In [1]:

```
num1 = int(input("Enter number: "))
num2 = int(input("Enter number: "))
res= num1/num2
print("The result is" ,res)
```

```
Enter number: 5
Enter number: 0

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Input In [1], in <cell line: 3>()
      1 num1 = int(input("Enter number: "))
      2 num2 = int(input("Enter number: "))
----> 3 res= num1/num2
      4 print("The result is" ,c)

ZeroDivisionError: division by zero
```

In [5]:

```
try:
    num1 = int(input("Enter number: "))
    num2 = int(input("Enter number: "))
    res= num1/num2
except ZeroDivisionError as z:
    print(z)
except ValueError as v:
    print(v)
else:
    print("The result is" ,res)
finally:
    print("Completed")
```

```
Enter number: 5
Enter number: g
invalid literal for int() with base 10: 'g'
Completed
```

In [7]:

```
k = [1,2,3,4]
try:
    inp = int(input("Enter the index: "))
    temp = k[inp]
except IndexError as E:
    print(E)
else:
    print("The Element is",temp)
finally:
    print("Done")
```

```
Enter the index: 5
list index out of range
Done
```

Raise an Error

In [8]:

```
try:
    a = 10/2
    raise NameError
    raise ZeroDivisionError
except NameError:
    print("Name Error")
except ValueError:
    print("Value Error")
else:
    print(a)
```

```
Name Error
```

In [10]:

```
try:
    a = 10/2
    raise ZeroDivisionError
except NameError:
    print("Name Error")
except ValueError:
    print("Value Error")
except:
    print("Some Error")
else:
    print(a)
```

```
Some Error
```

```
Write a program that prompts the user to enter a number if the number is positive or
0,print it.
otherwise raise an exception
```

In [15]:

```
try:
    num = int(input("Enter Number: "))
    if num>= 0:
        print(num)
    else:
        raise ValueError("Negative number not allowed.")
except ValueError as v:
    print(v)
```

```
Enter Number: 5
5
```

```
Write a number game program and ask the user to enter a number. If the number is
greater than number to be guessed raise a value to large exception. If the value is
smaller the number to be guessed then raise a value to small exception and prompt
the user to enter again quit the program only when the user enters the correct
number
```

In [20]:

```
class ValueToLarge(Exception):#user defined function will extract the feature from exce
    def display(self): #if the function is inside class the default parameter is self
        print("The value guess is too large")
class ValueToSmall(Exception):
     def display(self):
        print("The value guess is too small")
key =100
while True:
    try:
        num = int(input("Enter number to be guessed"))
        if num == key:
            print("You have guessed it right")
            break
        if num<key:
            raise ValueToSmall
        elif num>key:
            raise ValueToLarge
    except ValueToSmall as s:
        s.display()#calling the value to small class function
    except ValueToLarge as v:
        v.display()#calling the value to large class function
```

```
Enter number to be guessed52
The value guess is too small
Enter number to be guessed90
The value guess is too small
Enter number to be guessed105
The value guess is too large
Enter number to be guessed50
The value guess is too small
Enter number to be guessed150
The value guess is too large
Enter number to be guessed80
The value guess is too small
Enter number to be guessed95
The value guess is too small
Enter number to be guessed105
The value guess is too large
Enter number to be guessed100
You have guessed it right
```

```
Indentation Error cannot be handled by Exception Handling
```

User Defined Exception

```
In a user defined exception the exception need to be derived from the exception
class
```