Global Trend Programming Profile Assessment

# Ayush Vishwakarma

[ayush.vishwakarma2016@gmail.com](mailto:ayush.vishwakarma2016@gmail.com)  
+91 9554051373  
<https://github.com/iusekarma/>  
<https://linkedin.com/in/iusekarma/>

Answers on Next Page

## Question. 1:

class MaxHeap:

def \_\_init\_\_(self):

self.h = []

def insert(self, val):

self.h.append(val)

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

def delete(self):

if len(self.h) == 0:

return None

if len(self.h) == 1:

return self.h.pop()

max\_val = self.h[0]

self.h[0] = self.h.pop()

self.\_heapify\_down(0)

return max\_val

def get\_max(self):

if not self.h:

return None

return self.h[0]

def \_heapify\_up(self, idx):

p\_idx = (idx - 1) // 2

if p\_idx >= 0 and self.h[idx] > self.h[p\_idx]:

self.h[idx], self.h[p\_idx] = self.h[p\_idx], self.h[idx]

self.\_heapify\_up(p\_idx)

def \_heapify\_down(self, idx):

largest = idx

l\_idx = 2 \* idx + 1

r\_idx = 2 \* idx + 2

if l\_idx < len(self.h) and self.h[l\_idx] > self.h[largest]:

largest = l\_idx

if r\_idx < len(self.h) and self.h[r\_idx] > self.h[largest]:

largest = r\_idx

if largest != idx:

self.h[idx], self.h[largest] = self.h[largest], self.h[idx]

self.\_heapify\_down(largest)

heap = MaxHeap()

heap.insert(10)

heap.insert(20)

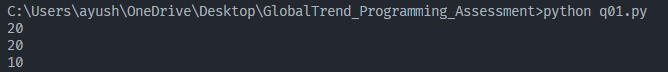
heap.insert(5)

print(heap.get\_max())

print(heap.delete())

print(heap.get\_max())

### Output:



## Question 2.

import requests

from requests.exceptions import RequestException

def download\_urls(urls):

results = {}

for url in urls:

attempt = 0

success = False

while attempt < 3 and not success:

try:

response = requests.get(url)

response.raise\_for\_status()

success = True

except RequestException as e:

attempt += 1

if success:

results[url] = response.text

else:

results[url] = None

return results

urls = ["https://google.com","https://amazon.in","https://example.com"]

res = download\_urls(urls)

for url in res:

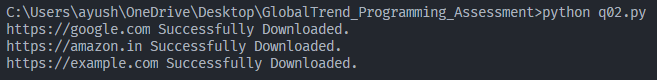
if res[url]:

print(url,"Successfully Downloaded.")

else:

print(url,"Error: No Content")

### Output:



## Question 3.

from sklearn.datasets import load\_diabetes

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from matplotlib import pyplot as plt

# import numpy as np

diabetes = load\_diabetes()

X = diabetes.data[:,None,8]

Y = diabetes.target

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)

r2 = r2\_score(y\_test, y\_pred)

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

print(f"R-squared: {r2}")

plt.scatter(X\_test,y\_test,label="Data Points")

plt.plot(X\_test,y\_pred,label="Fitted Line")

plt.xlabel("Feature")

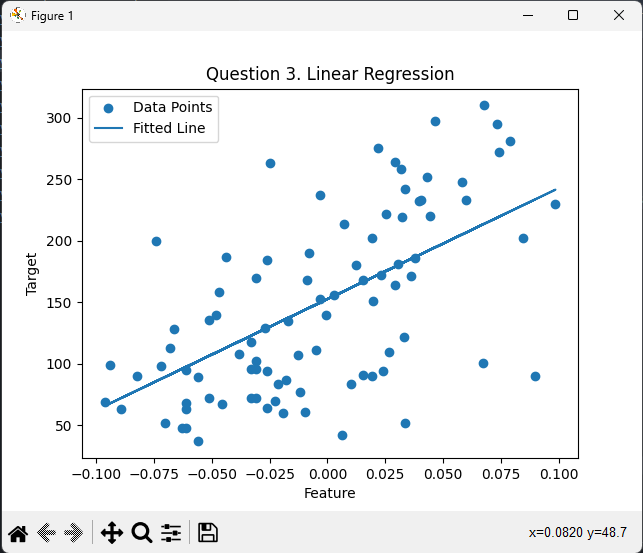
plt.ylabel("Target")

plt.title("Question 3. Linear Regression")

plt.legend()

plt.show()

### Output:



## Question 4.

import pandas as pd

def clean\_and\_preprocess(df):

for column in df.columns:

if df[column].dtype in ["int64", "float64"]:

df[column].fillna(df[column].mean(), inplace=True)

else:

df[column].fillna(df[column].mode()[0], inplace=True)

num\_cols = df.select\_dtypes(include=["int64", "float64"]).columns

df[num\_cols] = (df[num\_cols] - df[num\_cols].mean()) / df[num\_cols].std()

cat\_cols = df.select\_dtypes(include=["object", "category"]).columns

df = pd.get\_dummies(df, columns=cat\_cols)

return df

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

# Dummy Data

data = {

"age": [25, 30, 35, None, 40],

"salary": [50000, 60000, 70000, 80000, None],

"city": ["New York", "Los Angeles", "New York", "San Francisco", None],

}

df = pd.DataFrame(data)

print("Original Data:")

print(df)

df\_cleaned = clean\_and\_preprocess(df)

print("\nCleaned & Preprocessed Data:")

print(df\_cleaned)

### Output:



## Question 5.

def fibonacci(n):

if n == 1 or n == 2:

return 1

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

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

print(fibonacci(5))

### Output:



## Question 6.

def divide(x, y):

if y != 0:

return x / y

else:

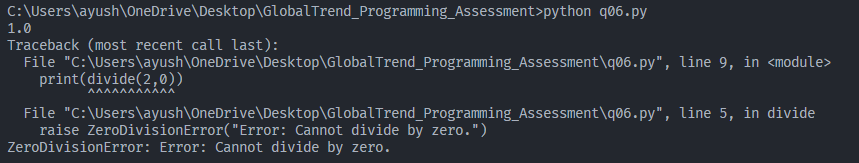
raise ZeroDivisionError("Error: Cannot divide by zero.")

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

print(divide(2,2))

print(divide(2,0))

### Output:



## Question 7.

import time

import logging

logging.basicConfig(level=logging.INFO)

def log\_time(func):

def wrapper(\*args, \*\*kwargs):

s\_time = time.time()

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

e\_time = time.time()

exec\_time = e\_time - s\_time

logging.info(f"Executed {func.\_\_name\_\_} in {exec\_time:.4f} seconds")

return res

return wrapper

@log\_time

def fibonacci(n):

def fib\_recursion(n):

if n == 1 or n == 2:

return 1

return fib\_recursion(n - 1) + fib\_recursion(n - 2)

return fib\_recursion(n)

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

n = 35

res = fibonacci(n)

print(f"Fibonacci value of {n}: {res}")

### Output:



## Question 8.

def operate(n1, n2, o):

if o == '+':

return n1 + n2

elif o == '-':

return n1 - n2

elif o == '\*':

return n1 \* n2

elif o == '/':

if n2 == 0:

raise ZeroDivisionError("Error: Cannot divide by zero.")

return n1 / n2

else:

return "Error: Invalid operator. Please only use one of '+', '-', '\*', '/'."

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

n1 = 10

n2 = 5

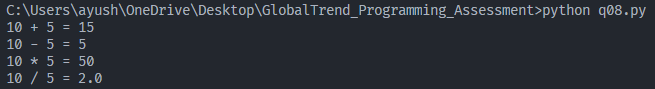
operators = ['+', '-', '\*', '/']

for o in operators:

res = operate(n1, n2, o)

print(f"{n1} {o} {n2} = {res}")

### Output:



## Question 9.

import string, random

def generate\_password(length):

char\_set = string.ascii\_letters + string.digits + string.punctuation

return "".join(random.sample(char\_set, length))

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

password = generate\_password(10)

print("Random password :",password)

### Output:



## Question 10.

def transpose(mat):

r = len(mat)

c = len(mat[0])

t\_mat = [[0 for \_ in range(r)] for \_ in range(c)]

for i in range(r):

for j in range(c):

t\_mat[j][i] = mat[i][j]

return t\_mat

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

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print("Original Matrix :")

print(matrix)

print("/nTransposed Matrix :")

print(transpose(matrix))

### Output:

