SE-ASSIGNMENT-6: INTRO TO PYTHON.

1.1.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, interpreted programming language mostly known because of its ease of use and readability. Its key features include;

* Readability and simplicity- its syntax is clean and easy to understand thus developers can write clear and logical code.
* Interpreted language- code is executed line b line thus making debugging easier.
* Extensive standard library- Python comes with a vast standard library that supports many common programming tasks such as file I/O, system calls, and even Internet protocols.
* Portability- python runs on many platforms.
* Integration capabilities- python can integrate with other languages and technologies such as C/C++, JAVA and NET.

Where python is particularly effective;

* Web development.
* Data science and machine learning.
* Automation and scripting.
* Scientific computing.

1.2. 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.

Installing python on Windows;

* Download python installer- go to python website and download the latest version.
* Run the installer- run the downloaded installer ensure to check the “add python to path” and select install now to start the installation process.
* Verify the installation- open the command prompt, type “python –version” and press enter. This should show the version you installed.
* Set up a virtual environment- in your command prompt, navigate to the project directory using cd command, run python -m venv myenv to create a virtual environment named myenv. Activate the virtual environment using myenv\Scripts\activate. You should see (myenv) prefix in your command prompt.

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

print("Hello, World!")

print() – is a built in python used to output text to the console.

String – is a sequence of characters enclosed in quotation marks in this case “hello world”.

1.4. 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 in Python:
* Integer (int):
  + Represents whole numbers, both positive and negative.
  + Example: 5, -10, 0
* Float (float):
  + Represents decimal numbers (floating-point numbers).
  + Example: 3.14, -0.001, 2.0
* String (str):
  + Represents sequences of characters enclosed in single or double quotes.
  + Example: "Hello", 'World'
* Boolean (bool):
  + Represents truth values: True or False.
  + Example: True, False
* List (list):
  + An ordered, mutable collection of items, which can be of different data types.
  + Example: [1, 2, 3], ["apple", "banana", 3.14]
* Tuple (tuple):
  + An ordered, immutable collection of items, similar to lists.
  + Example: (1, 2, 3), ("apple", "banana")
* Dictionary (dict):
  + An unordered collection of key-value pairs, where each key must be unique.
  + Example: {"name": "Alice", "age": 25}
* Set (set):
  + An unordered collection of unique items.
  + Example: {1, 2, 3}, {"apple", "banana"}

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

Conditional statements allow you to execute certain blocks of code based on whether a condition is true or false. The primary conditional statement in Python is the if statement, which can be combined with elif (else if) and else to create more complex conditions.

Example;

age = 18

if age < 18:

print("You are a minor.")

elif age == 18:

print("You just turned 18!")

else:

print("You are an adult.")

1.6. 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 are reusable blocks of code that perform specific tasks. They help organize code, make it more readable, and promote reuse, allowing you to execute the same logic multiple times without rewriting it.

Why they’re useful:

* Modularity.
* Reusability.
* Readability.
* Testing.

Example;

def add\_numbers(a, b):

return a + b

#this is how to call it.

result = add\_numbers(5, 10)

print("The sum is:", result)

1.7. 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.

Differences:

* Lists are defined using square brackets ‘[ ]’ while dictionaries are enclosed in curly brackets’{ }’.
* Lists are an ordered collection of items while unordered collection of key-value pairs.
* Lists can contain duplicate items while dictionaries keys are unique.

Bellow is how lists and dictionaries can be manipulated:

# Creating a list of numbers

numbers = [10, 20, 30, 40, 50]

# Basic operations on the list

print("Original List:", numbers)

# Adding an item

numbers.append(60)

print("After Append:", numbers)

# Removing an item

numbers.remove(30)

print("After Remove:", numbers)

# Accessing an item by index

print("First Item:", numbers[0])

# Iterating through the list

print("All Numbers:")

for num in numbers:

print(num)

# Creating a dictionary with key-value pairs

person = {

"name": "Alice",

"age": 30,

"city": "New York"

}

# Basic operations on the dictionary

print("\nOriginal Dictionary:", person)

# Adding a new key-value pair

person["email"] = "alice@example.com"

print("After Adding Email:", person)

# Updating a value

person["age"] = 31

print("After Updating Age:", person)

# Accessing a value by key

print("Name:", person["name"])

# Iterating through the dictionary

print("Person's Details:")

for key, value in person.items():

print(f"{key}: {value}")

1.8. 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 in Python is a mechanism to manage errors that occur during the execution of a program. The primary keywords used for exception handling are try, except, and finally.

* try: The block of code to test for errors.
* except: The block of code that executes if an error occurs in the try block.
* finally: The block of code that executes after the try and except blocks, regardless of whether an exception was raised or not.

Example:

def divide\_numbers(num1, num2):

try:

result = num1 / num2

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

return None

except TypeError:

print("Error: Invalid input type. Please provide numbers.")

return None

else:

print("Division successful!")

return result

finally:

print("Execution of divide\_numbers complete.")

# Test the function

print(divide\_numbers(10, 2)) # Should print 5.0

print(divide\_numbers(10, 0)) # Should handle division by zero

print(divide\_numbers(10, "a")) # Should handle invalid input type

1.9. 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.

A **module** is a single file (with a .py extension) that contains Python code, including functions, classes, and variables. Modules allow you to break your code into manageable parts and reuse code across different programs.

Example:

# my\_module.py

def greet(name):

return f"Hello, {name}!"

A **package** is a collection of related modules organized in a directory hierarchy. A package must contain a special file named \_\_init\_\_.py, which can be empty or contain initialization code for the package.

Example:

my\_package/

\_\_init\_\_.py

my\_module.py

another\_module.py

how to import using a module: use the ‘import’ statement

example: import math

1.10. 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 contents of a file:

file\_path = 'example.txt'

try:

with open(file\_path, 'r') as file:

content = file.read()

print("File Content:")

print(content)

except FileNotFoundError:

print(f"Error: The file {file\_path} does not exist.")

writes strings to a file:

# Writing to a file

file\_path = 'output.txt'

lines = ["Hello, World!", "Welcome to Python.", "File handling is easy!"]

with open(file\_path, 'w') as file:

for line in lines:

file.write(line + "\n") # Write each line followed by a newline