MERITSHOT

1. Real world use cases of Python

Real World Use cases of Python

- Python is a versatile and powerful programming language,
    - gained immense popularity in various industries.

Real-world use cases to understand its importance:

- ***Web Development:***
- Python is widely used for web development,
    - Frameworks like Django and Flask.
- Frameworks provide a robust and efficient way to build websites and web applications.

Real World Use cases of Python

- ***Data Science and Machine Learning:***
- Python has become the go-to language for data scientists and machine learning practitioners.
- Libraries like NumPy, Pandas, and Scikit-learn offer powerful tools,
    - data manipulation, analysis, and building machine learning models.

Real World Use cases of Python

- ***Automation and Scripting:***
- Python's simplicity and readability for automating repetitive tasks and writing scripts.
- Used for tasks such as
  - file manipulation, data scraping, and system administration.

Real World Use cases of Python

- ***Scientific Computing:***
- Python provides a rich ecosystem of libraries for scientific computing,
  - such as SciPy and Matplotlib.
- Scientists and researchers use Python
  - Simulations, numerical computations, and visualizations.

Real World Use cases of Python

- ***Internet of Things (IoT):***
- Python's lightweight nature and support for microcontrollers,
  - Preferred language for IoT projects.
- Enables developers to connect devices, collect data, and control IoT systems.
- ***Game Development:***
- Python's simplicity and popular frameworks like Pygame.
- Create 2D and 3D games, prototypes, and simulations using Python.

Why is it Important to Learn Python as Beginner

Easy to Learn:

- Python has a clean and readable syntax.
- Allows beginners to grasp programming concepts and start building applications.

Versatility:

- Used for a wide range of tasks, from web development to data analysis and machine learning.
- Learning Python opens up diverse career opportunities.

Why is it Important to Learn Python as Beginner

Community and Resources:

- Vast community of developers.
- Online resources, tutorials, and libraries available.

Industry Demand:

- Python is in high demand across industries. Its popularity in data science, web development, and automation makes it a valuable skill for job seekers.

Why is it Important to Learn Python as Beginner

Future-proof:

- Expected to grow in the coming years.
- Its adoption in emerging technologies,
  - AI, IoT, and robotics.

MERITSHOT

4. Introduction to Data Types and Type Casting

Data Types

- Three major data types:
  - Integer
  - Float, and
  - String.
- Integer is a Data Type,
  - which Represents Whole Numbers.
  - ***For Example***: 8, 45, 965, 2 etc.
- All the Variables having values in whole Numbers are Integers.

Data Types

- Float is a Data Type,
  - which Represents Decimals,
  - ***For Example:*** 2.5, 0.6356, 85.1 etc.
- All the Variables having decimal values are categorized as Float Data Type.
- String is a Data Type,
  - which Represents Text.
  - ***For Example:*** “Ram”, “Car”, “Flower”.
- All the Variables having Values as Text will be Categorized as Strings.

Inbuilt Functions

- Function which is already defined in a program or programming framework with a set of statements,
  - which together performs a task
  - A built-in function.
- Can use it directly in their program or application.

Type Casting

- Process of converting the data type of a variable or value from one type to another.
- Allow you to perform type casting.
- Python again has in-built functions.
- ***Three typecasting functions are:***
  - int()
  - float()
  - str()

Type Casting

int() function:

- Used to convert a string or float value to an integer value.

float() function:

- Used to convert an integer or string value to a float value.

str() function:

- Used to convert an integer or float value to a string value.

5. Scope of Variables

Scope of Variables

- Part of a program where a variable is accessible,
  - Scope.
- Scope is the visibility of a variable declared within a specific region.
- ***There are two types of scopes:***
  - Local Scope.
  - Global Scope.

MERITSHOT

6. Introduction to Operators

Operators

- Operators are functions that perform
  - Arithmetic,
  - Assignment,
  - Logical and other types of computation.
- ***Total of Seven different types of Operators:***
- Operators are special symbols or characters
  - to perform operations on operands.
- Mathematical, logical, comparison, assignment, and other operations.
- Values that the operators operate on are called operands.

Operators

Arithmetic Operators:

- Arithmetic operators are used in Python to perform basic mathematical calculations.
- Operate on numerical values and return a new value as a result.
- Arithmetic Operators helps us to perform Operations such as
  - Addition, Subtraction, Multiplication, Division, Floor Division, Exponentiation.

MERITSHOT

6. Introduction to Searching Algorithms

Searching Algorithms

- Find the presence or position of a specific element within a collection of data.
- Efficiently locating desired items within a given dataset or data structure.

Searching Algorithms

Efficient Information Retrieval:

- Locate specific data elements,
    - where large amounts of data need to be searched or accessed frequently.
- ***For example:***
- Searching algorithms are used in databases, search engines, and information retrieval systems to efficiently retrieve relevant information.

Searching Algorithms

MERITSHOT

Data Validation:

- To check if a particular element exists within a dataset.
- For verifying user input or validating data integrity.
- ***For example:***
- When building forms or handling user interactions, searching algorithms help validate input values.

Searching Algorithms

Sorting and Ordering:

- Binary search, require the data to be sorted.
- Organizing data in a particular order, enabling efficient searching operations.
- Sorting is fundamental for various applications,
  - data analysis, data processing, and optimizing search efficiency.

Searching Algorithms

Decision Making:

- Searching algorithms can also be used to make decisions based on the presence or absence of specific elements in a dataset.
- Implementing conditional logic or branching in a program.
- ***For example:*** Determining if a username already exists in a database before allowing registration.

Searching Algorithms

- Searching algorithms in Python play a significant role in efficiently
    - Locating,
    - validating, and processing data elements within collections.
    - Leading to improved information retrieval, decision-making, and data manipulation capabilities.

1. Introduction to Parameters and Arguments

Functions in Python

- Function is a group of related statements,
  - performs a specific task only runs when it is called.
- Help us to break our program into smaller and modular chunks.
- As our program grows larger and larger,
  - Functions make it more organized and manageable.

Create a Function

- Define a function using the ‘def’ keyword.
- Need to allot a unique name to the function name
  - to uniquely identify the function.
- Function naming follows the same rules of writing identifiers in Python.
- Parameters and arguments in a function are optional.
- Colon **(:)** is used to mark the end of the function header.

Create a Function

- One or more valid python statements,
  - make up the function body.
- Statements must have the same indentation level.
- An optional return statement to return a value from the function.

1. Introduction to Parameters and Arguments

Create a Function

- One or more valid python statements,
  - make up the function body.
- Statements must have the same indentation level.
- An optional return statement to return a value from the function.

4. Introduction to Lambda Functions

Lambda Functions

- Lambda functions also known as anonymous functions.
- A way to create small,
  - one-line functions without a formal name.
- Defined using the lambda keyword,
  - used for simple, short-lived functions.

Lambda Functions

Lambda functions are useful in Python:

- ***Concise and Readable Code:***
- Lambda functions allow you to write functions in a compact manner,
  - reducing need for defining a separate function with a name.
- Make the code more concise and easier to read,
  - for simple operations.

Lambda Functions

- ***Functional Programming:***
- Used in functional programming paradigms,
    - where functions are treated as first-class objects.
- Can be passed as arguments to other functions,
    - returned from other functions,
    - or stored in data structures.
- Use of higher-order functions and supports functional programming techniques.

Lambda Functions

- ***Inline Function Definitions:***
- Defined directly within the context.
- Eliminates the need for defining functions separately.
- Helps in situations where a function is required for a specific purpose,
  - but does not need to be reused elsewhere in code.

Lambda Functions

- ***Callback Functions:***
- Lambda functions are often used as callback functions,
  - functions that are passed as arguments to other functions,
  - called later in the program's execution.
- Define small, on-the-fly functions,
  - used for specific tasks within a larger program.

Lambda Functions

- ***Sorting and Filtering:***
- Lambda functions are commonly used in conjunction with built-in functions
  - sorted() and filter().
- Allow for custom sorting criteria or filtering conditions
  - to be defined inline without the need for writing a separate named function.

Lambda Functions

- Flexibility and convenience in writing short,
  - one-off functions without the need for defining formal function names.
- Useful in situations where small, temporary functions are needed for specific operations,
  - working with higher-order functions and functional programming concepts.

2. Introduction to Inheritance

Inheritance

- Inheritance is a fundamental concept in object-oriented programming (OOP).
- Allows you to create new classes based on existing classes.
- Derived classes inherit attributes and methods from the base class,
  - enables code reuse, and
  - promotes a hierarchical organization of code.

Inheritance

Syntax:

```
class BaseClass:

    # Attributes and methods of the base class

class DerivedClass(BaseClass):

    # Additional attributes and methods specific to derived class
```

Uses of Inheritance

Code Reusability:

- Inheritance promotes code reuse by allowing derived classes to inherit attributes and methods.
- Don't have to rewrite the same code in multiple classes,
  - resulting in more efficient and maintainable code.

Uses of Inheritance

Hierarchy and Organization:

- Inheritance facilitates the creation of a class hierarchy,
    - where classes are organized based on their relationships.
- A base class representing a general concept.
- Create more specialized derived classes that inherit from it.
- ***For example:*** Have a base class called Vehicle, and
    - derived classes like Car, Bicycle, and Motorcycle.

Uses of Inheritance

Overriding and Extension:

- Derived classes have the ability to override methods and attributes inherited from the base class.
- Can redefine or customize the behavior of specific methods in the derived class.
- Derived classes can also add new attributes and methods,
    - to extend the functionality inherited from the base class.

MERITSHOT

4. Introduction to Polymorphism

Polymorphism

- Polymorphism is a fundamental concept in object-oriented programming (OOP).
  - allows objects of different classes to be treated as objects of a common superclass.
- Enables methods of different classes to have the same name
  - but different implementations.

Two Types of Polymorphism

Compile-time polymorphism:

- Method overloading.
- Compile-time polymorphism allows multiple methods with the same name,
  - but different parameters to coexist in a class.
- Appropriate method to execute is determined by the number, type, and order of the arguments,
  - provided during the method call.
- Resolved by the compiler at compile-time.

Two Types of Polymorphism

Runtime polymorphism:

- Method overriding.
- Runtime polymorphism occurs when a subclass,
  - its own implementation of a method that is already defined in its superclass.
- Method in the subclass must have the same name, return type, and parameters as the method in superclass.
- Method to execute is made at runtime, based on the actual type of the object.

Polymorphism Uses

Code reusability:

- Polymorphism promotes code reuse by allowing different objects to be treated uniformly.
- Can create generic code that operates on a superclass, and
- Used with any subclass instances, without needing to modify the code.

Polymorphism Uses

Flexibility and extensibility:

- Polymorphism allows you to introduce new classes that inherit from a common superclass
  - provide their own specialized implementations of methods.
- Add new functionality to your program without modifying existing code.

Polymorphism Uses

Simplifying complex code:

- Polymorphism helps simplify complex code by abstracting away the specific implementation details of objects.
- Can work with objects at a higher level of abstraction,
- Focusing on their common behaviors defined in the superclass.

Polymorphism Uses

- ***Polymorphic method calls:***
- Write code that calls methods on objects without knowing their specific types.
- More modular and flexible code, as you can pass different objects to the same method, and
- Get the appropriate behavior based on their actual types.