ASSIGNMENT

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 an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
- Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
  (Python Software foundation , n.d.)
- Use cases
  a. Data Analysis
  b. Artificial intelligence

2. Installing Python:
    a. 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.
        1. Download the Python installer
        2. Run the installer
        3. Customize the installation (optional)
        4. Install Python
        5. Verify the installation
    b. Once the installation is complete, you can verify that Python has been installed correctly by opening the Command Prompt (search for “cmd” in the Start menu) and typing the following command:
       python --version
       Press Enter, and you should see the version of Python you installed displayed in the output. This confirms that Python has been successfully installed on your computer.
    (Kinsta Inc. , 2024)
    c. On Windows, invoke the venv command as follows:
    c:\>Python35\python -m venv c:\path\to\myenv

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!")
- print function: This built-in function outputs data to the console.
- Parentheses (): These enclose the argument passed to the print function, which is the string "Hello, World!" in this case.
- String literal: "Hello, World!" enclosed in double quotes (") is a string literal representing the text to be printed.

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.
   Datatypes

- Integer
- Float
- Character
- String

Demonstration of Datatypes

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> num1 = 5
>>> print(num1, 'is of type', type(num1))
5 is of type <class 'int'>
>>> num2 = 2.0
>>> print(num2, 'is of type', type(num2))
2.0 is of type <class 'float'>
>>> um3 = 1+2j
>>> print(um3, 'is of type', type(um3))
(1+2j) is of type <class 'complex'>
>>>
```

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

- Loops and conditional statements are fundamental building blocks of programming in Python. These constructs allow programmers to automate repetitive tasks and control the flow of their programs based on certain conditions.

For loop

```
>>> fruits = ['apple', 'banana', 'cherry']
>>> for fruit in fruits:
...     print(fruit)
...
apple
banana
cherry
```

If-else loop

```
>>> x = -5
>>> if x > 0:
...     print("x is positive")
... else:
...     print("x is negative")
...
x is negative
>>>
```

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.

- Python Functions is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again. (Geeks for Geeks, 2024)

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> def sum_numbers(x, y):
...
...   return x + y
...
>>> result = sum_numbers(51,13)
>>> print(result)
64
>>>
```

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. (Geeks for Geeks , 2024)

- Lists and Dictionaries in Python are inbuilt data structures that are used to store data. Lists are linear in nature whereas the dictionaries stored the data in key-value pair.

```python
# Create a list of numbers
numbers = [12, 59, 39, 62, 87]

# Accessing elements by index
last_number = numbers[-1]  # Accessing the last element (index -1)
print(f"last number: {last_number}")

# Looping through the list
print("Numbers in the list:")
for number in numbers:
  print(number)

# Modifying an element
numbers[0] = 10  # Replacing the first element (index 2) with 10
print("\nList after modification:", numbers)

# Create a dictionary
person = {
  "name": "Lisa",
  "age": 12,
  "city": "Kumasi"
}

# Accessing dictionary values by key
name = person["name"]
print(f"\nPerson's name: {name}")

# Adding a new key-value pair
person["occupation"] = "Student"
print(f"\nDictionary after adding occupation: {person}")

# Checking if a key exists
if "age" in person:
  print("\nThe 'age' key exists in the dictionary.")
```

```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Thinkpad/OneDrive/Desktop/Python-assignement-
plp.py', wdir='C:/Users/Thinkpad/OneDrive/Desktop')
last number: 87
Numbers in the list:
12
59
39
62
87

List after modification: [10, 59, 39, 62, 87]

Person's name: Lisa

Dictionary after adding occupation: {'name': 'Lisa', 'age': 12, 'city':
'Kumasi', 'occupation': 'Student'}

The 'age' key exists in the dictionary.

In [2]:
```

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.

- The requirement for handling exceptions in Python arises when an error occurs that can cause the program to terminate. Errors interrupt the flow of the program at the point where they appear, so any further code stops executing. This error is called an exception. (Indeed, 2024)
- **Try**-All exception-handling blocks in Python begin with the "try" keyword. It is used to check the code for errors. Programmers write only those codes within this block, which might raise an exception. If the code in the try block is error-free, the try block executes, and the subsequent except block is skipped. Here is an example of how the "try" keyword is written in code:
  try:
  # statements

- **Except** -The "except" keyword is always used in conjunction with the "try" keyword to form try-except blocks. When the program encounters an error in the code within the preceding try block, the code gets handled in the except block. The "except" keyword is used to define what to do in case of specific exceptions. Here is an example of how the "except" keyword is written in code:
  try:
  # statements
  except:
  # statements

  (Indeed, 2024)

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.

- A module is a single file containing Python code, whereas a package is a collection of modules that are organized in a directory hierarchy.
  (Info Edge (India) Ltd, 2024)

```
>>> import math
>>> print(math.sqrt(9809))
99.04039579888602
>>>
```

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.

- **Read Only ('r') :** Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exist, raises the I/O error. This is also the default mode in which a file is opened.
- **Write Only ('w') :** Open the file for writing. For the existing files, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist. (Geeks for Geeks , 2024)

```
Created on Sat Jun 22 10:28:28 2024

@author: Thinkpad
"""

def read_file(file_path):

  try:
    # Open the file in read mode
    with open(file_path, 'r') as file:
      # Read all content into a variable
      content = file.read()
      # Print the content
      print(content)
  except FileNotFoundError:
    print(f"Error: File '{file_path}' not found.")


file_path = 'C:/Users/Thinkpad/OneDrive/Desktop/plp-file-read-from.txt'
read_file(file_path)
```

```
In [3]: runfile('C:/Users/Thinkpad/OneDrive/Desktop/reading-files-plp.py',
wdir='C:/Users/Thinkpad/OneDrive/Desktop')
In cases of late diagnosis, neonatal jaundice can lead to permanent damage of
brain cells, a condition
known as kernicterus. This lethal condition can also cause deafness or
hearing loss, cerebral palsy, and
profound developmental delay. Fortunately, kernicterus is avoidable through
early detection and
treatment. High levels of bilirubin can be controlled through phototherapy, a
process that involves the
use of blue light to break down unconjugated bilirubin into a conjugated form
that is water soluble and,
as such, can be easily excreted out of the body. For extremely high levels,
excess bilirubin must be
removed through exchange transfusion.
```

```
Created on Sat Jun 22 10:39:04 2024

@author: Thinkpad
"""

def write_to_file(file_path, data):
  """
  This function writes a list of strings to a file.
  """
  try:
    # Open the file in write mode
    with open(file_path, 'w') as file:
      # Write each element in the list to the file, followed by a newline
      for item in data:
        file.write(f"{item}\n")
  except (IOError, TypeError) as e:
    print(f"Error writing to file: {e}")

# Sample list of strings to write
data = ["BILIRUBIN LEVEL DETECTION OF NEONATAL JAUNDICE USING COLOR SENSOR IN PHOTOTHERAPY"]

# Replace 'path/to/your/file.txt' with the desired output file path
file_path = 'C:/Users/Thinkpad/OneDrive/Desktop/plp-file-read-from.txt'
write_to_file(file_path, data)

print(f"Successfully wrote data to {file_path}")
print(file_path)
```

```
In [5]: runfile('C:/Users/Thinkpad/OneDrive/Desktop/plp-write-to-file.py',
wdir='C:/Users/Thinkpad/OneDrive/Desktop')
Successfully wrote data to C:/Users/Thinkpad/OneDrive/Desktop/plp-file-read-
from.txt
C:/Users/Thinkpad/OneDrive/Desktop/plp-file-read-from.txt
```