ASSIGNMENT

Programming in Python

APDMF2501DSBA(PR)

TP087108

HAND OUT DATE: February 2025
HAND IN DATE: 23 February 2025
WEIGHTAGE: 100%

INSTRUCTIONS TO CANDIDATES:

1. **Submit your assignment at the Moodle.**
2. **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld**
3. **Cases of plagiarism will be penalized**
4. **You must obtain 50% overall to pass this module**

Task 1: Fundamental Concepts of Python

Python is a high-level, interpreted programming language known for its simplicity and readability. It provides various built-in features, including different data types, control structures, and functions that help in efficient programming.

1. Data Types and Variables in Python:

Variables in Python are used to store data, and they do not require explicit type declaration. The type is inferred based on the assigned value. Some common data types include:

- **Integer (int):** Represents whole numbers (e.g., x = 10).
- **Floating-point (float):** Represents decimal numbers (e.g., y = 3.14).
- **String (str):** Represents text enclosed in quotes (e.g., name = "Alice").
- **Boolean (bool):** Represents truth values True or False (e.g., is active = True).
- **List:** An ordered collection of values (e.g., fruits = ["apple", "banana", "cherry"]).
- **Tuple:** Similar to lists but immutable (e.g., coordinates = (10, 20)).
- **Dictionary (dict):** Stores key-value pairs (e.g., person = {"name": "Alice", "age": 25}).
- **Set:** An unordered collection of unique elements (e.g., unique_numbers = {1, 2, 3, 4}).

2. Control Structures (Conditional Statements and Loops):

Control structures help in decision-making and executing code repeatedly.

Conditional Statements:

Used to execute different blocks of code based on conditions.

Example:

```
age = 18
if age >= 18:
    print("You are eligible to vote.")
elif age == 17:
    print("You will be eligible soon.")
else:
    print("You are not eligible yet.")

#Output: You are eligible to vote.
```

Loops:

Loops are used to execute a block of code multiple times.

For Loop: Iterates over a sequence (e.g., list, range, tuple).

Example:

```
for i in range(5):
    print(i)  # Outputs: 0, 1, 2, 3, 4
```

While Loop: Repeats as long as a condition is true.

Example:

```
count = 0
while count > 5:
    print("Count:", count)
    count += 1

#Output: 0, 1, 2, 3, 4
```

3. Functions and Their Importance in Programming

Functions allow modular and reusable code. They encapsulate a set of instructions that perform a specific task.

Defining a Function:

```
def greet(name):
    return f"Hello, {name}!"
print(greet("Alice"))  # Outputs: Hello, Alice!
```

Advantages of Using Functions:

- Improve code organization and readability.
- Reduce code repetition through reuse.
- Simplify debugging and maintenance.

Task 2: Problem-Based Strategy for a Task Tracking System

1. Key Components of a Task Tracker:

A simple task tracking system requires the following essential components:

- **Task Storage:** A structured way to store tasks and their statuses typically using a list or dictionary.
- **User Interaction:** A menu-driven interface that allows users to perform actions such as adding, viewing and updating tasks.
- **Data Persistence:** File handling to save and retrieve tasks from a file (tasks.txt) to maintain data across multiple program executions.
- **Menu System:** A simple command-line interface to navigate functionalities.

2. How Users Will Interact with the System

Start the Program: The system displays a menu with available options.

User Selects an Option: Options include:

- Add a new task.
- View existing tasks.

- Mark a task as completed.
- Exit the program.

Processing User Input: The system updates the task list based on user selections.

Saving Data: Tasks are saved to a file to ensure they persist after the program exits.

3. Steps needed to Development system in Python:

Define the Data Structure: Use a list of dictionaries where each task has a title and a status (e.g., {"task": "Finish assignment", "status": "Pending"}).

Implement Core Functions:

- **add_task():** Accepts user input and appends a new task to the list.
- **view_tasks():** Displays all tasks along with their statuses.
- **mark_completed():** Updates the status of a task.
- **save_tasks() and load_tasks():** Handle file operations for persistence.
- **Build a Menu System:** A loop that continuously displays options until the user decides to exit.
- **Validate Inputs:** Ensure that user input is valid and prevents incorrect entries.
- **Test the System:** Check functionality to handle edge cases like an empty task list or marking non-existent tasks as completed.

4. Potential Challenges and Solutions

Challenges	Solution
Handling duplicate tasks	Check if a task already exists before adding it to prevent redundancy.
User Input Errors	Validate inputs and handle incorrect entries gracefully.
Data loss after program termination	Use file handling to store and retrieve tasks persistently.
Task Management Complexity:	Use a dictionary structure to store tasks efficiently and allow easy retrieval.