforms), integrating with third-party software (e.g., payment gateways, social logins), and building modular, scalable applications.

Examples of well-known APIs include:

1. Google Maps API: Allows developers to integrate maps and location-related services into their applications.
2. Twitter API: Provides access to Twitter's data and functionalities, allowing developers to post tweets, retrieve user timelines, and more.
3. Facebook Graph API: Enables interactions with Facebook's social graph and user data.
4. GitHub API: Allows developers to interact with GitHub repositories, issues, and other platform features programmatically.

Developers often use APIs to enhance their applications, add new features, or integrate with third-party services, saving time and effort by leveraging existing functionality.

IDE (Integrated Development Environment) :

- There are many interpreters available freely to run Python scripts like IDE (Integrated Development Environment) that comes bundled with the Python software available for developers to choose from.

For Example - PyCharm, Visual Studio Code (VS Code), Spyder, IDLE(Integrated Development and Learning Environment)

2. The rising popularity of Python :

1. **Enhanced Code Readability and Conciseness**: Python places a strong emphasis on code readability and allows programmers to write shorter, more concise code, making it easier to understand and maintain. Less verbosity.

2. **Efficient Expression of Logical Concepts**: Python enables developers to express complex logical concepts using fewer lines of code when compared to languages such as C++ or Java.

3. **Support for Multiple Programming Paradigms**: Python is versatile, accommodating multiple programming paradigms, including object-oriented, imperative, functional, and procedural programming. This flexibility appeals to a wide range of developers.

4. **Rich Set of Built-in Functions**: Python provides an extensive collection of built-in functions that cover a broad spectrum of frequently used programming concepts. This simplifies development by reducing the need for custom code for common tasks.

- Known as the “batteries included” (to describe Python's philosophy) ;It can help do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, email, XML, HTML, WAV files, cryptography, GUI and many more.

5. **Philosophy of Simplicity**: Python's core philosophy is rooted in the principle that "simplicity is the best." This philosophy permeates the language's design, making it accessible and intuitive for both beginners and experienced programmers alike.

6. **Robust Handling**: Robustness is evident through its exceptional

   - **exception handling capabilities**(it can handle them gracefully without crashing or causing major disruptions. This is important for ensuring the stability and reliability of the system) and

   - **built-in memory management techniques**(Memory management is crucial in computing to allocate and deallocate memory efficiently, preventing issues like memory leaks or excessive memory usage. Having built-in memory management techniques means that the system can efficiently utilize memory resources, further contributing to its robustness).

7. **Presence of third-party modules**: In Python it refers to the extensive ecosystem of external libraries and packages that are created by developers and organizations outside of the Python core development team. These third-party modules can be seamlessly integrated into Python programs to extend its functionality and provide solutions to a wide range of tasks and challenges

8. **(IoT)Internet of Things Opportunities**: It refers to the interconnectedness of everyday objects and devices through the internet, enabling them to collect and exchange data, communicate with each other, and be remotely controlled or monitored.

DISADVANTAGES OF PYTHON :

1. **Performance Concerns**: Python, being an interpreted language, may exhibit slower execution compared to compiled languages like C or Java, posing challenges for performance-critical tasks.

2. **Global Interpreter Lock (GIL)**: Python's Global Interpreter Lock (GIL) restricts concurrent execution of Python code by multiple threads, potentially limiting the parallelism and concurrency capabilities of certain applications, which can limit its ability to take advantage of multiple CPU cores.

3. **High Memory Usage**: Python can be memory-intensive, particularly when handling extensive datasets or executing intricate algorithms, leading to increased memory consumption.

4. **Dynamic Typing**: Python's dynamic typing allows variable types to change during runtime, which can complicate error detection and potentially introduce bugs.

5. **Package and Version Management**: Python's extensive library ecosystem may occasionally result in version conflicts and package-related challenges when integrating multiple libraries into projects.

6. **Flexibility vs. Rigidity**: Python's flexibility, while advantageous for rapid development and prototyping, can also make code less structured and harder to maintain, presenting a trade-off between adaptability and maintainability.

7. **Learning Curve**: Python, often considered beginner-friendly, can still pose a steep learning curve for newcomers, especially those with no prior programming experience.

8. **Exclusion from Mobile and Browser Environments**: Python may not be well-suited for mobile computing and browser-based applications, as it may lack comprehensive support and performance optimization in these contexts.

3. Python Variable

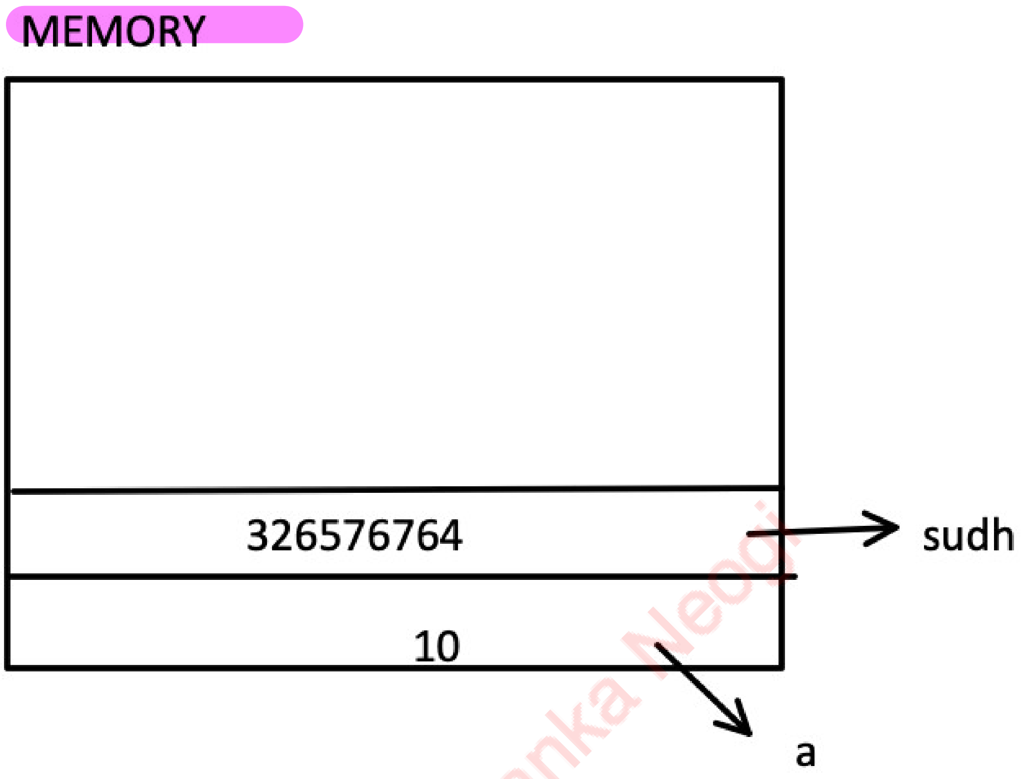
```
1. Question - When writing code in any programming language, what is
the purpose of writing that code?

We write code to implement logic.

2. Question - Why do we write this logic?

We write logic because, ultimately, we need to manipulate and work
with data.

Consider a simple website as an example. Even on such websites, when
we interact with buttons or call APIs, it involves processing data.

In every programming language and in every process, the logic we write
ultimately deals with processing
data. However, for the system to process data, it first needs to know
where to store that data.

The system must have an understanding of the data's location before it
can begin processing it. Without this understanding, the system will
not initiate the processing of the data.

This is where variables come into play.
```

In Python, when we're working on solving a problem or implementing a particular logic, for processing or to manipulate and work with data we have to store the data. To do this effectively, we use variables to store and manage that data.

A variable is essentially a placeholder that allows us to store various types of data, which we can then manipulate using the logic we've written. That's what a variable is all about.

```python
a = 10
```

```
a
10
```



When 'a' equals 10, it signifies that the system is attempting to establish a space for data, with 10 representing my specific data. 'a' serves as the container for storing the value 10.

Now, what occurs internally?

- A computer comprises components like the CPU, RAM, and hard disk for data storage. RAM is used for temporary data storage, while the CPU handles various operations. When we assign a value of 10 to a variable, the system attempts to access memory, which could be either RAM or CPU memory, depending on how the variable is processed.

- Since we are dealing with a temporary variable, it is stored in the main memory (RAM). The system allocates space for the value 10 within RAM, associating it with variable 'a.' Thus, the value 10 is stored in this reserved space.

The value 10 serves as a reference for what?

- It acts as a reference for the variable 'a.' I am associating 'a' with the data 10, so whenever you request the value of 'a,' it will provide you with the value 10.

- For instance, consider my name, 'my name.' As a person, my name is a placeholder that represents me, and it reflects my characteristics and attributes when talking or working. Similarly, when we use a variable to store data, like variable 'a,' it serves as a container where the value 10 is stored. When you access variable 'a,' it returns the property, which is 10 in this case.

In essence, a variable is a placeholder that can hold various types of data.

You have the flexibility to assign any name you like your name as a variable name.

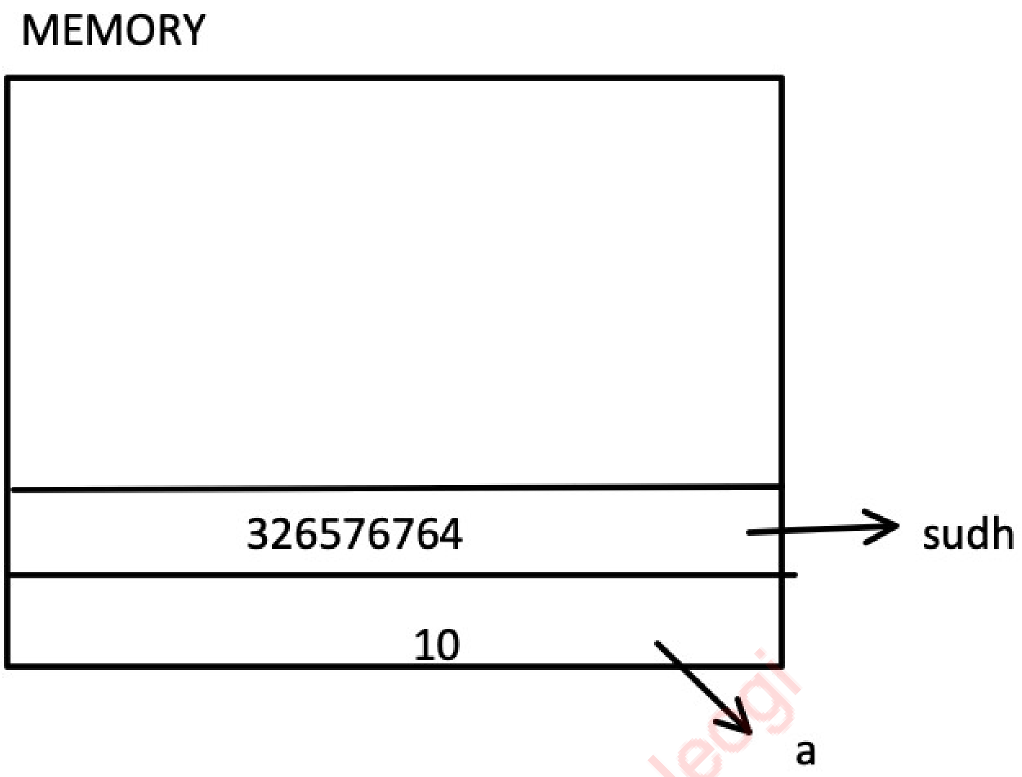
- The '=' symbol is indeed the assignment operator in many programming languages. Its primary purpose is to assign a value to a variable, effectively storing that value in the variable.

```
type(a)

int
```

type( ):

- To determine the type of a variable in many programming languages, you can use the `type` function or a similar built-in function. This function typically returns the data type or class of the variable, helping you understand the kind of data it contains.
- I'm attempting to store data within a variable called 'a,' which resides in memory. In contrast to some other programming languages where you must explicitly specify the data type before assigning a value to a variable, data in Python is more flexible.
- In Python, there's no requirement for explicit data type declarations when defining variables. You don't need to specify the data type for a particular variable beforehand.
- Despite not explicitly declaring the data type, Python has the ability to automatically deduce the data type based on the characteristics of the data itself.
- Python is considered a dynamically typed language, meaning that it can decide the data type without prior specification. This self-detecting property allows Python to adapt to the nature of the data, even if you've never explicitly mentioned the type you intend to store.
- With these different types of stored variables having different data types in memory we can further process the data as per requirement.

```
sudh = 326576764
```



When I attempt to call 'sudh,' it provides me with the exact data contained within it.

- It is attempting to allocate a dedicated space in memory for the data associated with the variable named 'sudh.' * Within this allocated memory space, it stores the value '326576764.'
- This value serves as a reference to the variable 'sudh.'
- So, whenever I make a call to 'sudh,' it retrieves and provides me with the data '326576764.'

```
sudh
```

```
326576764
```

Keyword - None:

This is a unique constant employed to represent a null value or absence of a value.

It's crucial to keep in mind that neither 0 nor any empty container (such as an empty list) equates to None.

None is an instance of its own data type, known as NoneType.

It is not feasible to generate multiple instances of None and assign them to variables.

```
print(None == 0)
```

```
False
```