Leah Sigana

Assignment 2 Week 5

1. **What is Python, and what are some of its key features that make it popular among developers? Provide examples of use cases where Python is particularly effective.**

   Python is a high-level, general-purpose programming language known for its readability, simplicity, and versatility. It's widely used for various applications, from web development and data science to scripting and automation.

   Key Features:

   - Python's syntax is clear and concise, resembling natural language, making it easier to learn
   - Python code doesn't need to be compiled into machine code before execution. An interpreter reads the code line by line, making development faster and more interactive.
   - Large Standard Library: Python comes with a rich collection of built-in modules and functions for various tasks, reducing the need for external libraries for common operations.
   - Can run on different operating systems (Windows, macOS, Linux) with minimal modifications.
   - It has Extensive Community & Libraries

   Use Cases:

   - Web Development
   - Data Science & Machine Learning
   - Scripting & Automation.
   - Software Development & Prototyping

2. **Describe the steps to install Python on your operating system (Windows, macOS, or Linux). Include how to verify the installation and set up a virtual environment.**

   Steps:

   i. Download the latest Python installer from the official website

ii. **Windows:** Run the downloaded installer, following the on-screen instructions. Ensure "Add Python to PATH" is checked for system-wide access.

iii. **Verification:** Open a terminal/command prompt and type python3 --version (or python on Windows if added to PATH).

3. **Write a simple Python program that prints "Hello, World!" to the console. Explain the basic syntax elements used in the program.**

   Simple Program:

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

   Explanation:

- `print()`: Built-in function to display output on the console.
- `"Hello, World!"`: String literal enclosed in double quotes, the message to be printed.

4. **List and describe the basic data types in Python. Write a short script that demonstrates how to create and use variables of different data types.**

   Basic Data Types:

- **Integers (int):** Whole numbers (e.g., 10, -5).
- **Floats (float):** Decimal numbers (e.g., 3.14, -2.5).
- **Strings (str):** Text data enclosed in quotes (e.g., "This is a string").
- **Booleans (bool):** Logical values, True or False.

   Example Script:

```
Python
age = 25  # Integer
pi = 3.14159  # Float
name = "Alice"  # String
is_active = True  # Boolean

print(f"Age: {age}, Name: {name}")  # f-string for
formatted output
```

5. **Explain the use of conditional statements and loops in Python. Provide examples of an if-else statement and a for loop.**

Conditional Statements (if-else):

Python
```
x = 10
y = 5

if x > y:
    print("x is greater than y")
else:
    print("x is less than or equal to y")
```

Loops (for):

Python
```
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

6. **What are functions in Python, and why are they useful? Write a Python function that takes two arguments and returns their sum. Include an example of how to call this function.**

Functions: Reusable blocks of code that perform specific tasks.

Python
```
def add(x, y):
  """This function adds two numbers and returns the sum."""
  return x + y

result = add(10, 20)
print(result)  # Output: 30
```

7. **Describe the differences between lists and dictionaries in Python. Write a script that creates a list of numbers and a dictionary with some key-value pairs, then demonstrates basic operations on both.**

Lists: Ordered collections of items, similar to arrays in other languages. You access elements using their **index** (position) within the list, starting from 0. Lists can contain duplicate elements and elements of different data types.

Dictionaries: Unordered collections of key-value pairs. You access elements using their **keys**, which can be any immutable data type (strings, integers, tuples). Dictionaries cannot contain duplicate keys, and each key has a corresponding value of any data type.

```
# Shopping list (list)
shopping_list = ["milk", "bread", "eggs"]
first_item = shopping_list[0]  # first_item will be
"milk"

# Phonebook (dictionary)
phonebook = {"Alice": "123-456-7890", "Bob": "987-654-
3210"}
alice_number = phonebook["Alice"]  # alice_number will
be "123-456-7890"    "
```

8. **What is exception handling in Python? Provide an example of how to use try, except, and finally blocks to handle errors in a Python script.**

Exception Handling: Mechanism to handle errors or unexpected situations during program execution.

Example:

Python

```
try:
  # Code that might raise an exception (e.g., division by
zero)
  result = 10 / 0
except ZeroDivisionError:
  print("Error: Division by zero")
finally:
  # Code that always executes, regardless of exceptions
(e.g., closing files)
  print("This block always runs")
```

9. **Explain the concepts of modules and packages in Python. How can you import and use a module in your script? Provide an example using the math module.**

Modules: Reusable Python files containing code that can be imported into other scripts.

Packages: Collections of modules organized hierarchically using directories.

Python
```
import math  # Import the math module

print(math.pi)  # Access the pi constant from the math module

# Example using a package (assuming pandas is installed)

import pandas as pd

data = pd.read_csv("data.csv")  # Use pandas to read a CSV file
print(data.head())  # Display the first few rows of the data
```

10. **How do you read from and write to files in Python? Write a script that reads the content of a file and prints it to the console, and another script that writes a list of strings to a file.**

Reading from Files:

Python
```
with open("data.txt", "r") as file:
  contents = file.read()  # Read the entire file content
  print(contents)

# Read line by line
with open("data.txt", "r") as file:
  for line in file:
    print(line.strip())  # Read and remove trailing newline characters
```

Writing to Files:

Python
```
with open("output.txt", "w") as file:
  file.write("This is written to the file.\n")
  file.write("Adding another line.")

# Append to existing file (use "a" mode)
with open("output.txt", "a") as file:
  file.write("\nMore content appended.")
```