Programming Fundamentals - Week 2 Lectures

Authors: Refat Othman and Diaeddin Rimawi.

Lecture 1: Introduction to Python & Sequential Programming

Lecture Goals:

- Familiarize students with the Python programming environment.
- Understand the basic structure and syntax of a Python program.
- Learn how to perform sequential operations using variables, input/output, and conditionals.
- Develop foundational skills for writing basic Python programs involving decisions.

Topic 1: Installing Python and VS Code

- **What is Python?**
  Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used in web development, data analysis, artificial intelligence, automation, and education. Python emphasizes code readability and has a large, supportive community.
- **Install Python from python.org:**
  Guide students to python.org to download and install the latest version of Python. Explain the importance of adding Python to the system PATH during installation.
- **Install and configure VS Code with Python extension:**
  Introduce Visual Studio Code (VS Code) as a lightweight and powerful code editor. Demonstrate how to install the Python extension and configure it to detect the Python interpreter. Highlight features such as syntax highlighting, IntelliSense, and debugging tools.
- **Run a sample file:**

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

Explain how the interpreter processes the code line by line and produces output.

Topic 2: Run a Simple Program + Syntax/Runtime Errors

- **Discuss the structure of a simple program:**
  Describe how Python executes code sequentially, and explain the importance of indentation and syntax.
- **Syntax errors:**

```
print("Hello"  # missing closing parenthesis
x == 5         # using comparison without context
```

Use these to explain error messages and how to interpret them.

- **Runtime errors:**

```
x = int("abc")  # ValueError because 'abc' cannot be converted to an integer
y = 1 / 0       # ZeroDivisionError
```

Encourage students to read error messages carefully and use them for debugging. For instance, the `ValueError` in `x = int("abc")` occurs because the program attempts to cast the string 'abc' to an integer, which is not permitted—Python cannot interpret non-numeric characters as a valid integer. Understanding the cause of such errors helps students correct their code more efficiently.

Topic 3: Sequential Programming

- **Variables and data types:**
  Introduce variable declaration and the three basic data types: `int`, `float`, and `str`. Emphasize Python's dynamic typing.
- **Assign values, take input, print output:**

```
name = input("Enter your name: ")
print("Hello", name)
```

- **Basic calculations:**

```
a = float(input("Enter a: "))
b = float(input("Enter b: "))
print("Sum:", a + b)
```

Exercises:

- Read two numbers and calculate their sum and average.
- Convert a temperature from Celsius to Fahrenheit using the formula: `F = C * 9/5 + 32`.
- Calculate and print the perimeter of a rectangle from user input (2 × (length + width)).

Topic 4: If Statements

- **Conditional statements overview:**
  Explain decision-making in programs using `if`, `if-else`, and `if-elif-else`. Clarify the role of conditions in altering program flow.
- **Types of if statements:**
  - `if`: Executes a block of code if a condition is true.
  - `if-else`: Provides an alternative block if the condition is false.
  - `if-elif-else`: Allows checking multiple conditions in sequence.
- **Conditions and operators:**
  Introduce comparison operators: ==, `!=`, <, >, <=, >=
  Explain logical operators: `and`, `or`, `not`
  Mention identity (`is`) and membership (`in`) operators briefly for awareness.
- **Example:**

```
if score >= 90:
    print("A")
elif score >= 80:
    print("B")
else:
    print("C or below")
```

Exercises:

- Compare two numbers and print the greater.
- Check if a character is inside a string or not.
- Create a condition where a user is granted access only if both the username and password match predefined values. Use logical operators to combine conditions:

  ```
  username = input("Enter username: ")
  password = input("Enter password: ")

  if username == "admin" and password == "1234":
      print("Login successful")
  else:
      print("Invalid credentials")
  ```

Lecture 2: Arrays and Loops

Lecture Goals:

- Learn to define and manipulate collections using arrays (lists).
- Understand and apply iteration using `for` and `while` loops.
- Gain the ability to combine lists and loops to solve common programming problems.

Topic 1: Arrays

- **Introduction to lists:**
  Lists in Python are ordered, mutable collections that can hold elements of any data type. They are one of the most versatile and commonly used data structures in Python.

```
nums = [1, 2, 3]
names = ["Ali", "Sara"]
mixed = [42, "Hello", 3.14]
```

- **Accessing and modifying elements:**

```
print(nums[0])
nums[1] = 20
print(nums)
```

- **Iterating through a list:**

```
for name in names:
    print("Hello", name)
```

- **Input from user to fill a list:**

```
arr = []
for i in range(3):
    arr.append(int(input("Enter number: ")))
```

- **Common operations on lists:**

```
print(len(arr))
arr.append(99)
arr.remove(99)
arr.sort()
print("Sara" in names)
```

Topic 2: Loops

- **For loop basics:**

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

```
for i in range(1, 10, 2):
    print(i)
```

- **While loop basics:**

```
x = 0
```

```
while x < 5:
    print(x)
    x += 1
```

- **Combining loops with arrays:**

```
arr = [5, 3, 8]
for value in arr:
    print(value)
```

- **Input validation example:**

```
password = ""
while password != "1234":
    password = input("Enter password: ")
print("Access granted")
```

Exercises:

- Print even numbers up to a user-defined number N.
- Generate and print the multiplication table for a given number.
- Search for a specific value in a list.
- Find and print the smallest number in a list.
- Find the maximum value in a list.
- Compute the sum of all list elements.
- Count how many numbers are positive or negative.
- Print the list in reverse order.