Pre-Class Reading for Lecture 1: Introduction to Programming

Learning Objectives:

In the upcoming lecture, you will:

- Grasp the fundamental idea behind programming and how it relates to problem-solving in everyday life.
- Understand the role of algorithms in programming and how they serve as a blueprint for solving problems.
- Set up your Python development environment, including installing an Integrated Development Environment (IDE) and the Python interpreter.
- Learn the basics of Python's syntax and write your first "Hello, World!" program.
- Explore essential programming elements like variables, data types, and functions.
- Begin your journey toward understanding Python as an object-oriented language and what that means for how you approach programming.
- Gain a brief understanding of machine code: how computers store data in the form of 0s and 1s and execute instructions using 0s and 1s.

What Is Programming?

Programming is simply the process of giving instructions to a computer to perform specific tasks. You do this by writing lines of code in a programming language, like Python. Just like a recipe helps you bake a cake, a program helps the computer follow precise steps to achieve a goal.

Key Concepts or Vocabulary:

- **Programming**: Writing instructions for a computer to execute. Think of programming like writing a list of instructions for a friend. Instead of saying "turn left at the next street," you're telling the computer to do things like "display this message" or "calculate this value." You write these instructions in a programming language such as Python.
- **Algorithm**: A structured series of steps to solve a specific problem. An algorithm can be thought of as a recipe in cooking. For example, if you want to bake a cake, your algorithm would be the list of steps you follow (e.g., mix ingredients, bake for 30 minutes).

- **Source Code**: The text of a program written by the programmer. This is the raw code that you write and edit. For example, writing `print("Hello, World!")` in Python is a simple piece of source code. The source code is the human-readable version of your program before it gets translated into machine language.
- **Compiler/Interpreter**: Tools that convert source code into machine code that the computer can understand. In Python, we use an interpreter, which reads and executes code line-by-line, translating it into actions the computer can perform.
- **Variables**: Containers in which we store data that can be changed and used later. Imagine a variable as a storage box. You can put a piece of information inside, like a number or a word, and then retrieve it whenever you need it.
- **Data Types**: The types of data that can be stored in variables, such as:
  - **Integers (int)**: Whole numbers like 5 or -10.
  - **Floating-point numbers (float)**: Numbers with decimals, such as 3.14.
  - **Strings (str)**: Sequences of characters, like "Hello, World!"
  - **Booleans (bool)**: True or False values.
- **Text Editor or IDE**: A software application where you write and edit your source code. An IDE (Integrated Development Environment) like VSCode or PyCharm offers additional tools like syntax highlighting and auto-completion to make programming easier.
- **Type Casting**: Converting data from one type to another, such as converting a string `"21"` into an integer `21`.

Real-World Example:

To understand the importance of programming, imagine you have a long list of tasks to complete. Instead of manually writing everything down and checking items off one by one, what if you could create a program to automate the process?

For example, you could write a program that:

- Reminds you to drink water every hour.
- Automatically sends emails.
- Tracks your expenses and generates a weekly report.

In essence, programming can automate tasks that would otherwise be time-consuming, repetitive, or prone to human error. Apps like Google Calendar, calculators, and even social media platforms like Instagram are built on similar principles of task automation using programming.

Mini Code Snippets:

Below are some simple Python programs to help you understand how Python works:

1. **Your First Program: Hello, World!**

```
# This prints "Hello, World!" to the screen
print("Hello, World!")
```

Here, the `print()` function is used to display the message `"Hello, World!"`. The quotation marks indicate that `"Hello, World!"` is a string, a type of data representing text.

2. **Working with Variables:**

```
# Storing a name in a variable and printing a greeting
name = "Alice"
print("Hello, " + name + "!")
```

In this code, `name = "Alice"` assigns the string `"Alice"` to the variable `name`. The `+` operator concatenates (joins) strings together, so the output would be `"Hello, Alice!"`.

3. **User Input and Data Conversion:**

```
# Taking input from the user and converting it to an integer
age = input("Enter your age: ")  # This takes the user's input as a string
age = int(age)  # Convert the string to an integer
print("You are " + str(age) + " years old!")
```

`input()` allows the user to enter data, but it is stored as a string. `int()` is used to convert the string into an integer for further calculations.

Pre-Reading Questions or Simple Exercises:

To prepare for the upcoming class, try the following:

1. **Thought Exercise**: Imagine you are giving instructions to a robot to make a cup of coffee. What steps would you include in your instructions? How might that translate to an algorithm in programming?

2. **Exercise**: Write a Python program that asks for the user's name and greets them with: `"Hello, [NAME]!"`.
   Hint: Use the `input()` function to take the user's name as input, and then use `print()` to display the greeting.
3. **Challenge**: Think of an everyday task you do manually. How would you break it down into smaller, more manageable steps (i.e., an algorithm) that a computer could perform?

Common Mistakes or Misconceptions:

- **Indentation in Python**: One of the unique things about Python is that it uses indentation (spaces) to define code blocks. Forgetting to indent properly or using inconsistent indentation is one of the most common mistakes beginners make.
  - Example of correct indentation:

```
if True:
    print("This line is indented correctly!")
```

- **Mixing Data Types**: Another common issue is trying to mix incompatible data types without converting them. For instance, adding an integer to a string will cause an error.
  - Example of an error:

```
age = 25
print("You are " + age + " years old!")  # This will cause an error!
```

  To fix this, you must convert the integer to a string first:

```
print("You are " + str(age) + " years old!")
```

- **Using Functions Incorrectly**: Forgetting the parentheses when using functions like `print()` can lead to errors. Always remember that `print` needs parentheses, like this: `print("Hello")`.

Teaser for the Lecture:

In the next class, we'll dive deeper into **Conditional Statements**, explore the **Mathematical** and **Comparison Operators**, and learn how to implement logic to make decisions within your Python programs.