Errors and Exceptions

What are Exceptions?

- Exceptions are errors that occur during the execution of a program. When Python encounters an error, it raises an exception. If the exception is not handled, the program will terminate abruptly.

Common types of exceptions

- `ZeroDivisionError`: Division by zero.
- `TypeError`: Operation applied to an object of inappropriate type.
- `ValueError`: Function receives an argument of the correct type but inappropriate value.
- `FileNotFoundError`: File operation fails because the file does not exist.

Basic Exception Handling

- Python provides a way to handle exceptions using the `try`, `except`, `else`, and `finally` blocks.

```
try:
    # Code that might raise an exception
    risky_code()
except ExceptionType:
    # Code that runs if the exception occurs
    handle_exception()
else:
    # Code that runs if no exception occurs
    no_exception_code()
finally:
    # Code that runs no matter what (exception or not)
    always_run_code()
```

Handling Multiple Exceptions

- You can handle multiple exceptions by specifying multiple except blocks:

```
try:
    value = int(input("Enter a number: "))
    result = 10 / value
except ZeroDivisionError:
    print("Error: Cannot divide by zero.")
except ValueError:
    print("Error: Invalid input. Please enter an integer.")
```

Raising Exceptions

- You can raise exceptions manually using the `raise` keyword:

Custom Exceptions

- You can define your own exceptions by creating a new exception class that inherits from the `Exception` class:

Lets solve some Questions

- **Divide Two Numbers with Exception Handling**
  - Write a function that takes two numbers as input and returns their division. Include exception handling for division by zero and invalid input types.

- **Custom Exception for Negative Age**
  - Define a custom exception `NegativeAgeError`. Write a function that takes an age as input and raises `NegativeAgeError` if the age is negative.

Problems

- **Multiple Exceptions in List Operations**
  - Write a function that takes a list and an index as input and returns the element at that index. Handle exceptions for index out of range and invalid index type

- **User Input Validation**
  - Write a program that asks the user to enter a positive integer. Raise a `ValueError` if the input is not a positive integer and handle this exception.