Bildad Wahome SE-Assignment-6: Introduction to Python

1. Python Basics:

Python is a high-level, interpreted programming language known for its readability and simplicity. Key features include:

- Interpreted Language: Executes code line-by-line, easing debugging.
- Dynamic Typing: No need to declare variable types.
- Extensive Standard Library: Provides modules for various tasks (e.g., math, datetime).
- Cross-platform: Runs on Windows, macOS, and Linux.
- Community Support: Vast resources and libraries like NumPy, Pandas, and Django.

Use Cases:

- Web Development: Frameworks like Django and Flask.
- Data Science: Libraries like Pandas, NumPy, and SciPy.
- Automation/Scripting: Automating repetitive tasks.
- Machine Learning: Libraries like TensorFlow and scikit-learn.

2. Installing Python:

Windows:

Download Python from python.org.

Run the installer, check "Add Python to PATH," and click "Install Now."

Verify installation: Open Command Prompt and type python --version.

macOS:

Install Homebrew if not installed: /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Install Python: brew install python

Verify installation: Open Terminal and type python3 --version.

Linux:

Install Python: sudo apt-get install python3

Verify installation: Open Terminal and type python3 --version.

Virtual Environment:

Create: python3 -m venv env

Activate:

- Windows: .\env\Scripts\activate
- macOS/Linux: source env/bin/activate

3. Python Syntax and Semantics:

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

print(): Function to output text to the console.

"Hello, World!": String literal.

4. Data Types and Variables:

Basic Data Types:

- int: Integer (e.g., x = 5)
- float: Floating-point number (e.g., y = 3.14)
- str: String (e.g., name = "Alice")
- bool: Boolean (e.g., is_valid = True)

Example:

```
x = 5
y = 3.14
name = "Alice"
is_valid = True
print(x, y, name, is_valid)
```

5. Control Structures:

Conditional Statements:

```
age = 18
if age >= 18:
   print("Adult")
else:
   print("Minor")
```

Loops:

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

6. Functions in Python:

Functions:

- Encapsulate reusable code.
- Improve code organization and readability.

Example:

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


result = add(3, 4)
print(result)  # Output: 7
```

7. Lists and Dictionaries:

Lists: Ordered, mutable collections.

```
numbers = [1, 2, 3, 4, 5]
numbers.append(6)
print(numbers)  # Output: [1, 2, 3, 4, 5, 6]
```

Dictionaries: Unordered, mutable collections of key-value pairs.

```
person = {"name": "Alice", "age": 25}
person["city"] = "Wonderland"
print(person)  # Output: {'name': 'Alice', 'age': 25, 'city': 'Wonderland'}
```

8. Exception Handling:

Example:

```
try:
   result = 10 / 0
except ZeroDivisionError:
   print("Cannot divide by zero!")
finally:
   print("Execution complete.")
```

9. Modules and Packages:

Modules: Files containing Python code (functions, classes).

Packages: Directories containing multiple modules.

Example:

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

10. File I/O:

Reading from a File:

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

Writing to a File:

```
lines = ["First line", "Second line", "Third line"]
with open('output.txt', 'w') as file:
  for line in lines:  file.write(line + "\n")
```