Exception Handling

try-except Example:

```
try:
   # Try to execute this code block
   result = 10 / 0
except ZeroDivisionError:
   # Handle the specific exception
   print("Error: Division by zero!")
```

Assignment:

Write a function that takes two numbers as input from the user and divides them. Handle the **ZeroDivisionError** if the user attempts to divide by zero.

Solution –

```
def divide_numbers():
   try:
      num1 = float(input("Enter the first number: "))
      num2 = float(input("Enter the second number: "))
      result = num1 / num2
   except ValueError:
      print("Error: Please enter valid numbers.")
   except ZeroDivisionError:
      print("Error: Division by zero!")
   else:
      print("Result:", result)


divide_numbers()
```

try-except-else Example:

```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
except ValueError:
    print("Error: Please enter valid numbers.")
except ZeroDivisionError:
    print("Error: Division by zero!")
else:
    print("Result:", result)
```

Assignment:

Enhance the previous assignment by adding an **else** clause to print a message only if the division is successful.

```
def divide_numbers_with_message():
    try:
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))
        result = num1 / num2
    except ValueError:
        print("Error: Please enter valid numbers.")
    except ZeroDivisionError:
        print("Error: Division by zero!")
    else:
        print("Result:", result)
        print("Division successful!")
```

```
divide_numbers_with_message()
```

try-finally Example:

```
try:
    file = open("example.txt", "r")
    content = file.read()
    # Intentionally causing an error
    result = 10 / 0
except ZeroDivisionError:
    print("Error: Division by zero!")
finally:
    # This block will be executed regardless of whether an exception occurred
    file.close()
    print("File closed.")
```

Assignment:

Write a function that opens a file, reads its content, and prints the content. Ensure that the file is closed, even if an exception occurs during the process.

Solution –

```
def read_file_content(file_path):
    try:
        file = open(file_path, "r")
        content = file.read()
        # Intentionally causing an error
        result = 10 / 0
    except ZeroDivisionError:
        print("Error: Division by zero!")
    finally:
        # This block will be executed regardless of whether an exception occurred
        file.close()
        print("File closed.")

file_path = "example.txt"  # Replace with the actual file path
read_file_content(file_path)
```