```
"""
Error / Bug:
Programming errors, often referred to as "bugs," are problems
within a program's code that prevent it from functioning as intended.
These errors can manifest in various ways and are generally categorized into three
main types:

There are three types of Errors:
1. Compile Time Errors:
   These errors occur when the code violates the grammatical rules of
   the programming language. The compiler or interpreter identifies
   these errors during the compilation or parsing phase,
   preventing the program from being built or executed.
   Examples include typos, missing semicolons,
   incorrect keyword usage, or unclosed parentheses.
   (Syntax Errors)

2. Runtime Errors:
   These errors occur during the execution of a program,
   even if the syntax is correct. They arise from situations that the
   program cannot handle, leading to unexpected termination or crashes.
   Common examples include:
Division by zero: Attempting to divide a number by zero.
Memory access violations: Trying to access memory that the program does not have
permission to use.
File not found errors: Attempting to open a file that does not exist at the
specified path.

3. Logical Errors:
These errors are the most challenging to detect because they do not
cause the program to crash or produce syntax errors.
Instead, the program runs successfully but produces incorrect or
unexpected output. Logical errors stem from flaws in the
program's algorithm or reasoning, such as:
Incorrect calculations: Using the wrong formula or order of operations.
Flawed conditional logic: Conditions that do not evaluate as intended, leading to
incorrect program flow.
Off-by-one errors: Looping one too many or one too few times.

Exceptions:
Runtime Errors are known as Exceptions.
"""
# Compile Time Errors:
# print "Hello World!!!"


# Logical Errors
def add(a, b):
    return a * b


print(add(10, 20)) # 30


# Runtime Errors:
"""
1. Division By Zero
a = int(input("Enter a Number: "))
b = int(input("Enter another Number: "))
```

```
print(a / b)

2. Index Error:
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
i = int(input("Enter which index you want: "))
print(a[i])

3. FileNotFound
with open("./wekjfsdfksf.txt", "r") as fp:
    data = fp.read()
"""


# Handling Exceptions
"""
a = int(input("Enter a Number: "))
b = int(input("Enter another Number: "))
if b != 0:
    print(a / b)
else:
    print("Cannot perform Division by Zero...")

a = int(input("Enter a Number: "))
b = 0
while b == 0:
    b = int(input("Enter another Number: "))
print(a / b)

a = int(input("Enter a Number: "))
while True:
    b = int(input("Enter another Number: "))
    if b != 0:
        break

print(a / b)


# Index Error:
def is_index_valid(n, i):
    return ((i >= 0) and (i < n)) or ((i <= -1) and (i >= -n))

a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
i = int(input("Enter which index you want: "))
if is_index_valid(len(a), i):
    print(a[i])
else:
    print(f"Please provide indices between {0} and {len(a) - 1} or {-1} and {-
len(a)}")


a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
while True:
    i = int(input("Enter which index you want: "))
    if is_index_valid(len(a), i):
        break
    print(f"Please provide indices between {0} and {len(a) - 1} or {-1} and {-
len(a)}")
print(a[i])
```

```
# File Not Found Error:
import os

file_name = "./isjeskjdsd"
if os.path.exists(file_name):
    with open(file_name, "r") as fp:
        data = fp.read()
else:
    print("The File Doesn't exist")
"""


"""
Exception Handling
try:
    The code that might create an exception
    is placed inside the try block.

    The try block must follow an an except or finally block.

    Note:
    When an exception occurs, all the lines below that line inside the try block
will be
    skipped and the control immediately goes to the except block.
except:
    If any exception is raised by the code inside the try block,
    then the code inside the except block will be executed.

    Note:
    The Except block will be executed only if the code inside the
    try block raises an exception.
    The Except block will be executed only if the exception occurs
    in the code which is placed inside the try block.
    There can be multiple except blocks after one try block.
    The exception will be caught by the appropriate except block.
    Generic (or) Broad Exceptions should be placed at the bottom.
else:
    The Code inside the else block will be executed if there is no exception
raised.
    If an exception is raised, the code inside the else block will not be executed
and will be skipped.
    Usage:
    We use else block to make the code execute only after there is no exception
    (After all the validations)
raise:
    To Create an exception object and throw.
finally:
    The Code inside the finally block will execute irrespective of exception.
    Usage:
    Generally used for clean up operations, closing files, db connections etc.


# Syntax:
try:
    # Code that might throw an exception.
except:
   # Code if exception occurs
```

```
# Example:
a = int(input("Enter a Number: "))
b = int(input("Enter another Number: "))
try:
    print(a / b)
except:
    print("Exception Occurred...")
"""


"""
try:
    a = int(input("Enter a Number: "))
    b = int(input("Enter another Number: "))
    print(a / b)
except ZeroDivisionError as zde:
    print("Exception Occurred because of Division By Zero...")
    print(zde)
except ValueError as ve:
    print("Exception Occured because of Value Error...")
    print(ve)
"""


"""
a = input("Enter a Number: ")
while not a.isnumeric():
    print("Enter a Number only...")
    a = input("Enter a Number: ")


b = input("Enter another Number: ")
while not b.isnumeric():
    print("Enter Number only...")
    b = input("Enter another Number: ")

a = int(a)
b = int(b)

if b != 0:
    print(a / b)
else:
    print("Division by zero is not possible...")


try:
    a = 10
"""


"""
try:
    a = int(input("Enter a Number: "))
    b = int(input("Enter another Number: "))
    print(a / b)
except Exception as e:
    print(e)

print(dir(locals()["__builtins__"]))
```

```
"""


"""
try:
    a = int(input("Enter a Number: "))
    b = int(input("Enter another Number: "))
    print(a / b)

    l = [1, 2, 3, 4, 5]
    # print(l[5])

    d = {1: "One", 2: "Two"}
    print(d[3])
except Exception as e:
    print("Inside Exception")
    print(e)
except ZeroDivisionError as zde:
    print("Inside ZeroDivisionError")
    print("Exception Occurred because of Division By Zero...")
    print(zde)
except ValueError as ve:
    print("Inside ValueError")
    print("Exception Occured because of Value Error...")
    print(ve)
except IndexError as ie:
    print("Inside IndexError")
    print("Index out of range")
    print(ie)
except KeyError as ke:
    print("Inside KeyError")
    print("Key is not present in the Dictionary")
    print(ke)
"""

"""
try:
    a = int(input("Enter a Number: "))
    b = int(input("Enter another Number: "))
    print(a / b)

    l = [1, 2, 3, 4, 5]
    # print(l[5])

    d = {1: "One", 2: "Two"}
    print(d[3])
except ZeroDivisionError as zde:
    print("Inside ZeroDivisionError")
    print("Exception Occurred because of Division By Zero...")
    print(zde)
except ValueError as ve:
    print("Inside ValueError")
    print("Exception Occured because of Value Error...")
    print(ve)
except IndexError as ie:
    print("Inside IndexError")
    print("Index out of range")
    print(ie)
except KeyError as ke:
```

```
    print("Inside KeyError")
    print("Key is not present in the Dictionary")
    print(ke)
except Exception as e:
    print("Inside Exception")
    print(e)
"""


# else
try:
    l = [1, 2, 3, 4, 5, 6, 7]
    i = int(input("Enter Which Index you want: "))
    print(l[i])
except ValueError as ve:
    print(ve)
except IndexError as ie:
    print(ie)
else:
    print("Inside Else...")
    print("No Exception Occured...")


# finally
try:
    l = [1, 2, 3, 4, 5, 6, 7]
    i = int(input("Enter Which Index you want: "))
    print(l[i])
except ValueError as ve:
    print(ve)
except IndexError as ie:
    print(ie)
finally:
    print("Inside Finally...")


# raise
# Eligibility to Vote:
try:
    age = int(input("Enter Age of the candidate: "))
    if age < 18:
        raise ValueError("Age should be greater than or equal to 18 for voting...")
    print("Candidate is eligible to Vote...")
except ValueError as ve:
    print(ve)


# Code below the exception in the try block:
try:
    l = [1, 2, 3, 4, 5]
    print(l[10])
    print("Line 334")
except IndexError as ie:
    print(ie)



"""
Custom Exceptions
```

```
Syntax:
class <New_Exception_Name>(Exception):
    # Body
"""
class MyCustomException(Exception):
    pass


# Sample Example
class BankingError(Exception):
    pass


class InvalidAmountError(BankingError):
    pass

class InsufficientFundsError(BankingError):
    pass


class BankAccount:
    def __init__(self, account_number, balance):
        self.account_number = account_number
        self.balance = balance


    def deposit(self, deposit_amount):
        if deposit_amount <= 0:
            raise InvalidAmountError
        self.balance += deposit_amount

    def withdrawl(self, withdrawl_amount):
        if withdrawl_amount > self.balance:
            raise InsufficientFundsError
        self.balance -= withdrawl_amount

    def display_balance(self):
        print(f"Current Balance is {self.balance}")


p1 = BankAccount(12345, 10000)
try:
    print("Deposit")
    p1.deposit(10000)
    p1.display_balance()

    print("Withdrwaing Amount")
    p1.withdrawl(50000)
    p1.display_balance()
except InvalidAmountError as iae:
    print("The Amount you entered to deposit is invalid..\n Please enter a valid
amount...")
    print(iae)
except InsufficientFundsError as ife:
    print("Your Balance is less than the amount that you are trying to
Withdraw...")
    print(ife)
```

```
# try, except, else, finally, raise
try:
    print("Hello World!!!")
    print(10 / 0)
    print("After Division By Zero")
    print("Line 409")
except:
    print("Exception Occured...")


def foo():
    return 10


try:
    print("Hello World!!!")
    print(10 / 10)
    print("After Division By Zero")
    print("Line 409")
    # print(kdrje)
    # print("Python" + 10)
    result = foo()
    # result()
    # len(455)
except ZeroDivisionError as zde:
    print("Exception Occured...")
    print(zde)
except ValueError as ve:
    print(ve)
except TypeError as te:
    print("Type Error Occured...")
    print(te)
except Exception as e:
    print(e)
```