Notes for CSE101 - Programming Fundamentals (Lecture 1)

1. Introduction to Programming

What is Programming?

In the simplest terms, programming is giving instructions to a computer to perform specific tasks. Think of it like giving directions to a friend on how to reach your house. But instead of using words like "turn left," we use a programming language that the computer understands. In this lecture, we'll explore the basics of programming and set you up to write your first lines of code.

Why Learn Programming?

Programming is everywhere today, from the apps on your phone to the websites you visit. Learning how to program enables you to build applications, automate tasks, solve problems, and much more! Plus, with technology growing at such a fast pace, it's a skill that's in high demand.

Basic Concepts of Programming

- Algorithm: A step-by-step process to solve a problem. Example: A recipe for baking a cake.
- Source Code : The text or instructions written in a programming language.
- Compiler/Interpreter: Tools that convert your code into a form that the computer can understand.

Setting Up the Environment

To start programming, you need a suitable environment. For this course, we'll use Python. Here's what you need to set up:

- Text Editor or IDE: Install an Integrated Development Environment (IDE) like PyCharm or VSCode. These tools make coding easier by highlighting syntax errors and offering auto-completions.
- Python Interpreter: Make sure Python is installed on your computer. You can download it from Python's official website.

Your First Program

Let's start with the classic "Hello, World!" program. It's a simple way to see how programming works.

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

Explanation

- `print()` is a function in Python used to display messages.
- `"Hello, World!"` is the message inside the parentheses, and the output will be shown on your screen.

Real-Life Example:

Imagine programming like a conversation. Here, you're telling your computer to say "Hello, World!" like how you might greet someone.

Basic Syntax in Python

Python's syntax is designed to be easy to read and write. Here are some key rules:

- **Indentation**: In Python, indentation (spaces before lines of code) is crucial. For example:

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

- **Comments**: Use the # symbol to add comments in your code. Comments help you explain what the code does but don't affect how it runs.

  ```
  # This is a comment and will be ignored by the Python interpreter
  print("This is code!")
  ```

Problem Statement:

Write a Python program to print your name.

- Hint: Use the print() function.

2. Introduction to Python: Variables and Data Types

What are Variables?

Variables are like storage containers in programming. They hold information that you can use and manipulate later. For example, think of variables as a box where you can store your favorite snack. You can open the box (the variable) whenever you want to eat the snack (access the data).

Example:

```
name = "Alice"
age = 21
```

Here, `name` is a variable storing the string `"Alice"`, and `age` is a variable storing the number `21`.

Primitive Data Types in Python

1. Integers (**int**): Whole numbers, like 5 , 100 , -25 .
2. Floating-point numbers (**float**): Numbers with decimals, like `3.14`, `-9.8` .
3. Strings (**str**) : `Text, like "Hello" , "Python" .`
4. Booleans (**bool**) : Either True or False .

Real-Life Example

If you think of a contact form on a website, the name you enter is a string, your age is an integer, and whether you agree to the terms might be a boolean.

Variables and Constants

- Variable: A value that can change. Example: Your age changes every year.
- Constant: A value that doesn’t change. Example: The number of days in a week is always 7.

In Python, constants are typically written in uppercase to distinguish them. Example:

```
PI = 3.14159  # This value shouldn't change
```

Problem Statement:

Create a Python program that calculates and displays the total price of two products in a store. For each product, define variables to store the product name, its price, and the quantity available. The program should:

1. Use variables for each product's name, price (as a floating-point number), and quantity (as an integer).
2. Calculate the total cost for each product by multiplying the quantity by the price.
3. Add the total costs of both products to get the overall total.
4. Display the name of each product, its total cost, and the overall total using f-strings.

The f in the print(f"...") statement is used for f-string formatting, introduced in Python 3.6. An f-string allows you to embed expressions inside string literals, using curly braces {}. It stands for formatted string literal.erywhere! It's even used by NASA.

Solution

```
# Defining product 1 details
product1_name = "Apples"
product1_price = 1.50
product1_quantity = 3
product1_total = product1_price * product1_quantity  # Calculating total cost for product 1

# Defining product 2 details
product2_name = "Milk"
product2_price = 2.00
product2_quantity = 2
product2_total = product2_price * product2_quantity  # Calculating total cost for product 2

# Calculating overall total
overall_total = product1_total + product2_total

# Displaying results using f-strings
print(f"Product 1: {product1_name}, Quantity: {product1_quantity}, Price per unit: ${product1_price}")
print(f"Product 1 Total: ${product1_total:.2f}\n")

print(f"Product 2: {product2_name}, Quantity: {product2_quantity}, Price per unit: ${product2_price}")
print(f"Product 2 Total: ${product2_total:.2f}\n")

print(f"Overall Total: ${overall_total:.2f}")
```

Output

```
Product 1: Apples, Quantity: 3, Price per unit: $1.5
Product 1 Total: $4.50

Product 2: Milk, Quantity: 2, Price per unit: $2.0
Product 2 Total: $4.00

Overall Total: $8.50
```

Type Casting

Sometimes, you need to convert one data type into another. This is called type casting.

Example:

```python
age = "21"
age = int(age)  # Converts the string "21" into an integer 21
```

Introduction to Objects

Python is an object-oriented language. This means that everything in Python is an object, and objects have attributes (data) and methods (functions that act on the data). For example, a string is an object in Python with built-in methods like .upper() to convert it to uppercase.

Example:

```python
name = "alice"
print(name.upper())  # Outputs 'ALICE'
```

Problem Statement (*Topic might not be covered in lecture*):

- Create a Python program that does the following:

1. Takes a user's name (string) as input.
2. Converts the name to uppercase.
3. Prints the message: "HELLO, [UPPERCASE_NAME]!" .

Step-by-Step Solution

1. Step 1: Use the input() function to get the user's name.
2. Step 2: Use the .upper() method to convert the name.
3. Step 3: Use print() to display the final message.

Solution:

```python
name = input("Enter your name: ")
name_upper = name.upper()
print(f"HELLO, {name_upper}!")
```

The f in the print(f"...") statement is used for f-string formatting, introduced in Python 3.6. An f-string allows you to embed expressions inside string literals, using curly braces {}. It stands for formatted string literal.erywhere! It's even used by NASA.

Fun Facts about Python Programming

- Did you know? Python is named after the British comedy group Monty Python, not after the snake!
- Where is Python used? From web development (like Instagram) to AI research and scientific computing, Python is everywhere! It's even used by NASA.

Extra Challenge (*Might not covered in the class*):

Write a Python program that asks for two numbers, adds them together, and prints the result.

Hint: Use input() to take input and int() to convert strings to integers.