My name is Ifshita Ghosh Saha, and I am currently in my final semester of the Master of Computer Applications (MCA) program at Techno India University, Kolkata. I hold a Bachelor's degree in Economics with honors from Ashutosh College, under the University of Calcutta, where I also studied Statistics and Mathematics as my additional subjects. I completed my schooling at Kendriya Vidyalaya No. 2, Salt Lake.

In terms of technical skills, I am proficient in Python, SQL, and HTML. Beyond my academic and technical pursuits, I enjoy dancing, cooking, and watching movies.

Certainly! Here’s another polished version of your introduction:

My name is Ifshita Ghosh Saha, currently in my final semester of the MCA program at Techno India University, Kolkata. I have a Bachelor's degree in Economics with honors from Ashutosh College, University of Calcutta, where I also studied Statistics and Mathematics. I completed my schooling at Kendriya Vidyalaya No. 2, Salt Lake

Technically, I am skilled in Python, SQL, and HTML. Outside academics, I enjoy dancing, cooking, and watching movies.

One of my weaknesses is that I find it difficult to say no to requests, which sometimes leads to me taking on too much work. To address this, I am learning to set clearer boundaries and prioritize tasks better.

One of my strengths is staying calm and patient in difficult situations. While working at a bank, I often dealt with upset customers, resolving their issues with patience and empathy, which enhanced my conflict resolution and customer service skills.

Moreover, I excel at multitasking. My involvement in extracurricular activities like playing on the kho-kho team during school helped me develop excellent time management and teamwork skills, allowing me to handle multiple responsibilities effectively.

---

This version maintains a professional tone while succinctly highlighting your background, skills, and experiences.

**Question: Why did you shift from an economics background to IT?**

**Answer:**

My journey from economics to IT has been driven by my evolving interests and the desire to solve complex problems using technology. During my undergraduate studies in Economics at Ashutosh College, I developed a strong analytical and quantitative foundation, which was further enhanced by my coursework in Statistics and Mathematics. This background provided me with a solid understanding of data analysis, critical thinking, and problem-solving skills.

After graduation, I started working in the banking sector, where I was exposed to various financial technologies and the critical role they play in optimizing banking operations. My experience in the bank allowed me to see firsthand how technology can streamline processes, enhance customer experiences, and drive innovation in the financial industry.

This exposure sparked my interest in technology and inspired me to transition into the IT field. I decided to pursue a Master of Computer Applications (MCA) at Techno India University to gain the technical skills necessary for a career in IT. Through this program, I have become proficient in programming languages like Python, and acquired expertise in SQL and HTML, which have equipped me with the tools to develop software solutions and manage data effectively.

Moreover, my background in economics has given me a unique perspective on data-driven decision-making and financial analysis, which are valuable in IT roles that involve data science, business analysis, and fintech solutions. By combining my analytical skills with my technical knowledge, I am confident in my ability to contribute to innovative IT projects and drive business success.

Overall, my shift to IT is a natural progression of my interests and skills, allowing me to leverage my economics background while embracing the opportunities and challenges of the technology sector.

or

I started with a background in Economics, where I learned a lot about data analysis and problem-solving. When I worked at a bank, I saw how important technology is for making banking easier and more efficient. This experience made me interested in IT.

To gain the technical skills needed for a career in IT, I decided to pursue an MCA at Techno India University. Now, I have learned programming languages like Python and skills in SQL and HTML.

My economics background helps me understand data and business processes, which is very useful in IT. By combining my knowledge of economics and my new IT skills, I can create effective technology solutions.

**Question: Why are you interested in this IT job at our company, TCS?**

**Answer:**

I am interested in this IT job at TCS because TCS is known for its innovative solutions and its strong presence in the global IT industry. I am impressed by TCS’s commitment to technology-driven transformation and its impact on various industries.

My background in economics and my technical skills in Python, SQL, and HTML align well with the projects TCS handles. I am particularly excited about the opportunity to work on cutting-edge technologies and contribute to meaningful projects that can make a real difference.

Additionally, TCS’s emphasis on continuous learning and professional growth is very appealing to me. I am eager to develop my skills further and be part of a dynamic team that encourages innovation and excellence

Or

I am interested in this IT job at TCS because TCS is a leading company known for its innovative technology solutions. I have the skills in Python, SQL, and HTML that match well with the kind of work TCS does.

I am excited about the chance to work on new and exciting projects and to be part of a team that makes a real impact. I also like that TCS supports learning and growth, and I want to keep improving my skills and contribute to the company's success.

"One of my weaknesses is that I find it difficult to say no to requests, which sometimes leads to me taking on too much work. To address this, I am learning to set clearer boundaries and prioritize tasks better. I am working on assessing my workload realistically and communicating more openly with my team to manage my commitments effectively."

"One of my key strengths is my ability to remain calm and patient in challenging situations, particularly when dealing with aggressive customers. During my time working at a bank, I frequently encountered customers who were upset about not receiving the service they expected. In these situations, I consistently maintained my composure and handled their concerns with patience and empathy, ensuring that their issues were resolved effectively and professionally. This experience has honed my skills in conflict resolution and customer service, making me adept at managing stressful interactions with a positive and constructive approach."

"My strength is my ability to multitask effectively. During my school days, I actively participated in extracurricular activities, including being a member of the kho-kho team. This experience helped me develop strong time management and teamwork skills, which have been invaluable in balancing multiple tasks and responsibilities efficiently."

Hard work gives you experience,  
Smart work comes with experience,  
  
There is no smart work without hard work.

. Is Python a compiled language or an interpreted language?

ChatGPT

Python is an interpreted language. When you run a Python script, the Python interpreter reads and executes the code line by line, translating it into machine-readable bytecode on-the-fly. This differs from compiled languages like C or C++, where the code is translated into machine-readable binary before execution. In Python, the source code is directly executed by the interpreter, which makes it easier to write and test code quickly without the need for a separate compilation step.

I didn't choose Python personally, but Python is a popular language in part because of its simplicity and readability. It's often praised for its clean syntax, which makes it easier to write and understand code, especially for beginners. Python's extensive libraries and frameworks also contribute to its popularity, enabling developers to accomplish a wide range of tasks without reinventing the wheel.

Python differs from other languages in several ways:

1. \*\*Readability\*\*: Python emphasizes readability, using indentation and whitespace to define code blocks, which makes it easier to understand compared to languages with more complex syntax.

2. \*\*Versatility\*\*: Python is a versatile language that can be used for various purposes, including web development, data analysis, artificial intelligence, scientific computing, and more.

3. \*\*Interpreted\*\*: Python is an interpreted language, meaning that code is executed line by line, which can make development and debugging faster compared to compiled languages.

4. \*\*Dynamic Typing\*\*: Python uses dynamic typing, allowing variables to be assigned without declaring their type explicitly. This can lead to more flexible code but may also introduce potential errors if not handled carefully.

5. \*\*Large Ecosystem\*\*: Python has a vast ecosystem of libraries and frameworks that facilitate various tasks, from web development with Django or Flask to data analysis with pandas and machine learning with TensorFlow or PyTorch.

6. \*\*Community Support\*\*: Python has a large and active community of developers who contribute to its development, create open-source projects, and provide support through forums, tutorials, and other resources.

Overall, Python's simplicity, versatility, and extensive ecosystem make it a popular choice for both beginners and experienced developers alike.

What is a string in Python?

String is a collection of alphabets, words or other characters. It is one of the primitive data structures and are the building blocks for data manipulation. Python has a built-in string class named str . Python strings are "immutable" which means they cannot be changed after they are created.

A shallow copy creates a new array, but it does not create new copies of the elements within the array. Instead, it points to the same elements as the original array. A deep copy, on the other hand, creates a completely independent copy of both the array and its data. It does not share any data with the original array.

Python is an interpreted language, meaning it's not directly compiled into machine code. Instead, Python code is executed line by line by the Python interpreter. This interpreter reads, parses, and executes code in real-time, converting it into bytecode. This bytecode is then run by the Python Virtual Machine (PVM), which translates it into machine code and manages memory for efficient, platform-independent execution.

**.Dynamic Typing:** One of Python’s distinctive features is its dynamic typing system. In Python, you don’t need to declare the data type of a variable explicitly. The interpreter determines the type of a variable dynamically during runtime. This flexibility allows for more concise and readable code but requires careful attention to variable types to avoid unexpected behavior.

**5. Memory Management:**Python manages memory automatically through a mechanism called garbage collection. When an object is no longer in use, the garbage collector identifies and frees up the memory occupied by that object. This helps developers focus on writing code without worrying too much about memory management, making Python an approachable language for programmers of various skill levels.

The python id() function is used to return a unique identification value of the object stored in the memory.

What is a static and non static method in Python?

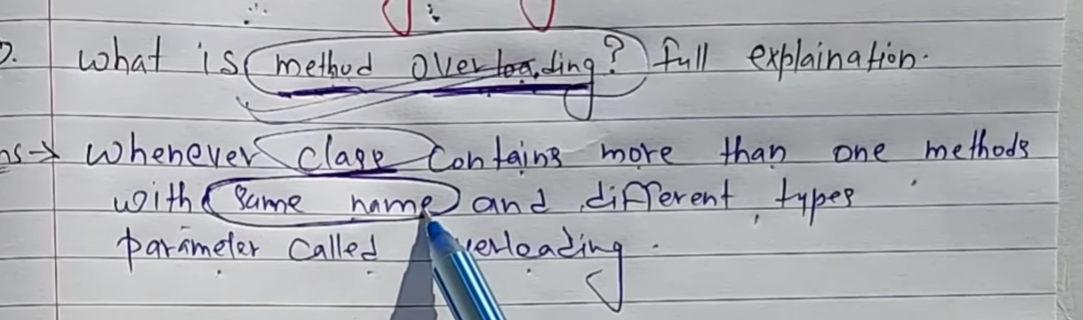
While a static method requires no specific parameters, a class method takes cls as its first argument. While a static method cannot access or modify the class state, a class method can. Static methods are typically unaware of the class state.

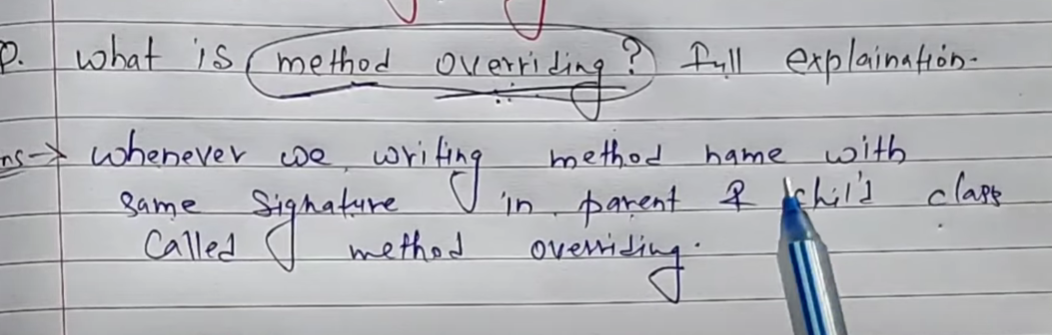
What is static vs non-static variables Python?

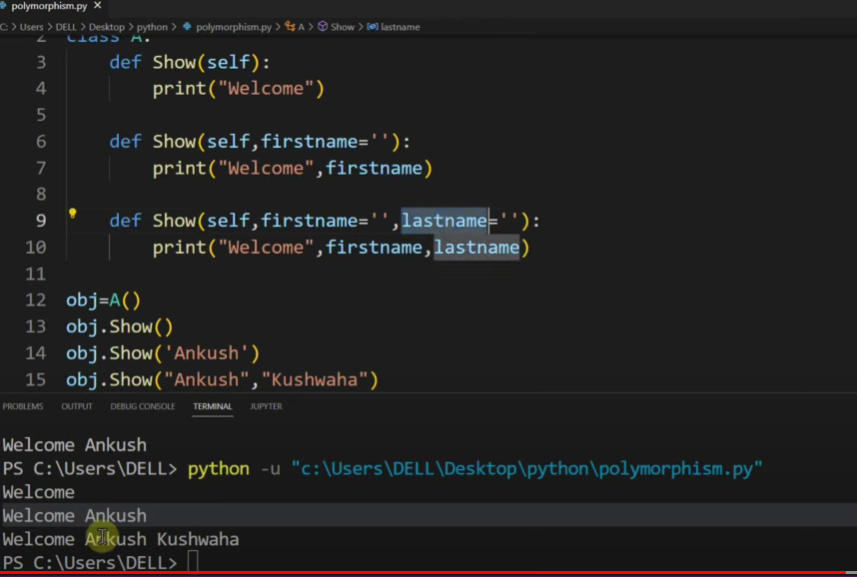
In Python programming, a static variable is said to be a class variable that is 2common to all class members. A static variable is declared within that class but outside the methods within the given class. Non-static variables are those variables that are associated with the objects or methods declared within the class.

What is pointer with example?

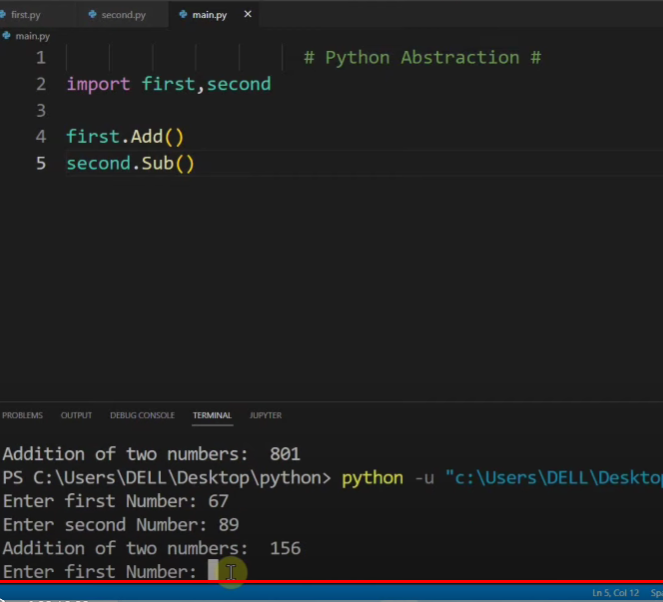
A pointer is a variable that stores the address of another variable. Unlike other variables that hold values of a certain type, pointer holds the address of a variable. For example, an integer variable holds (or you can say stores) an integer value, however an integer pointer holds the address of a integer variable.

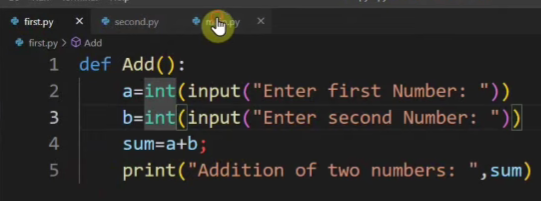


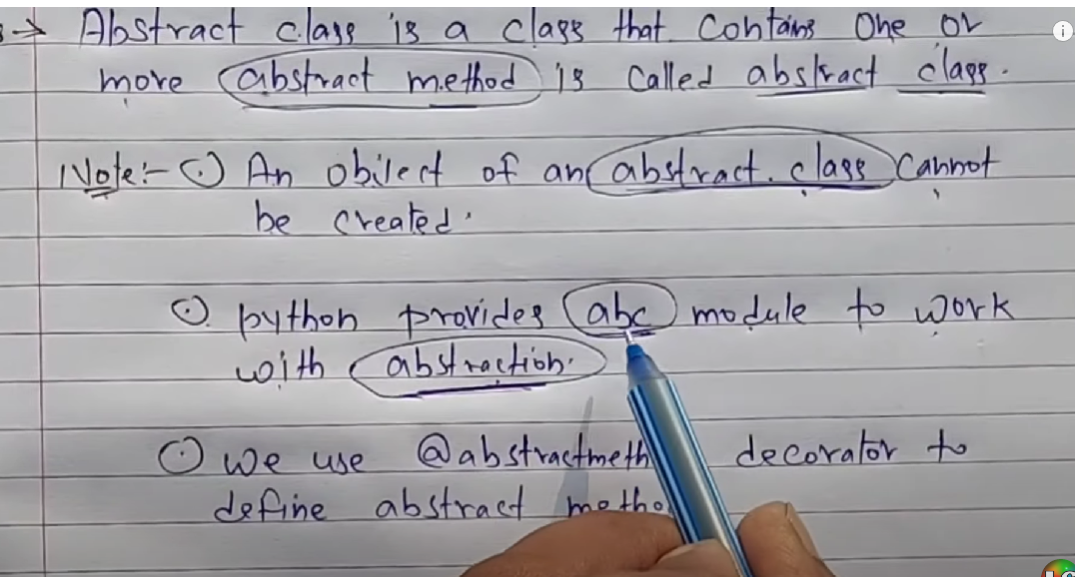


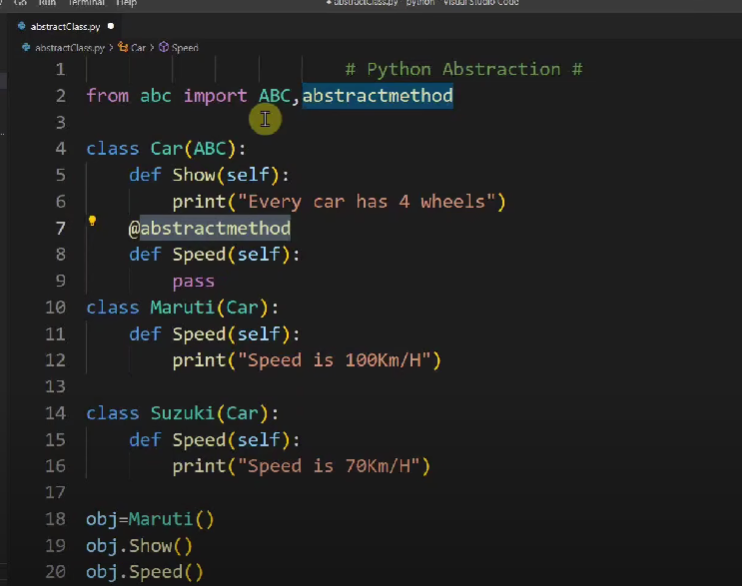




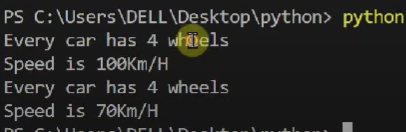


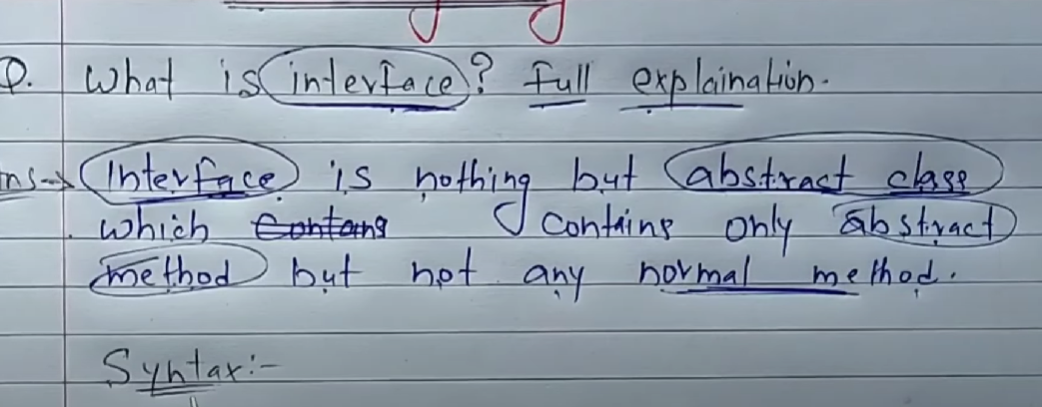


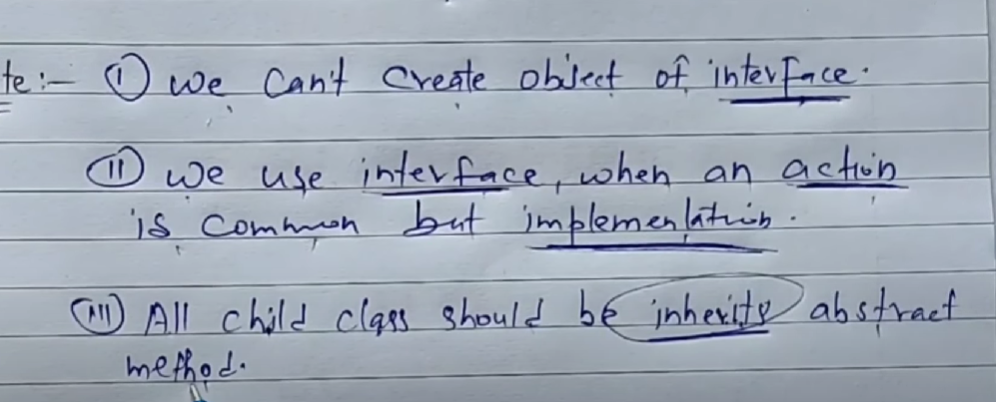


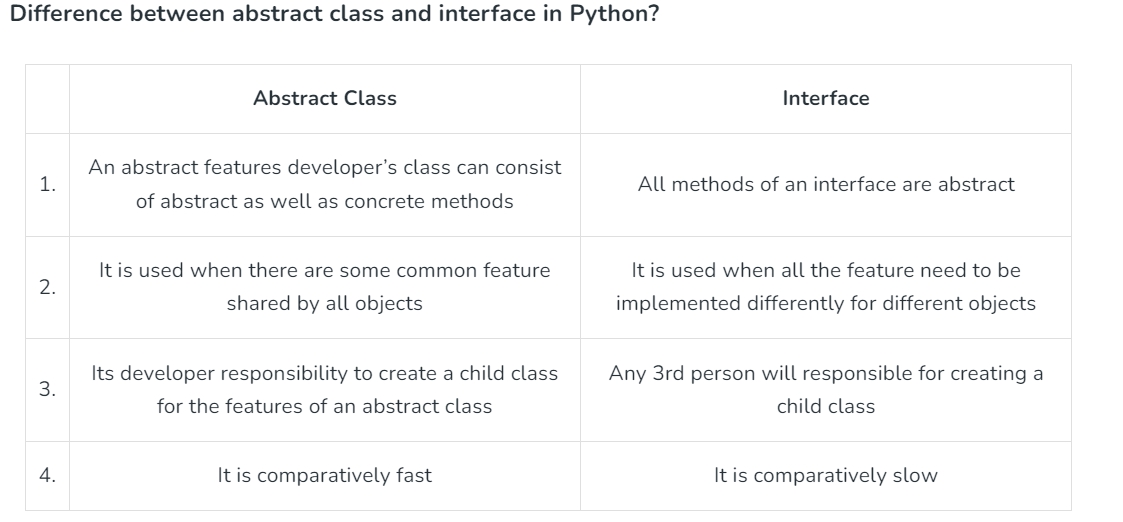






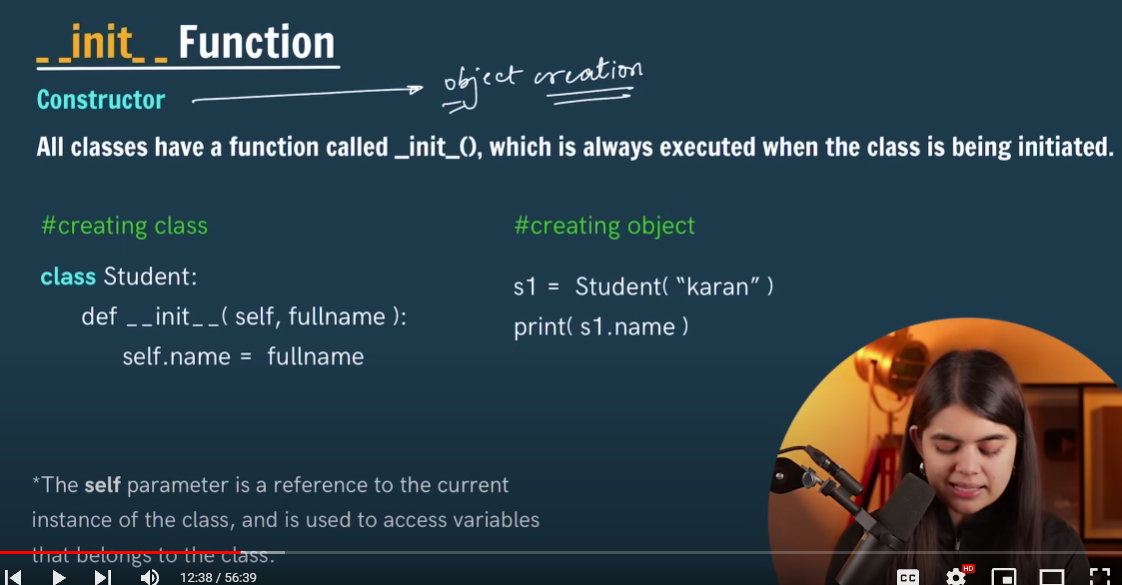






What are the differences between Classmethod and StaticMehtod?

|  |  |
| --- | --- |
| **Class Method** | **Static Method** |
| The class method takes cls (class) as first argument. | The static method does not take any specific parameter. |
| Class method can access and modify the class state. | Static Method cannot access or modify the class state. |
| The class method takes the class as parameter to know about the state of that class. | Static methods do not know about class state. These methods are used to do some utility tasks by taking some parameters. |
| @classmethod decorator is used here. | @staticmethod decorator is used here. |

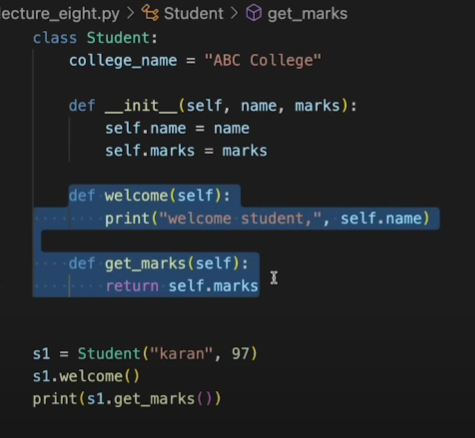


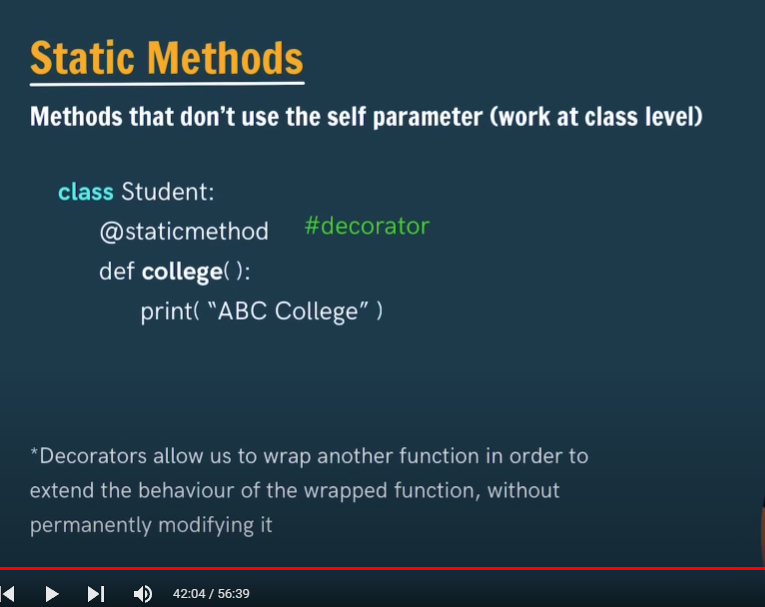
In Python programming, a static variable is said to be a class variable that is common to all class members. A static variable is declared within that class but outside the methods within the given class. Non-static variables are those variables that are associated with the objects or methods declared within the class

What is the difference between static and non-static variables?

Static variables are shared among all instances of a class. Non static variables are specific to that instance of a class. Static variable is like a global variable and is available to all methods. Non static variable is like a local variable and they can be accessed through only instance of a class.

The three different types of variables in OOP in Python are: Instance Variables: Variables that are defined inside \_\_init\_\_ are called instance variables and these are of object level. Class/Static Variables: Variables that are defined outside the \_\_init\_\_ method are called class variables and these are of class level.24 Aug 2021





## ****Static Methods****

Static methods do not take the self or “cls” parameter and cannot access the instance, class variables, or methods.

### **Characteristics**

* Static methods are independent of instances of the class and the class itself. They don’t take any implicit first parameter like self or cls.
* Static methods can’t access class-level data or instance-level data. They work like normal functions outside of the class and have no access to data except the data passed as arguments.
* To define a static method, you must use the @staticmethod decorator before the method definition.
* Static methods are often utilised for utility functions that don’t depend on any data of the class or instance. They can be used to perform calculations or operations related to the class but not dependent on it.
* [**Types of methods in Python**](https://www.shiksha.com/online-courses/articles/understanding-python-methods/#Types-of-Methods-in-Python-)
  + [Instance Methods](https://www.shiksha.com/online-courses/articles/understanding-python-methods/#Instance-Methods-)
  + [Class Methods](https://www.shiksha.com/online-courses/articles/understanding-python-methods/#Class-Methods-)
  + [Static Methods](https://www.shiksha.com/online-courses/articles/understanding-python-methods/#Static-Methods-)
* [**Comparison between Python Methods**](https://www.shiksha.com/online-courses/articles/understanding-python-methods/#Comparison-between-Python-Methods)

## ****Types of Methods in Python****

In Python, there are three types of methods: instance, class, and static.

1. Instance methods are associated with an instance of a class and can modify the data stored within the instance.
2. Class methods are associated with the class rather than an instance and can access and modify class-level data.
3. Static methods are similar to functions outside of the class and cannot access any instance or class data.

## ****Instance Methods****

**Characteristics:**

* Instance methods are defined within the class definition and are called on an instance of the class using the dot notation.
* “Self” is often the first parameter of an instance method.
* They can access and modify the attributes of the instance on which they are called.
* They can also access and call other instance and class methods of the same class.

## ****Class Methods****

Class methods operate on the class itself rather than on an instance. Class methods are defined within the class definition and are called on the class rather than on an instance.

### **Characteristics**

* Python class methods are associated with the class and are defined using the @classmethod decorator.
* They are conventionally named “cls” and have access to class-level data.
* Class methods are frequently used to create alternative constructors that allow for easier creation of class instances with attributes set differently.
* Class methods have access to class-level variables but not instance-level variables. However, because they are associated with the class, they are often used to modify class-level data.

### **Examples of class methods**

### What is an Exception?

An **exception** is an event that occurs during the execution of a program that disrupts the normal flow of instructions. Exceptions are typically errors that occur due to various reasons, such as invalid user input, file not found, network issues, division by zero, etc. Exceptions indicate that something has gone wrong in the program.

### What is Exception Handling?

**Exception handling** is a mechanism to handle runtime errors so that the normal flow of the program can be maintained. It allows a program to deal with unexpected situations and continue functioning or gracefully terminate instead of crashing abruptly.

### Procedural Programming vs Object-Oriented Programming

Below are some of the differences between procedural and object-oriented programming:

| **Procedural Oriented Programming** | **Object-Oriented Programming** |
| --- | --- |
| In procedural programming, the program is divided into small parts called ***functions***. | In object-oriented programming, the program is divided into small parts called ***objects***. |
| Procedural programming follows a ***top-down approach***. | Object-oriented programming follows a ***bottom-up approach***. |
| There is no access specifier in procedural programming. | Object-oriented programming has access specifiers like private, public, protected, etc. |
| Adding new data and functions is not easy. | Adding new data and function is easy. |
| Procedural programming does not have any proper way of hiding data so it is ***less secure***. | Object-oriented programming provides data hiding so it is ***more secure***. |
| In procedural programming, overloading is not possible. | Overloading is possible in object-oriented programming. |
| In procedural programming, there is no concept of data hiding and inheritance. | In object-oriented programming, the concept of data hiding and inheritance is used. |
| In procedural programming, the function is more important than the data. | In object-oriented programming, data is more important than function. |
| Procedural programming is based on the ***unreal world***. | Object-oriented programming is based on the ***real world***. |
| Procedural programming is used for designing medium-sized programs. | Object-oriented programming is used for designing large and complex programs. |
| Procedural programming uses the concept of procedure abstraction. | Object-oriented programming uses the concept of data abstraction. |
| Code reusability absent in procedural programming. | Code reusability present in object-oriented programming. |
| **Examples:** C, FORTRAN, Pascal, Basic, etc. | **Examples:**C++, Java, Python, C#, etc. |

| **Functional Programming** | **Object Oriented Programming** |
| --- | --- |
| This programming paradigm emphasizes on the use of functions where each function performs a specific task. | This programming paradigm is based on object oriented concept. Classes are used where instance of objects are created |
| Fundamental elements used are variables and functions.The data in the functions are immutable(cannot be changed after creation). | Fundamental elements used are objects and methods and the data used here are mutable data. |
| Importance is not given to data but to functions. | Importance is given to data rather than procedures. |
| It follows declarative programming model. | It follows imperative programming model. |
| It uses recursion for iteration. | It uses loops for iteration. |
| It is parallel programming supported. | It does not support parallel programming. |
| The statements in this programming paradigm does not need to follow a particular order while execution. | The statements in this programming paradigm need to follow an order i.e., bottom up approach while execution. |
| Does not have any access specifier. | Has three access specifiers namely, Public, Private and Protected. |
| To add new data and functions is not so easy. | Provides an easy way to add new data and functions. |
| No data hiding is possible. Hence, Security is not possible. | Provides data hiding. Hence, secured programs are p |

File handling in Python refers to the process of reading from and writing to files on the filesystem. It allows Python programs to interact with files to store data, retrieve data, or perform other operations. Here's a short overview of file handling in Python:

What is a file handle in Python?

File handling is an integral part of programming. File handling in Python is simplified with built-in methods, which include creating, opening, and closing files. While files are open, Python additionally allows performing various file operations, such as reading, writing, and appending information.

File handling refers to the process of creating, opening, reading, writing, and closing files in a computer program. It allows you to store data permanently on a disk, manipulate that data, and retrieve it when needed.

1. \*\*Opening a File:\*\* Use the `open()` function to open a file. You specify the file path and mode (e.g., read mode, write mode, append mode).

```python

file = open('filename.txt', 'r') # Open file in read mode

```

2. \*\*Reading from a File:\*\* Use methods like `read()`, `readline()`, or `readlines()` to read data from the file.

```python

content = file.read() # Read entire file content

```

3. \*\*Writing to a File:\*\* Use methods like `write()` to write data to a file. Remember to open the file in write mode (`'w'`) or append mode (`'a'`).

```python

file.write('Hello, world!\n') # Write a line to the file

```

4. \*\*Closing a File:\*\* Always close the file after performing operations using the `close()` method. This ensures that resources are properly released.

```python

file.close() # Close the file

```

5. \*\*Using `with` Statement:\*\* It's recommended to use the `with` statement, which automatically closes the file when the block is exited, ensuring proper resource management.

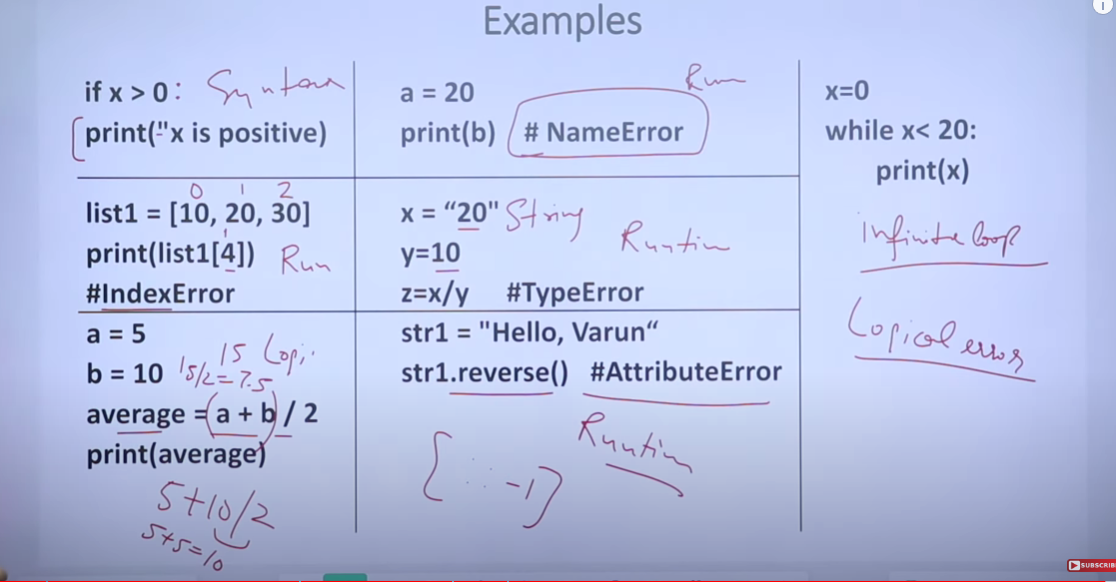
```python

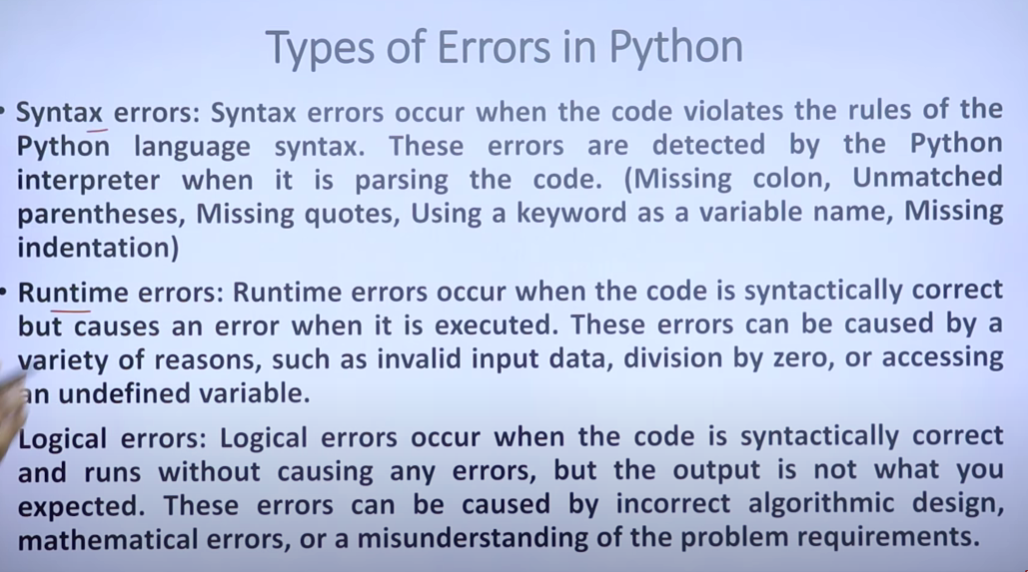
with open('filename.txt', 'r') as file:

content = file.read()

```

File handling in Python is versatile and allows you to work with various file formats, such as text files, CSV files, JSON files, and more. Always ensure proper error handling and resource management when working with files to avoid issues like resource leaks or data corruption.





Asymptotic notation is a mathematical concept used in computer science to describe the behavior of algorithms as the input size grows. It provides a way to analyze and compare the efficiency of algorithms, particularly in terms of time and space complexity. The three most common types of asymptotic notation are Big O (O), Omega (Ω), and Theta (Θ). Each of these notations provides a different perspective on the algorithm's performance:

Asymptotic notation is a mathematical concept used in computer science to describe the behavior of algorithms as the input size grows. It provides a way to analyze and compare the efficiency of algorithms, particularly in terms of time and space complexity. The three most common types of asymptotic notation are Big O (O), Omega (Ω), and Theta (Θ). Each of these notations provides a different perspective on the algorithm's performance:

1. \*\*Big O Notation (O)\*\*:

- Describes the upper bound of an algorithm's running time.

- Represents the worst-case scenario.

- Indicates the maximum amount of time an algorithm can take to complete.

- Example: O(n) means the algorithm's running time grows linearly with the input size.

2. \*\*Omega Notation (Ω)\*\*:

- Describes the lower bound of an algorithm's running time.

- Represents the best-case scenario.

- Indicates the minimum amount of time an algorithm can take to complete.

- Example: Ω(n) means the algorithm's running time grows at least linearly with the input size.

3. \*\*Theta Notation (Θ)\*\*:

- Describes the tight bound of an algorithm's running time.

- Represents both the upper and lower bounds.

- Indicates the exact growth rate of the algorithm's running time.

- Example: Θ(n) means the algorithm's running time grows exactly linearly with the input size.

### Examples

1. \*\*Constant Time - O(1)\*\*:

- The algorithm's running time does not change with the input size.

- Example: Accessing a specific element in an array.

2. \*\*Linear Time - O(n)\*\*:

- The running time increases linearly with the input size.

- Example: Finding an element in an unsorted list.

3. \*\*Quadratic Time - O(n^2)\*\*:

- The running time increases quadratically with the input size.

- Example: Bubble sort, which involves nested loops.

4. \*\*Logarithmic Time - O(log n)\*\*:

- The running time increases logarithmically with the input size.

- Example: Binary search in a sorted array.

### Practical Use

- \*\*Algorithm Comparison\*\*: Asymptotic notation helps in comparing the efficiency of different algorithms, especially for large input sizes.

- \*\*Predicting Performance\*\*: It provides insights into how an algorithm will perform as the input size scales.

- \*\*Optimization\*\*: Identifying the bottlenecks and potential inefficiencies in algorithms.

### Limitations

- \*\*Constant Factors Ignored\*\*: Asymptotic notation does not consider constant factors and lower-order terms, which might be significant for small input sizes.

- \*\*Real-world Performance\*\*: It might not accurately predict performance in real-world scenarios where other factors like hardware and system architecture play a role.

In summary, asymptotic notation is a fundamental tool in computer science for evaluating and comparing the efficiency of algorithms, focusing on their behavior as the input size becomes large.