Practical-19

May 19, 2025

0.1 Practical 19 :-

Name: Sarthak Sanay
Enrollment No: 230031101611051

0.1.1 Problem Statement 1:-

Division with Multiple Exception Types:
Write a function safe_divide(a, b) that takes two arguments and performs division. Handle the following exceptions:
`ZeroDivisionError`: When b is zero, print "Cannot divide by zero" and return None.
`TypeError`: When either a or b is not a number, print "Invalid input type" and return None.

[1]:
```python
def safe_divide(a, b):
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        print("Cannot divide by zero")
        return None
    except TypeError:
        print("Invalid input type")
        return None

print(safe_divide(10, 2))
print(safe_divide(10, 0))
print(safe_divide(10, "a"))
```

```
5.0
Cannot divide by zero
None
Invalid input type
None
```

0.1.2 Problem Statement 2:-

File Processing with Specific Exceptions:
Create a function process_file(filename) that attempts to open and read a file. Handle exceptions for:
`FileNotFoundError`: Print "File not found"

`PermissionError`: Print “Permission denied”
`IOError`: Print “An IOError occurred”.

[2]:
```
def process_file(filename):
    try:
        with open(filename, 'r') as file:
            content = file.read()
            print(content)
    except FileNotFoundError:
        print("File not found")
    except PermissionError:
        print("Permission denied")
    except IOError:
        print("An IOError occurred")

filename = input("Enter the filename: ")
process_file(filename)
```

```
Enter the filename:  samplefile.txt
```

```
File not found
```

0.1.3 Problem Statement 3:-

Custom Exception for Age Validation:
Define a custom exception `InvalidAgeError` and write a function `validate_age(age)` that raises this exception if the age is not between 0 and 120 (inclusive). Use this function to check a list of ages and handle the exception by printing an appropriate message.

[3]:
```
class InvalidAgeError(Exception):
    pass

def validate_age(age):
    if age < 0 or age > 120:
        raise InvalidAgeError(f"Invalid age: {age}. Age must be between 0 and␣
 ↪120.")

ages = [25, -5, 130, 50, 0, 120]

for age in ages:
    try:
        validate_age(age)
        print(f"Age {age} is valid.")
    except InvalidAgeError as e:
        print(e)
```

```
Age 25 is valid.
Invalid age: -5. Age must be between 0 and 120.
Invalid age: 130. Age must be between 0 and 120.
```

```
Age 50 is valid.
Age 0 is valid.
Age 120 is valid.
```

0.1.4 Problem Statement 4:-

Nested Exception Handling:
Write a function nested_exception_handling() that performs the following:
Tries to open a file and read an integer from it.
Catches exceptions for file-related errors (`FileNotFoundError`, `IOError`).
Within the same try block, convert the read value to an integer and handle potential ValueError.
Print specific error messages for each exception and ensure the file is closed properly using a final block.

[4]:
```python
def nested_exception_handling(filename):
    try:
        file = open(filename, 'r')
        try:
            data = file.read()
            number = int(data)
            print(f"Read integer: {number}")
        except ValueError:
            print("Error: The file does not contain a valid integer.")
        finally:
            file.close()
    except FileNotFoundError:
        print("Error: File not found.")
    except IOError:
        print("Error: An IOError occurred.")

filename = input("Enter the filename: ")
nested_exception_handling(filename)
```

```
Enter the filename:  sample.txt
```

```
Error: The file does not contain a valid integer.
```

[ ]:
```

```