SE-Assignment-6

Assignment: Introduction to Python
Instructions:

Questions:

1. Python Basics:

- 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, interpreted programming language known for its simplicity, readability, and versatility.

KEY FEATURES

1) Easy to learn and use: Python has a clean and concise syntax, making it accessible to beginners and experienced programmers alike.
2) Cross-platform compatibility: Python code can run on various operating systems, including Windows, macOS, and Linux.
3) Extensive standard library: Python comes with a vast collection of built-in modules and libraries that provide functionality for a wide range of tasks.
4) Dynamic typing: Python variables can hold values of different data types without explicit declaration.
5) Interpreted language: Python code is executed line by line, making it suitable for rapid prototyping and interactive development

USE CASES

1) Web development: Python frameworks like Django and Flask are popular choices for building web applications.
2) Data analysis and machine learning: Libraries like NumPy, Pandas, and Scikit-learn make Python a powerful tool for data manipulation, analysis, and machine learning.
8\3) Scripting and automation: Python's simplicity and readability make it a great choice for writing scripts to automate repetitive tasks.
4) Scientific computing: Python's libraries like SciPy and Matplotlib provide tools for scientific and numerical computing
5) Software Development: Developing desktop applications with Tkinter or PyQt.
6) Machine Learning: Using libraries like TensorFlow, Keras, and Scikit-learn.

2. Installing Python:

- 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.

-> Visit the official Python website , python.org and download the appropriate installer for your operating system.

-> Run the installer and follow the on-screen instructions to complete the installation process.

-> Open a terminal or command prompt and type python --version to verify the installation. If Python is installed correctly, it will display the version number

---> macOS:

Use the downloaded .pkg file and follow the installation prompts.

Alternatively, use Homebrew: brew install python3.

---> Linux:

Use your package manager, ie
: sudo apt-get install python3 for Debian-based systems.

---> Create a virtual environment: python3 -m venv myenv

Activate the virtual environment:

Windows: myenv\Scripts\activate

macOS/Linux: source myenv/bin/activate

3. Python Syntax and Semantics:

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

-> print("Hello, World!")

4. Data Types and Variables:

- 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.

-> int: Integer values, e.g., 10

-> float: Floating-point numbers, e.g., 10.5

-> str: String literals, e.g., "Hello"

-> bool: Boolean values, True or False

-> list: Ordered, mutable collection, e.g., [1, 2, 3]

-> tuple: Ordered, immutable collection, e.g., (1, 2, 3)

-> dict: Key-value pairs, e.g., {'key': 'value'}

-> tuple: Ordered, immutable collection, e.g., (1, 2, 3)

Script demonstrating the creation and usage of variables with different data types:

Integer

```
age = 27

print(age)
```

Float

```
pi = 3.14

print(pi)
```

String

name = "John Doe"

print(name)

List

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

print(fruits)

Tuple

coordinates = (10.0, 20.0)

print("Coordinates:", coordinates)

Dictionary

person = {"name": "John", "age": 30, "city": "New York"}

print(person)

5. Control Structures:

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

-> Conditional statements allow the execution of code based on certain conditions.

if-else statement

```
age = 18

if age >= 18:

 print("You are an adult.")

else:

print("You are a baby.")

# for loop

fruits = ["apple", "mango", "cherry"]

for fruit in fruits:

 print(fruit)
```

6. Functions in Python:

- 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 are reusable blocks of code that perform a specific task.

-> They help organize code, promote code reuse, and improve code readability.

```
def add_numbers(a, b):
```

result = a + b

return result

sum = add_numbers(10, 20)

print(sum)

7. Lists and Dictionaries:

- 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:

1) Ordered collection of items
2) Items can be of different data types
3) Accessed by index
4) Mutable (elements can be modified)

-> Dictionaries:

1) Unordered collection of key-value pairs
2) Keys must be unique and immutable (e.g., strings, numbers, tuples)
3) Values can be of any data type and are accessed using keys
4) Mutable (keys and values can be added, modified, or removed)

List example

numbers = [1, 2, 3, 4, 5]

```
print(numbers)  # Output: [1, 2, 3, 4, 5]
```

Accessing list elements

```
print(numbers[0])

print(numbers[-1])
```

Dictionary example

```
person = {

 "name": "John Doe",

 "age": 30,

"city": "New York"

}

print(person)
```

Accessing dictionary values

```
print(person["name"])

print(person.get("age"))
```

8. Exception Handling:

- 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: A mechanism to handle runtime errors, allowing the program to continue execution or gracefully exit.

```
try:

 result = 10 / 0  # Raises ZeroDivisionError

except ZeroDivisionError:

 print("Error: Division by zero.")

finally:

    print("This block will always execute.")
```

9. Modules and Packages:

- 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 in Python are files containing Python definitions and statements. They allow you to organize and reuse code.

-> Packages are collections of modules organized into hierarchical directories. They provide a way to structure Python's module namespace.

```
import math
```

Accessing module functions

result = math.sqrt(36)

print(result)

Accessing module constants

pi = math.pi

print(pi)

10. File I/O:

- 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 a file

with open("work.txt", "r") as file:

content = file.read()

****print(content)****

Writing to a file

lines = ["Line 1", "Line 2", "Line 3"]

with open("work.txt", "w") as file:

```
file.writelines(line + "\n" for line in lines)
```