## Course Module: Programming 1 - Python

**Topic 1: Variables, Assignments, Inputs/Outputs**

---

## Introduction to Algorithms and Programming in Python

**Key Concepts:**

- **Introduction:**
  - Algorithm
  - Program
  - Interpreter vs. Compiler
- **Variables and Expressions**
- **Inputs/Outputs**

**Compilation and Interpretation**

**Automation through Computers:**

- Computers help automate tasks, but they need a program written in a human-readable language.
- This program must be converted into machine-readable text (a sequence of bytes) via a compiler or an interpreter.

**Compiler:**

- Translates the entire source code into an executable file once.

**Interpreter:**

- Translates and executes code line by line each time the program runs.
- Python is an interpreted language.

---

**Algorithm vs. Program**

**Solving Complex Problems:**

1. **Problem Solving:**
   - Use approximate syntax, abbreviations, and diagrams to create an algorithm (often in pseudocode).
2. **Final Solution:**
   - Write the final program that instructs the computer to perform the required tasks.

**General Concept of an Algorithm:**

- Applies to cooking recipes, experimental protocols, directions, etc.

---

## Exercise: Making an Omelet

**Task: Write the algorithm for making an omelet with one egg.**

**Specify:**

- Ingredients needed
- Tools required
- Actions to be taken (in order)

**Algorithm:**

```
Start
   {Ingredients}
     1 egg, salt, pepper, butter
   {Tools}
     1 bowl, fork, pan, spatula

   {Procedure}
   Break the egg into a bowl.
   Add salt and pepper.
   Beat the egg with a fork.
   Heat butter in a pan.
   Pour the beaten egg into the pan.
   Cook slowly until desired texture is achieved (runny to
well-cooked).
   Serve.
End
```

---

## Python Programs

**Two Execution Modes:**

1. Using the Python interpreter, line by line, like a calculator.
2. Writing a set of instructions in a file and then executing it via a Python interpreter.

---

## Variables

**Definition:**

- A container for information, identified by a name (identifier), and has content.

**Python Identifiers:**

- No need to declare variables.
- Must start with a letter or underscore (_).
- Can contain letters, digits, and underscores.
- Cannot be a Python reserved word.
- Case-sensitive (e.g., `my_var` ≠ `My_Var`).

**Examples:**

- Valid: `toto`, `proch_val`, `max1`, `MY_VALUE`, `_myvar`
- Invalid: `2be`, `C-3PO`, `my var`

**Conventions:**

- Use lowercase.
- Avoid accents.

---

## Assignment

**Storing Values:**

- Use the `=` sign for assignment.

**Examples:**

```
n = 33
a = 42 + 25
Ch = "hello"
euro = 6.55957
```

**Note:**

- The left side (identifier) receives the value from the right side (expression).
- The first assignment is also called initialization.

**Example:**

```
>>> a = 6
>>> a
6
>>> b = 9
>>> a == b
False
```

---

## Typing

**Variable Types:**

- Integer
- Float
- String
- Boolean

**Static Typing vs. Dynamic Typing:**

- Static: Type declared in the program.
- Dynamic: Type determined by the interpreter.

**Python is dynamically typed.**

```
>>> a = 17
>>> type(a)
<class 'int'>
>>> a = "hello"
>>> type(a)
<class 'str'>
>>> a = 3.14
>>> type(a)
<class 'float'>
>>> type(21 == 7*3)
<class 'bool'>
```

---

## Expressions

**Definition:**

- A formula that can be evaluated.

**Examples:**

```
42 + 2 * 5.3
3 * 2.0 - 5
"hello"
20 / 3
```

**Operators:**

- Arithmetic: `+`, `-`, `*`, `**` (power), `/`, `%` (modulo), `//` (integer division)
- Comparison: `==`, `!=`, `<`, `>`, `<=`, `>=`
- Logical: `or`, `and`, `not`

---

## Inputs/Outputs

**Interactions:**

- Inputs: Providing data to the program (e.g., keyboard inputs).
- Outputs: Displaying results or messages (e.g., screen text).

**Input Function:**

```
>>> text = input()
123
>>> text + 1  # Error
>>> val = int(text)
>>> val + 1  # OK, result: 124

>>> x = float(input("Enter a number:"))
Enter a number:
12.3
>>> x + 2  # Result: 14.3
```

**Output Function:**

```
>>> print("Sum of", a, "and", b, "is", a + b)
Sum of 20 and 13 is 33.

>>> print(a, b, sep=";")
20;13

>>> print("a =", a, "b =", b, sep="\n")
a =
20
b =
13
```