Programming Terms: A to Z Glossary

We will use many programming terms in the course and explain them accordingly. Below is a glossary of all the terms. You don't need to read the glossary now, but you can refer to it whenever you want to know what a term means:

A

- **Algorithm**: A step-by-step procedure for solving a problem or performing a task.
- **Argument**: A value passed to a function when calling it.
  - *Example*: `print("Hello, World!")`

    The part `"Hello, World!"` is an argument.
- **Array**: The term that is equivalent to a Python list, but used in other languages such as Javascript. See "list" to learn what a Python list is.
- **Assignment**: Assigning a value to a variable.
  - *Example*: `x = 10`

    `x` is assigned the value `10`.

B

- **Boolean**: A data type representing `True` or `False`.
  - *Example*: `is_valid = True`
- **Break**: A statement that exits a loop.
  - *Example*:

    ```
    for i in range(10):
        if i == 5:
            break
    ```
- **Bug**: An error, flaw, or unintended behavior in a program that causes it to produce incorrect or unexpected results. Debugging is the process of

finding and fixing these issues.

- *Example*: If a program crashes when it tries to divide by zero, that is considered a bug.

C

- **Class**: A blueprint for creating objects that encapsulate data and methods.
    - *Example*:

    ```
    class Dog:
        def __init__(self, name):
            self.name = name
    ```

- **Code**: The set of instructions written by a programmer to be executed by a computer.
    - *Example*: `print("Hello, World!")` is a line of code in Python.
- **Command Line**: A text interface used to enter commands to the operating system or software. It often refers to the prompt where users type commands to perform specific tasks.
    - *Example*: In a command line interface, you might type `python script.py` to run a Python script. The term is used interchangeably with terminal and console.
- **Comment**: A line ignored by the Python interpreter, used to explain code.
    - *Example*: `# This is a comment`
- **Conditional**: A statement that executes code only if a condition is true.
    - *Example*:

    ```
    if x > 5:
        print("x is greater than 5")
    ```

- **Console**: A term often used interchangeably with terminal and command line; it refers to the interface for entering commands or viewing output from programs. It may also refer to the area in an Integrated Development Environment (IDE) where output is displayed.

    - *Example*: The console in an IDE shows the results of print statements and error messages.
- **CSV (Comma Separated Values)**: A file format where data is stored in rows and columns, with values separated by commas.
    - *Example*: A file with rows like `name,age` and `John,30` is in .csv file.

D

- **Data Type**: A classification that specifies the type of data a variable can hold, such as integers, strings, lists, etc. Each data type determines what operations can be performed on the data.
    - *Example*:

    ```
    age = 25  # Integer data type
    name = "Alice"  # String data type
    numbers = [1, 2, 3]  # List data type
    ```

- **Def (Function Definition)**: A keyword that denotes the start of a function definition.
    - *Example*:

    ```
    def greet():
        print("Hello")
    ```

- **Dependency**: A package or library that a program requires to run. Managing dependencies is crucial for ensuring that a program works correctly with the required versions of libraries.
    - *Example*: If you use the `requests` library in your Python project, it becomes a dependency that must be installed.

      The terms library and package are used interchangeably with package.
- **Dictionary**: A collection of key-value pairs.
    - *Example*: `person = {"name": "John", "age": 30}`

E

- **Exception**: An error that occurs during program execution.

    - *Example*:

      ```
      "Hello" + 10
      ```

      The above would throw an exception, also known as error, because adding a number to a string is not possible.

- **Execute**: To run a program or a block of code.
    - *Example*: When you press "Run" in your IDE, it executes your code.
- **Expression**: Any valid combination of variables, operators, and function calls that can be evaluated by the interpreter.
    - *Example*: `3 + 4 * 2`

F

- **File**: A collection of data or information stored on a computer.
    - *Example*: A `.txt` or `.csv` file.
- **For Loop**: A loop that iterates over a sequence.
    - *Example*:

      ```
      for product in ["butter", "milk", "bread"]:
          print(product)
      ```

- **Function**: A block of reusable code that performs a specific task, usually getting some input, processing it, and returning some output.
    - *Example*:

      ```
      def add(a, b):
          return a + b
      ```

G

- **Git**: A version control system that allows developers to track changes in their code, collaborate with others, and manage different versions of their projects.
    - *Example*: You can initialize a new Git repository with the command `git init`.

- **GitHub**: A web-based platform that uses Git for version control and offers collaboration features for software development. It allows developers to host and share code repositories, track issues, and manage project workflows.
    - *Example*: A project can be shared on GitHub by pushing local changes with the command `git push origin main`.

I

- **IDE (Integrated Development Environment)**: A software application that provides tools for writing and testing code.
    - *Example*: PyCharm or VSCode are popular Python IDEs.
- **If Statement**: A conditional that runs code if its condition is true.
    - *Example*:

    ```
    if x == 10:
        print("x is 10")
    ```

- **Import**: Bringing code from an external module or package into your program.
    - *Example*: `import math` brings Python's advanced math functions such as `cos()` into your program, so you can use those functions.
- **Indentation**: The spaces or tabs used at the beginning of a line to define the structure of the code. In Python, indentation is crucial for defining code blocks.
    - *Example*:

    ```
    if True:
        print("This is indented")
    ```

- **Input**: Data provided to a program by the user, usually via the keyboard on a console/terminal.
    - *Example*:

    ```
    name = input("Enter your name: ")
    ```

- **Inheritance**: A key feature of object-oriented programming (OOP) that allows a new class (called a subclass or derived class) to inherit attributes and methods from an existing class (called a superclass or base class). Inheritance promotes code reusability and establishes a hierarchical relationship between classes. Subclasses can override or extend the functionality of the superclass.

  Example:

  ```
  class Animal:  # Superclass
      def speak(self):
          return "Some sound"

  class Dog(Animal):  # Subclass
      def speak(self):  # Overriding method
          return "Woof!"
  ```

  The subclass Dog inherits from its superclass Animal.

- **Interpreter:** A program that reads and executes code line-by-line. Instead of converting the entire code into machine language at once (like a compiler does), it translates and runs each line sequentially. This is why Python, as an interpreted language, allows you to run code immediately without needing to compile it first. The program you download from python.org is an interpreter.

- **Iteration**: Repeatedly executing a block of code, typically using a loop.
  - *Example*:

    ```
    for i in range(3):
        print(i)
    ```

J

- **JSON (JavaScript Object Notation)**: A format for storing and exchanging data, similar to dictionaries in Python.
  - *Example*: `{"name": "John", "age": 30}` is a JSON object and it is normally stored in a .json file.

L

- **Lambda**: An anonymous function defined using the `lambda` keyword. A lambda function is used when you need a small, simple function that you want to define quickly, usually for a short, one-time use, and in some part of the code where you cannot write multiple lines of code to define a normal function.
    - *Example*: `students_sorted = sorted(students, key=lambda x: x[1])`
- **Library**: A collection of pre-written code that you can use to perform common tasks.
    - *Example*: `math` or `random` are built-in Python libraries. When a library is made of only one file, it is usually referred to as a module.
- **List**: A mutable collection of items in a specific order.
    - *Example*: `fruits = ["apple", "banana", "cherry"]`
- **Loop**: A control structure that repeatedly executes a block of code while a condition is true (includes `for` and `while` loops).
    - *Example*:

      ```
      for i in range(5):
          print(i)
      ```

M

- **Method**: A function that is associated with an object and typically operates on its data.
    - *Example*: `fruits.append("orange")`

      `append()` is a method of the list `fruits`.
- **Module**: A file that contains Python code, such as functions or classes, which can be imported into other Python programs.
    - *Example*: `import random` is used to import the `random` module.]

      When there are multiple files containing related code, it is referred to as a library.

O

- **Object**: An instance of a class.

    - *Example*:

      ```
      my_dog = Dog("Buddy")
      ```

  Here my_dog is an instance of the `Dog` class. Integer `15` is also an instance of the `int` class. String `"Hello"` is an instance of the `str` class, etc.

- **Object-Oriented Programming (OOP)**: A programming paradigm based on the concept of "objects," which can contain data (attributes) and code (methods). OOP emphasizes organizing code into reusable structures, making it easier to manage and scale software applications.
    - *Example*: A simple class definition in Python that illustrates OOP might look like this:

      ```
      class Dog:
          def __init__(self, name):
              self.name = name  # Attribute

          def bark(self):  # Method
              return f"{self.name} says woof!"

      my_dog = Dog("Buddy")
      print(my_dog.bark())  # Output: Buddy says woof!
      ```

- **Operator**: A symbol that performs an operation on variables or values.
    - *Example*: `+`, ` `, ` `, `/` are arithmetic operators.
- **Output**: Data produced by a program, typically displayed on the screen or written to a file.
    - *Example*:

      ```
      print("Hello, World!"
      ```

      The above code will display the output "Hello, World!" in the console.

P

- **Parameter**: A variable listed in a function's definition, used to receive arguments.

    - *Example*:

      ```
      def greet(name):
          print(f"Hello, {name}")
      ```

      `name` is a parameter of the `greet` function.

- **Package**: A bundle of Python modules that can be easily distributed and installed. Packages can include libraries, frameworks, and tools to extend Python's capabilities.
    - *Example*: The `numpy` package is widely used for numerical computations in Python.

  The term library is used interchangeably.

- **Pass**: A statement that does nothing, often used as a placeholder.
    - *Example*:

      ```
      def my_function():
          pass
      ```

- **Pip**: The package installer for Python. It allows users to install and manage additional libraries and dependencies that are not part of the Python standard library.
    - *Example*: To install the `requests` package, you would use:

      ```
      pip install requests
      ```

- **Program**: A set of instructions that a computer can execute to perform a specific task.
    - *Example*: A Python script that adds two numbers is a program.

      Other terms used interchangeably are script and code.

- **Python**: A high-level, interpreted programming language designed for readability and ease of use. It is known for its simple syntax, which allows beginners to quickly learn and start programming. Python is versatile and widely used in various fields such as web development, data analysis, artificial intelligence, scientific computing, automation, and more. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, making it suitable for a wide range of applications.

  - *Example*: A simple Python program that prints "Hello, World!" to the console looks like this:

    ```
    print("Hello, World!")  # Output: Hello, World!
    ```

- **Python Console**: An interactive command-line interface that allows users to execute Python code line-by-line and see immediate results. It's often used for testing snippets of code, experimenting with commands, and debugging.
  - *Example*: You can launch the Python console by typing `python`, `py -3` or `python3` in your terminal. Once in the console, you can enter commands directly, such as:

    ```
    >>> print("Hello, World!")
    Hello, World!
    ```

R

- **Repository**: A storage location for software packages, where files, including source code, documentation, and metadata, are managed using version control systems like Git.
  - *Example*: A Git repository can be hosted on platforms like GitHub, GitLab, or Bitbucket.
- **Requirements File**: A text file (usually named `requirements.txt`) that lists the libraries required for a Python project. It allows for easy installation of all specified packages using pip.
  - *Example*: A `requirements.txt` file might look like this:

    ```
    requests==2.25.1
    numpy>=1.20.0
    ```

    This file can be uploaded in a server where the program will be deployed and all the same libraries can be installed in that server so the program runs the same way as locally.

- **Return**: A statement used to exit a function and optionally send a value back to the caller.
    - *Example*:

      ```
      def add(a, b):
          return a + b
      ```

      In the above example, the `return` statement exits the function and returns the sum of `a` and `b`.

S

- **Script**: A file containing a sequence of Python statements that can be executed.
    - *Example*: `my_script.py` contains Python code that can be run.

      Other terms used interchangeably are program and even code.
- **Statement**: A line of code that performs some action.
    - *Example*: `x = 5`
- **String**: A sequence of characters enclosed in quotes.
    - *Example*: `name = "Alice"`
- **Syntax**: The rules that define the structure of a programming language.
    - *Example*: In Python, proper syntax would be `print("Hello")` instead of `Print "Hello"`.
- **Syntax Error**: An error that occurs when the syntax rules of the language have not been followed.
    - *Example*: ` name = 'Alice`

      Running the code above will produce a syntax error because the syntax rules require string Alice to be inside two quotes (i.e., `'Alice'`)not just one.

T

- **Terminal**: A text-based interface that allows users to interact with the operating system by entering commands. It provides a way to execute programs and perform system tasks. The term is used interchangeably with console and command line.

  - *Example*: On macOS, the Terminal application is used to run shell commands.
- **Tuple**: An immutable collection of items in a specific order.
  - *Example*: `coordinates = (10, 20)`
- **Type**: The category of data a value belongs to, such as integer, string, or list.
  - *Example*: `type(10)` will return `int`, indicating that `10` is an integer.

V

- **Value**: The data stored in a variable or used in an expression.
  - *Example*: In `x = 5`, the value of `x` is `5`.
- **Variable**: A name that refers to a value stored in memory.
  - *Example*: `age = 25`
- **Variable Scope**: The area in a program where a variable can be accessed.
  - *Example*: Variables defined inside a function have local scope and cannot be accessed outside that function.
- **Venv**: A module in Python used to create lightweight, isolated environments for Python projects, known as virtual envirionments. It allows developers to manage dependencies separately for different projects, avoiding conflicts between package versions.
  - *Example*: You can create a new virtual environment by running the following command in your terminal:

    ```
    python -m venv myenv
    ```
- **Version Control**: A system that keeps track of changes to code, often used in collaborative environments.
  - *Example*: Git is a version control system used to manage changes in software projects.

- **Virtual Environment**: An isolated environment in which Python projects can run independently of other projects. This allows developers to manage dependencies and package versions without conflicts. Virtual environments are particularly useful when working on multiple projects that may require different versions of the same libraries.

W

- **While Loop**: A loop that continues as long as a condition is true.
    - *Example*:

      ```
      while x < 5:
          x += 1
      ```

      The above code will run as long as x is less than 5.