Midpoint Review: Strengthening Our Python Foundations

Reflecting on Our Progress and Reinforcing Key Concepts

Michael Borck

Table of contents

Introduction

Objective: Review key programming concepts. - Importance of understanding these fundamentals for the project.

Simple 5-Step Development Methodology

Key Points: - Understand the Problem: Clearly define what needs to be solved. - Work out the Inputs and Outputs: Identify the data required and the expected results. - Work the Problems by Hand: Manually solve examples to create test cases. - Write out Pseudocode and Convert to Python: Plan the logic in pseudocode before coding. - Test with a Variety of Data: Ensure the program works with different inputs.

The Six Fundamental Operations of Computer Programs

1. **Input**
   - Receiving data from external sources (e.g., user input, file input).
2. **Output**
   - Sending data to external destinations (e.g., displaying data, writing to a file).
3. **Storage**
   - Saving and retrieving data (e.g., using variables, databases).
4. **Computation**
   - Performing arithmetic or logical operations (e.g., calculations, comparisons).
5. **Decision Making**
   - Evaluating conditions and making decisions (e.g., if-else statements).
6. **Iteration**
   - Repeating a set of instructions (e.g., loops such as for, while).

Basic Operators and Assignment

Key Points:

- Operators: Arithmetic (`+`, `-`, `*`, `/`), Comparison (`==`, `!=`, `>`, `<`), Logical (`and`, `or`, `not`).
- Assignment: Using `=` to assign values to variables.
- Example:

```
a = 10
b = 20
sum = a + b
is_equal = (a == b)
```

Basic Data Types and Structures

Key Points:

- Data Types: Integer, Float, String, Boolean.
- Data Structures: List, Dictionary, Tuple.
- Example:

```
integer = 10
float_num = 10.5
string = "Hello, World!"
boolean = True

# List
my_list = [1, 2, 3, 4, 5]

# Dictionary
my_dict = {"name": "Alice", "age": 25}

# Tuple
my_tuple = (1, 2, 3)
```

Sequence

- Definition: The order in which instructions are executed.
- Importance: Ensures that the program runs as expected.
- Example:

```
print("Step 1")
print("Step 2")
print("Step 3")
```

Selection (Conditionals)

Key Points:

- Definition: Making decisions based on conditions using `if`, `elif`, and `else`.
- Importance: Allows the program to take different actions based on different conditions.
- Example:

```
temperature = 20
if temperature > 25:
    print("It's hot!")
elif temperature > 15:
    print("It's warm!")
else:
    print("It's cold!")
```

Repetition (Loops)

Key Points:

- Definition: Repeating a set of instructions using `for` and `while` loops.
- Importance: Reduces code redundancy and handles repetitive tasks efficiently.
- Example:

```
for i in range(5):
    print(f"Iteration {i}")
```

Functions (Creation and Use)

Key Points:

- Definition: A block of code that performs a specific task, defined using `def`.
- Importance: Promotes code reuse and modularity.
- Example:

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

print(greet("Alice"))
```

Importing Packages

Key Points:

- Definition: Using external libraries to extend the functionality of your programs.
- Importance: Enables use of pre-written code for common tasks (e.g., data manipulation, visualization).
- Example:

```
import pandas as pd
import matplotlib.pyplot as plt
```

Basic File I/O

Key Points:

- Definition: Reading from and writing to files.
- Importance: Essential for data persistence and handling large datasets.
- Example:

```
# Writing to a file
with open('example.txt', 'w') as file:
    file.write('Hello, World!')

# Reading from a file
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

Data Management with CSV

Key Points:

- Definition: Reading from and writing to CSV files using `pandas`.
- Importance: Storing and manipulating data in a tabular format.
- Example:

```
import pandas as pd

# Reading a CSV file
data = pd.read_csv('weather_data.csv')
print(data.head())

# Writing to a CSV file
data.to_csv('processed_weather_data.csv', index=False)
```

Basic Data Visualization

Key Points:

- Definition: Creating visual representations of data using `matplotlib`.
- Importance: Helps in understanding and interpreting data.
- Example:

```
import matplotlib.pyplot as plt

temperatures = [20, 21, 19, 22, 23]
plt.plot(temperatures)
plt.title('Temperature over Days')
plt.xlabel('Day')
plt.ylabel('Temperature')
plt.show()
```