**Global Trend Programming Profile Assessment Questions**

**1.Implement a Python class MaxHeap that supports the following operations: insert, delete, and get\_max. Ensure the operations maintain the properties of a max-heap.**

PROGRAM:  
# Filename: max\_heap.py

class MaxHeap:

    def \_\_init\_\_(self):

        self.heap = []

    def insert(self, val):

        self.heap.append(val)

        self.\_heapify\_up(len(self.heap) - 1)

    def delete(self):

        if len(self.heap) > 1:

            self.\_swap(0, len(self.heap) - 1)

            max\_val = self.heap.pop()

            self.\_heapify\_down(0)

        elif self.heap:

            max\_val = self.heap.pop()

        else:

            max\_val = None

        return max\_val

    def get\_max(self):

        if self.heap:

            return self.heap[0]

        return None

    def \_heapify\_up(self, index):

        parent\_index = (index - 1) // 2

        if index > 0 and self.heap[index] > self.heap[parent\_index]:

            self.\_swap(index, parent\_index)

            self.\_heapify\_up(parent\_index)

    def \_heapify\_down(self, index):

        left\_child\_index = 2 \* index + 1

        right\_child\_index = 2 \* index + 2

        largest = index

        if left\_child\_index < len(self.heap) and self.heap[left\_child\_index] > self.heap[largest]:

            largest = left\_child\_index

        if right\_child\_index < len(self.heap) and self.heap[right\_child\_index] > self.heap[largest]:

            largest = right\_child\_index

        if largest != index:

            self.\_swap(index, largest)

            self.\_heapify\_down(largest)

    def \_swap(self, i, j):

        self.heap[i], self.heap[j] = self.heap[j], self.heap[i]

# Usage Example

heap = MaxHeap()

heap.insert(10)

heap.insert(20)

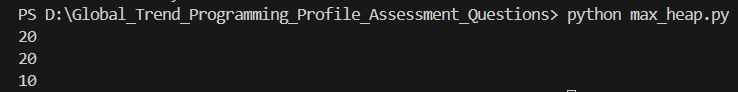
heap.insert(5)

print(heap.get\_max())  # Output: 20

print(heap.delete())  # Output: 20

print(heap.get\_max())  # Output: 10

OUTPUT:



**2.Write a Python function that takes a list of URLs, attempts to download their content, and retries up to 3 times if an error occurs. Use appropriate error handling to manage different types of exceptions.**

PROGRAM:

import requests

from time import sleep

def download\_content(urls, max\_retries=3):

    contents = []

    for url in urls:

        for attempt in range(max\_retries):

            try:

                response = requests.get(url)

                response.raise\_for\_status()

                contents.append(response.text)

                print(f"Downloaded content from {url}")

                break

            except requests.exceptions.RequestException as e:

                print(f"Attempt {attempt + 1} failed for {url}: {e}")

                sleep(1)  # wait before retrying

        else:

            contents.append(None)

            print(f"Failed to download content from {url} after {max\_retries} attempts")

    return contents

# Usage Example

if \_\_name\_\_ == "\_\_main\_\_":

    urls = ["http://example.com", "http://example.org", "http://nonexistenturl.xyz"]

    contents = download\_content(urls)

    for i, content in enumerate(contents):

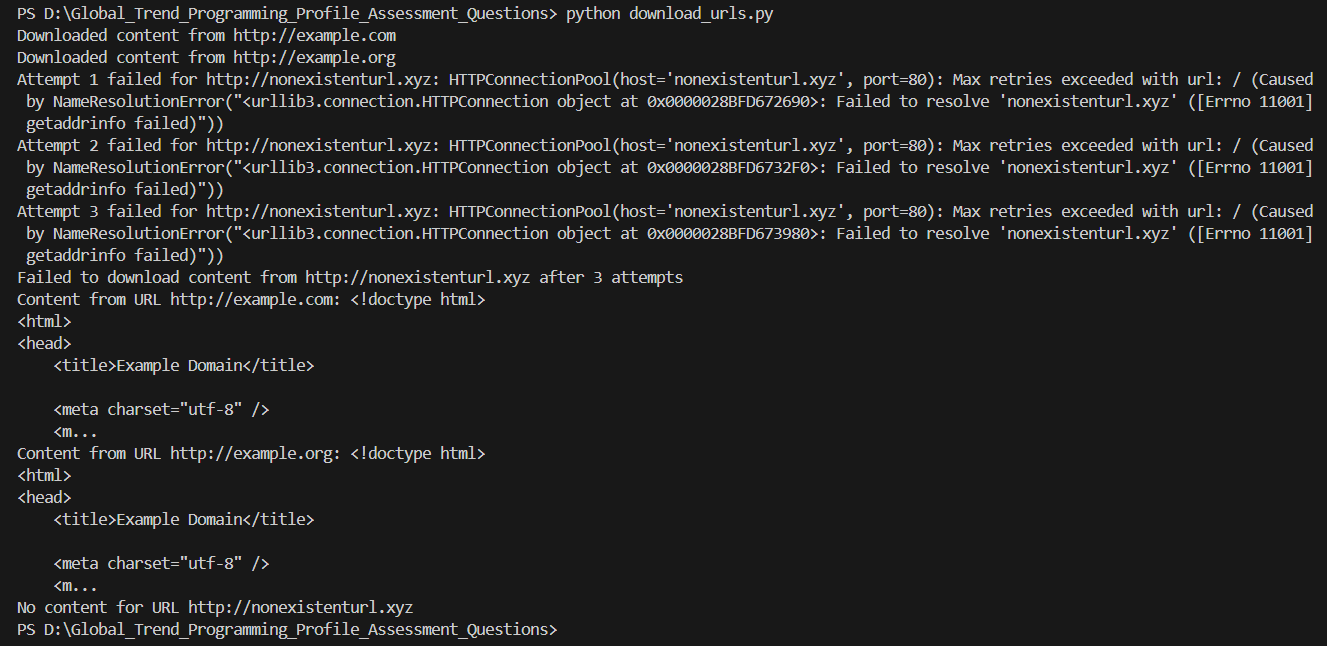
        if content:

            print(f"Content from URL {urls[i]}: {content[:100]}...")  # Print first 100 characters

        else:

            print(f"No content for URL {urls[i]}")

OUTPUT:



**3.Write a Python script that trains a simple linear regression model using scikit-learn. Use a dataset of your choice, split it into training and testing sets, and evaluate the model's performance.**

PROGRAM:

# Filename: linear\_regression.py

#Installing pandas pip install pandas

#PS D:\Global\_Trend\_Programming\_Profile\_Assessment\_Questions> pip install pandas scikit-learn requests

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Sample dataset

data = {'X': [1, 2, 3, 4, 5], 'Y': [1, 2, 3, 4, 5]}

df = pd.DataFrame(data)

X = df[['X']]

y = df['Y']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

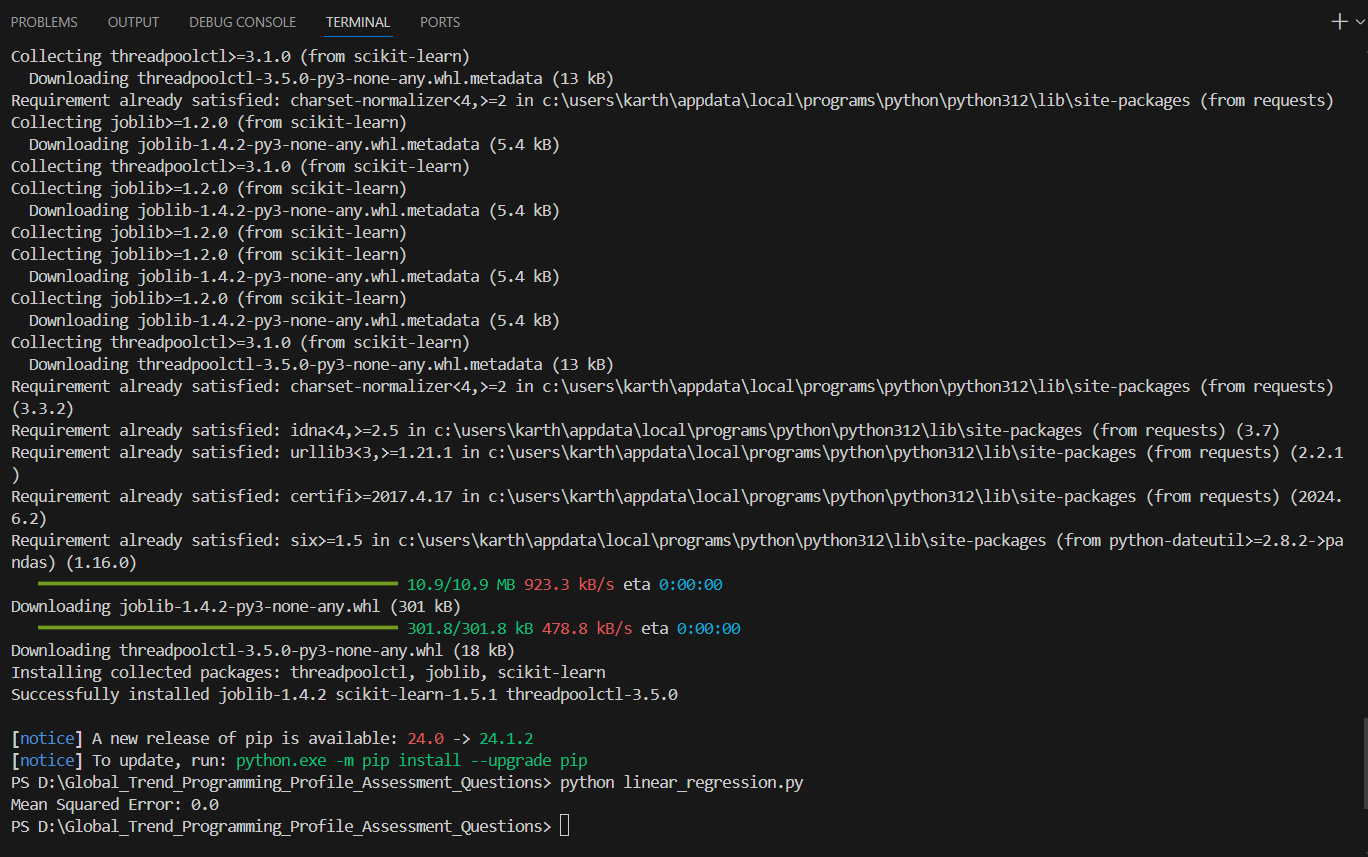
model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

OUTPUT:



**4.Using pandas, write a Python function to clean and preprocess a given DataFrame, which involves handling missing values, normalizing numerical columns, and encoding categorical columns.**

PROGRAM:

# Filename: preprocess\_dataframe.py

import pandas as pd

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

def preprocess\_dataframe(df):

    numerical\_features = df.select\_dtypes(include=['int64', 'float64']).columns

    categorical\_features = df.select\_dtypes(include=['object']).columns

    numerical\_transformer = Pipeline(steps=[

        ('imputer', SimpleImputer(strategy='median')),

        ('scaler', StandardScaler())

    ])

    categorical\_transformer = Pipeline(steps=[

        ('imputer', SimpleImputer(strategy='constant', fill\_value='missing')),

        ('onehot', OneHotEncoder(handle\_unknown='ignore'))

    ])

    preprocessor = ColumnTransformer(

        transformers=[

            ('num', numerical\_transformer, numerical\_features),

            ('cat', categorical\_transformer, categorical\_features)

        ])

    df\_processed = preprocessor.fit\_transform(df)

    return pd.DataFrame(df\_processed)

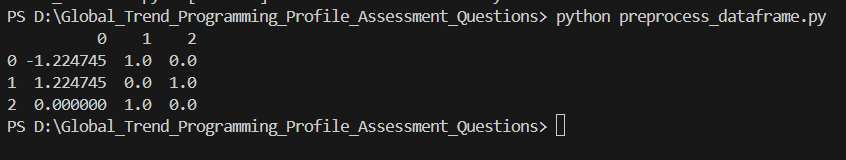
# Usage Example

data = {'A': [1, 2, None], 'B': ['a', 'b', 'a']}

df = pd.DataFrame(data)

print(preprocess\_dataframe(df))

OUTPUT:



**5.Write a Python function to compute the nth Fibonacci number using recursion.**

PROGRAM:

# Filename: fibonacci.py

def fibonacci(n):

    if n <= 0:

        return 0

    elif n == 1:

        return 1

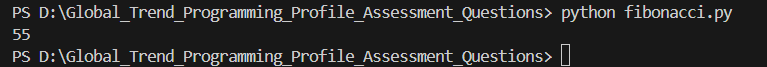
    else:

        return fibonacci(n - 1) + fibonacci(n - 2)

# Usage Example

print(fibonacci(10))  # Output: 55

OUTPUT:



6.Write a Python function that divides two numbers and handles the case where the divisor is zero by returning a custom error message.

PROGRAM:

# Filename: divide.py

def divide(a, b):

    try:

        return a / b

    except ZeroDivisionError:

        return "Error: Cannot divide by zero"

# Usage Example

print(divide(10, 2))  # Output: 5.0

print(divide(10, 0))  # Output: Error: Cannot divide by zero

OUTPUT:



7.Write a Python decorator that measures the execution time of a function and logs it. Apply this decorator to a function that performs a computationally expensive task.

PROGRAM:

# Filename: execution\_time\_decorator.py

import time

def execution\_time(func):

    def wrapper(\*args, \*\*kwargs):

        start\_time = time.time()

        result = func(\*args, \*\*kwargs)

        end\_time = time.time()

        print(f"Execution time: {end\_time - start\_time} seconds")

        return result

    return wrapper

@execution\_time

def expensive\_task():

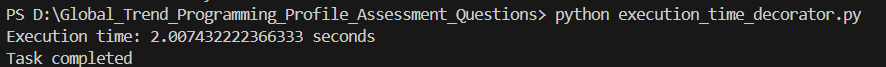
    time.sleep(2)

    return "Task completed"

# Usage Example

print(expensive\_task())

OUTPUT:



**8.Write a Python function that takes two numbers and an operator (as a string) and performs the corresponding arithmetic operation (addition, subtraction, multiplication, or division).**

PROGRAM:

# Filename: arithmetic\_operations.py

def arithmetic\_operation(a, b, operator):

    if operator == '+':

        return a + b

    elif operator == '-':

        return a - b

    elif operator == '\*':

        return a \* b

    elif operator == '/':

        return a / b if b != 0 else "Error: Division by zero"

    else:

        return "Error: Invalid operator"

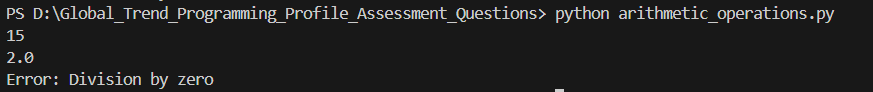
# Usage Example

print(arithmetic\_operation(10, 5, '+'))  # Output: 15

print(arithmetic\_operation(10, 5, '/'))  # Output: 2.0

print(arithmetic\_operation(10, 0, '/'))  # Output: Error: Division by zero

Output:



**9 . Write a Python function that generates a random password. The password should contain a mix of uppercase letters, lowercase letters, digits, and special characters.**

PROGRAM:

# Filename: random\_password.py

import random

import string

def generate\_password(length=12):

    characters = string.ascii\_letters + string.digits + string.punctuation

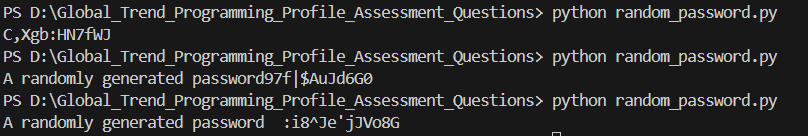
    password = ''.join(random.choice(characters) for i in range(length))

    return password

# Usage Example

print("A randomly generated password  :"+generate\_password())  # Output: A randomly generated password

OUTPUT:



**10. Write a Python function that takes a 2D list (matrix) and returns its transpose.**

PROGRAM:

# Filename: transpose\_matrix.py

def transpose(matrix):

    return [list(row) for row in zip(\*matrix)]

matrix = [

    [1, 2, 3],

    [4, 5, 6],

    [7, 8, 9]

]

print("Matrix before :" +str(matrix))

print("Transpose of matrix is: " + str(transpose(matrix)))

OUTPUT:

