**Abbottabad University Of Science And Technology**

**Department Of Computer Science 2nd Sem**

****

**Lab Task no 12**

**Name : M Hassan Ashraf**

**Roll No : 2023132**

**Semester : 2nd**

**Subject : OOP**

**Submitted To** : **Sir Jamal Abdul Ahad**

**Q1: Write a function in python that takes a parameter and use an assertion to raise an AssertionError if the parameter is not within a specified range.**

def check\_range(value):

# Specify the allowed range

min\_value = 0

max\_value = 100

# Use assertion to check if the value is within the specified range

assert min\_value <= value <= max\_value, f"Value should be between {min\_value} and {max\_value}"

# Example usage:

try:

check\_range(50) # This will not raise an error

check\_range(120) # This will raise an AssertionError

except AssertionError as e:

print(f"AssertionError: {e}")

**Q2: Write a program that attempts to connect to a server. Use exception handling to catch potential network-related errors, such as socket.error or ConnectionError.**

import socket

def connect\_to\_server(server\_address, port):

try:

create\_connection((server\_address, port), timeout=5) as sock:

print(f"Successfully connected to {server\_address}:{port}")

# Add your further logic here if the connection is successful

except (socket.error, ConnectionError) as e:

print(f"Error connecting to {server\_address}:{port}")

print(f"Exception: {e}")

# Example usage:

server\_address = "example.com"

port = 80

connect\_to\_server(server\_address, port)

**Q3: Create a program that reads data from a file. Use exception handling to catch and display an error message if the file does not exist.**

def read\_file(file\_path):

try:

# Attempt to open and read the file

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

data = file.read()

print(f"File content:\n{data}")

except FileNotFoundError:

print(f"Error: File not found. Please check the file path: {file\_path}")

# Example usage:

file\_path = "example.txt"

read\_file(file\_path)

**Q4: Define a custom exception class and use it in a program where a certain condition should raise this custom exception.**

# Define a custom exception class

class CustomError(Exception):

def \_\_init\_\_(self, message="A custom error occurred."):

self.message = message

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

def check\_condition(value):

# Check the condition that should trigger the custom exception

if value < 0:

raise CustomError("Value should be a non-negative number.")

# Example usage:

try:

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

check\_condition(user\_input)

print(f"The entered number is: {user\_input}")

except CustomError as ce:

print(f"CustomError: {ce}")

except ValueError:

print("Error: Please enter a valid number.")