

**NAME: EULITA MUBAIWA**

**REG NUMBER**: **PG247122H**

**YEAR: 1.2**

**ASSIGNMENT: 3**

**LECTURE: MS MURIMI**

1. write a python function named classify-number that takes an integer as input and returns positive if the number is greater than zero, negative if the number is less than zero and zero if the number is zero . use a while loop repeatedly prompt the user for a number until they enter a valid integer

def classify\_number(number):

if number > 0:

return "positive"

elf number < 0:

return "negative"

Else:

return "zero"

while True:

try:

user\_input = input ("Enter an integer: ")

number = int(user\_input)

result = classify\_number(number)

print(f"The number is {result}.")

break

except ValueError:

print("Invalid input. Please enter a valid integer.")

1. **Define a python function calculate \_ average that accepts a variable number of arguments (using args) and return the average of those number. Print an error message and prompt the user to enter a valid number again.**

def calculate average(\*args):

if len(args) == 0:

return "No numbers provided."

return sum(args) / len(args)

while True:

try:

user\_input = input ("Enter numbers separated by spaces: ")

numbers = [float(num) for num in user\_input. Split()]

average = calculate average(\*numbers)

print (f"The average of the numbers is {average}.")

break

except ValueError:

print("Invalid input. Please enter valid numbers.")

3**. Write a program that handles user input for a number. Use a try block to catch any ValueError exceptions that may occur if the user inputs an invalid number. Print an error message and prompt the user to enter valid number again**

While True:

try:

user\_input = input ("Enter a number: ")

number = float (user\_input)

print (f"You entered the number: {number}")

break # Exit the loop if input is valid

Except ValueError:

print ("Invalid input. Please enter a valid number.")

4. **Create a python script that writes a list of names to a file called names.txt.. Each name should be on new line. Then, read the file each name to the console. Use the with statement to handle file operations and ensure the file is properly closed.**

# List of names

names = ["Alice", "Bob", "Charlie", "David", "Eve"]

# Writing names to the file

with open ("names.txt", "w") as file:

for name in names:

file.write (name + "\n")

# Reading and printing names from the file

with open("names.txt", "r") as file:

for line in file:

Print (line.strip ()) # Strip removes any extra whitespace

5. **Write a python program that uses a lamda function and the map function to convert a list of temperature in Celsius to Fahrenheit. The formula for conversion is F=C\*9/5+32. Include a sample list of Celsius temperatures and print the converted list**

# Sample list of temperatures in Celsius

celsius\_temperatures = [0, 10, 20, 30, 40, 100]

# Conversion formula using lambda and map

fahrenheit\_temperatures = list(map(lambda c: c \* 9 / 5 + 32, celsius\_temperatures))

# printing the converted temperatures

print("Temperatures in Celsius:”, celsius\_temperatures)

print("Temperatures in Fahrenheit:", fahrenheit\_temperatures)

6. **Create a function called divide \_\_numbers that takes two parameters, numerators and denominator. Use a try block to handle:**

**Division by zero (ZeroDivisionError)**

**Invalid input types (TypeError)**

**The function should return the result of the division if successful, and print an appropriate error message for each exception**

Print (divide \_numbers (10, "a")) # Should handle invalid input type def divide numbers(numerator, denominator):

Try:

result = numerator / denominator

return f"The result of the division is {result}."

Except ZeroDivisionError:

return "Error: Division by zero is not allowed."

except TypeError:

return "Error: Invalid input type. Please provide numbers for both numerator and denominator."

# Example usage:

print(divide\_ numbers(10, 2)) # Should return the result

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

**7. Define custom exception class called NegativeNumberError that inherits from the built-in Exception class. Write a function check \_ positive that raises this exception. If a given number is negative**

**Demonstrate this using a Try Block.**

# Define custom exception

class NegativeNumberError(Exception):

def \_\_init\_\_ (self, message="The number is negative!"):

self.message = message

super().\_\_init\_\_ (self.message)

# Function to check if a number is positive

def check\_positive(number):

if number < 0:

raise NegativeNumberError (f"Error: {number} is a negative number.")

return f"{number} is a positive number."

# Demonstration with a try block

Try:

user\_input = int(input("Enter a number: "))

result = check\_positive(user\_input)

print (result)

except NegativeNumberError as e:

print(e)

except ValueError:

print("Invalid input. Please enter a valid integer.")

8**. Write a python program that uses the random module to generate a list of 10 random integers between 1 and 100. Then, calculate and print the average of the generated numbers. Use appropriate functions to achieve this**

import random

# Function to generate a list of random integers

def generate\_random\_numbers(count, start, end):

return [random.randint(start, end) for \_ in range(count)]

# Function to calculate the average of a list of numbers

def calculate\_average(numbers):

return sum(numbers) / len(numbers)

# Generate 10 random integers between 1 and 100

random\_numbers = generate\_random\_numbers(10, 1, 100)

# Calculate the average

average = calculate\_average(random\_numbers)

# Print the results

print("Generated random numbers:", random\_numbers)

print("Average of the numbers:", average)

9. (i) import re

# Function to extract email addresses

def extract\_emails(text):

# Regular expression pattern for email addresses

email\_pattern = r'[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'

# Find all email addresses in the text

emails = re.findall(email\_pattern, text)

return emails

# Sample text containing email addresses

sample\_text = """

Hello, please contact us at support@example.com or sales@example.co.uk.

You can also reach out to admin123@domain.net for further assistance.

"""

# Extract emails from the sample text

email\_addresses = extract\_emails(sample\_text)

# Print the extracted email addresses

print("Extracted Email Addresses:")

for email in email\_addresses:

print(email)

ii.

import re

# Function to validate a date

def validate\_date(date):

# Regular expression for YYYY-MM-DD format

date\_pattern = r'^\d{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2]\d|3[0-1])$'

if re.match(date\_pattern, date):

return f"'{date}' is a valid date."

else:

return f"'{date}' is not a valid date."

# Example usage

user\_input = input("Enter a date in the format YYYY-MM-DD: ")

validation\_result = validate\_date(user\_input)

print(validation\_result)

iii.

import re

# Function to replace all occurrences of a word

def replace\_word(text, old\_word, new\_word):

# Use re.sub to replace occurrences of old\_word with new\_word

# \b ensures that only whole words are replaced

replaced\_text = re.sub(rf'\b{old\_word}\b', new\_word, text)

return replaced\_text

# Sample text

sample\_text = "The cat jumped over the cat because the cat was agile."

# Word to replace and its replacement

iv. import re

# Function to split string by non-alphanumeric characters

def split\_by\_non\_alphanumeric(text):

# Regular expression pattern for non-alphanumeric characters

split\_pattern = r'\W+'

# Use re.split to split the string

result = re.split(split\_pattern, text)

# Remove any empty strings from the result

return [word for word in result if word]

# Sample string

sample\_text = "Hello

10.

import socket

def start\_server():

host = '127.0.0.1' # Localhost

port = 12345 # Port to listen on

Try:

# Create a socket object

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind((host, port))

server\_socket.listen(1) # Listen for one connection

print(f"Server is listening on {host}:{port}...")

# Accept a client connection

conn, addr = server\_socket.accept()

print(f"Connection established with {addr}")

# Send a message to the client

message = "Hello from the server!"

conn.sendall(message.encode('utf-8'))

print("Message sent to the client.")

# Close the connection

conn.close()

except socket.error as e:

print(f"Socket error: {e}")

except Exception as e:

print(f"An unexpected error occurred: {e}")

finally:

server\_socket.close()

print("Server socket closed.")

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

start\_server()