NAME: EMMANUEL CLINTON ODHIAMBO
SOFTWARE ENGINNERING ASSIGNMENT 6 SOLUTIONS
INTRODUCTION TO PYTHON

1. **What is Python, and what are some of its key features that make it popular among developers?**

Python is a high-level, interpreted programming language known for its readability and simplicity. Some key features that contribute to its popularity include:

- **Readability:** Python's syntax is clear and easy to understand, which helps developers write code faster and with fewer errors.
- **Versatility:** Python is used in web development, data analysis, artificial intelligence, scientific computing, and more.
- **Extensive Libraries:** Python has a rich set of libraries and frameworks (like NumPy, pandas, Django) that accelerate development.
- **Community Support:** A large and active community provides a wealth of resources, tutorials, and third-party modules.

Use Cases:

- **Web Development:** Using frameworks like Django and Flask.
- **Data Analysis:** With libraries like pandas and NumPy.
- **Machine Learning:** Leveraging libraries like TensorFlow and scikit-learn.
- **Automation:** Writing scripts to automate repetitive tasks.

2. **Installing Python**

Steps to Install Python:

On Windows:

- Download the installer from the official Python website.
- Run the installer and check the box that says “Add Python to PATH”.
- Click “Install Now” and follow the prompts.

On macOS:

- Download the installer from the official Python website.
- Open the downloaded .pkg file and follow the installation instructions.

On Linux:

- Use the package manager. For example, on Ubuntu, you can run:

```
sudo apt update

sudo apt install python3
```

- Verify the installation by running:

```
python3 –version
```

Setting Up a Virtual Environment:

- Install the venv package if it's not already installed:
  ```
  sudo apt install python3-venv  # For Linux
  ```

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

- Activate the virtual environment:
  On Windows: ‘myenv\Scripts\activate’
  On macOS/Linux: ‘source myenv/bin/activate’

3. Python Syntax and Semantics

Simple Python Program:

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

Explanation:

- print(): A built-in function that outputs text to the console.
- "Hello, World!": A string literal that will be printed.

4. Data Types and Variables

Basic Data Types:

- Integer: Whole numbers (e.g., 5, -3)
- Float: Decimal numbers (e.g., 3.14, -0.001)
- String: Text (e.g., "Hello", 'Python')
- Boolean: True or False values (e.g., True, False)

Example Script:

```
# Integer
age = 25

# Float
height = 5.9

# String
name = "Alice"

# Boolean
is_student = True

print(age, height, name, is_student)
```

5. **Control Structures**

Conditional Statements:

```
x = 10

if x > 5:
    print("x is greater than 5")
else:
    print("x is 5 or less")
```

Loops:

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

Explanation:

- if-else: Executes code based on a condition.
- for loop: Iterates over a sequence of values.

6. **Functions in Python**

What are Functions?

Functions are reusable blocks of code that perform a specific task. They help in modularizing code and avoiding repetition.

Example Function:

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

result = add_numbers(3, 5)
print(result)
```

Explanation:

- def add_numbers(a, b):: Defines a function named add_numbers with parameters a and b.
- return a + b: Returns the sum of a and b.

7. **Lists and Dictionaries**

Lists vs. Dictionaries:

Lists: Ordered, mutable collections of items. Example: [1, 2, 3]

Dictionaries: Unordered collections of key-value pairs. Example: {'name': 'Alice', 'age': 25}

Example Script:

```
# List
numbers = [1, 2, 3, 4, 5]

# Dictionary
person = {'name': 'Alice', 'age': 25}

# List operations
numbers.append(6)
print(numbers)

# Dictionary operations
person['city'] = 'New York'
print(person)
```

8. Exception Handling

What is Exception Handling?

Exception handling is used to manage errors that occur during the execution of a program, preventing crashes and allowing for graceful error recovery.

Example:

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("You cannot divide by zero!")
finally:
    print("This block always executes.")
```

Explanation:

- try: Block of code where exceptions may occur.
- except: Handles specific exceptions.
- finally: Executes code regardless of whether an exception occurred.

9. Modules and Packages

Modules and Packages:

Module: A file containing Python code (e.g., math.py).

Package: A collection of modules organized in directories.

Example Using math Module:

```
import math
print(math.sqrt(16))  # Output: 4.0
```

Explanation:

- import math: Imports the math module.
- math.sqrt(): Uses the sqrt function from the math module.

10. File I/O

Reading from a File:

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

Writing to a File:

```
with open('example.txt', 'w') as file:
    lines = ["Line 1", "Line 2", "Line 3"]
    for line in lines:
        file.write(line + "\n")
```

Explanation:

- with open(): Opens a file and ensures it is properly closed after operations.
- 'r' and 'w': Modes for reading and writing, respectively.