Syntax Error

```
a = 10
if a > 5:
print(a) #Syntax error, so program won't execute at all.
else:
    print("Enter proper number.")
```

```
  Cell In[1], line 3
    print(a) #Syntax error, so program won't execute at all.
    ^
IndentationError: expected an indented block after 'if' statement on
line 2
```

Runtime Error (Exception)

```
a = int(input("Enter first number : "))
b = int(input("Enter second number : "))
c = a // b # if b=0 ZeroDivisionError is raised, and statements after
this line won't be executed
print(c)
```

```
Enter first number :  10
Enter second number :  0

----------------------------------------------------------------------
-----
ZeroDivisionError                         Traceback (most recent call
last)
Cell In[4], line 3
      1 a = int(input("Enter first number : "))
      2 b = int(input("Enter second number : "))
----> 3 c = a // b # if b=0 ZeroDivisionError is raised, and
statements after this line won't be executed
      4 print(c)

ZeroDivisionError: integer division or modulo by zero
```

Exception Handling using try except

```
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except:
    print("Some Error")
```

```
Enter first number :  10
Enter second number :  0

Some Error
```

```python
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except:
    print("Some Error")
```

```
Enter first number :  10
Enter second number :  2

5
```

```python
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except ZeroDivisionError: #only handles ZerZeroDivisionError
    print("Some Error")
```

```
Enter first number :  10
Enter second number :  abc

----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[12], line 3
      1 try:
      2     a = int(input("Enter first number : "))
----> 3     b = int(input("Enter second number : "))
      4     c = a // b
      5     print(c)

ValueError: invalid literal for int() with base 10: 'abc'
```

```python
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except ZeroDivisionError:
    print("Can not divide a number by 0.")
```

```
Enter first number :  10
Enter second number :  0

Can not divide a number by 0.
```

```python
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except ZeroDivisionError:
    print("Can not divide a number by 0.")
except ValueError:
    print("Please enter valid integer number, string is not allowed.")
```

```
Enter first number :  10
Enter second number :  abc

Please enter valid integer number, string is not allowed.
```

```python
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
    print(2 + '3')
except ZeroDivisionError:
    print("Can not divide a number by 0.")
except ValueError:
    print("Please enter valid integer number, string is not allowed.")
except:
    print("Some Runtime Error.")  #Default Except Block
```

```
Enter first number :  10
Enter second number :  2

5
Some Runtime Error.
```

Exception Arguments

```python
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except (ZeroDivisionError, ValueError) as err:
    print(type(err))
    print(err)
    print(err.args)
    print(err.args[0])
    print(str(err))
    print(err.__str__())
```

```
Enter first number :  10
Enter second number :  abc

<class 'ValueError'>
invalid literal for int() with base 10: 'abc'
("invalid literal for int() with base 10: 'abc'",)
invalid literal for int() with base 10: 'abc'
invalid literal for int() with base 10: 'abc'
invalid literal for int() with base 10: 'abc'
```

```
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except Exception as err:  # as Excpetion is the common class of all
non-fatal exceptions
    print(err)
```

```
Enter first number :  10
Enter second number :  abc

invalid literal for int() with base 10: 'abc'
```

```
try:
    a = int(input("Enter first number : "))
    b = int(input("Enter second number : "))
    c = a // b
    print(c)
except Exception as err: # as Excpetion is the common class of all
non-fatal exceptions
    print(err)
```

```
Enter first number :  10
Enter second number :  0

integer division or modulo by zero
```

```
try:
    fp = open("abc1.txt","r")
except FileNotFoundError as err:
    print(err)
print(fp.read()) #NameError is raised as file is not opened. fp is not
created.
fp.close()
```

```
[Errno 2] No such file or directory: 'abc1.txt'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[31], line 5
      3 except FileNotFoundError as err:
      4     print(err)
----> 5 print(fp.read()) #NameError is raised as file is not opened. fp is not created.
      6 fp.close()

NameError: name 'fp' is not defined
```

else finally

```python
try:
    fp = open("abc.txt","r")
except FileNotFoundError as err:
    print(err)
else:
    print(fp.read())
    fp.close()
finally:
    print("This block will always be executed.")
```

```
123
This block will always be executed.
```

```python
try:
    fp = open("abc1.txt","r")
except FileNotFoundError as err:
    print(err)
else:
    print(fp.read())
    fp.close()
finally:
    print("This block will always be executed.")
```

```
[Errno 2] No such file or directory: 'abc1.txt'
This block will always be executed.
```

To print Error Name with Standard Error Message

```python
try:
    print(10/0)
except Exception as err:
    print(type(err).__name__ + " :  " + str(err))
```

```
ZeroDivisionError :  division by zero
```

To get the class hierarchy of Exception Classes

```
import inspect
parent_class =  inspect.getmro(ZeroDivisionError)
print(parent_class)
```

```
(<class 'ZeroDivisionError'>, <class 'ArithmeticError'>, <class
'Exception'>, <class 'BaseException'>, <class 'object'>)
```

```
import inspect
parent_class =  inspect.getmro(ZeroDivisionError)
for i in parent_class[::-1]:
    print(i)
```

```
<class 'object'>
<class 'BaseException'>
<class 'Exception'>
<class 'ArithmeticError'>
<class 'ZeroDivisionError'>
```

To raise the specified exception

```
try:
    raise ZeroDivisionError
except ZeroDivisionError:
    print("some error")
```

```
some error
```

Example : Operating file with exception handling

```
try:
    fp = open("abc.txt","r")
    ans = 5 + fp.read()
except Exception as err:
    print(type(err).__name__ + ":" + str(err))
else:
    print(ans)
finally:
    fp.close()
```

```
TypeError:unsupported operand type(s) for +: 'int' and 'str'
```

```
fp.closed
```

```
True
```

User Defined Exception

```
class FiveDivisionError(Exception):
    pass
```

```python
try:
    a = int(input("enter first number : "))
    b = int(input("Enter second number : "))
    if b != 5:
        print(a // b)
    else:
        raise FiveDivisionError

except FiveDivisionError:
    print("ERROR")
```

```
enter first number :  10
Enter second number :  2
```

```
5
```

```python
class FiveDivisionError(Exception):
    pass

try:
    a = int(input("enter first number : "))
    b = int(input("Enter second number : "))
    if b != 5:
        print(a // b)
    else:
        raise FiveDivisionError

except FiveDivisionError:
    print("ERROR")
```

```
enter first number :  10
Enter second number :  5
```

```
ERROR
```

```python
class FiveDivisionError(Exception):
    def __init__(self, msg):
        self.msg = msg

try:
    a = int(input("enter first number : "))
    b = int(input("Enter second number : "))
    if b != 5:
        print(a // b)
    else:
        raise FiveDivisionError("can not divide by five.")

except FiveDivisionError as err:
    print(err)
```

```
enter first number :  10
Enter second number :  5
```

```
can not divide by five.
```

```python
class NegativeNumberError(Exception):
    def __init__(self, msg):
        self.msg = msg

try:
    a = int(input("Enter a number: "))
    if  a >= 0:
        print(a)
    else:
        raise NegativeNumberError("number can not be negative.")
except NegativeNumberError as err:
    print(err)
```

```
Enter a number:  -6
```

```
number can not be negative.
```