Exception Handling in Python

“Handling the runtime errors”

Prepared by - Vinod Kumar Verma, PGT(CS)
, Sachin Bhardwaj PGT(CS)

What are Exceptions?

- an exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions.
- These exceptions can occur for various reasons, such as invalid user input, file not found, or division by zero.

Handling Exceptions

- It is a powerful mechanism to handle the runtime errors so that the normal flow of application can be maintained.
- Since exception abnormally terminate the execution of a program, it is important to handle the exceptions. In Python we use **try and except** block to handle the exception. Another keyword **finally** can also be used to perform cleanup task.

try block

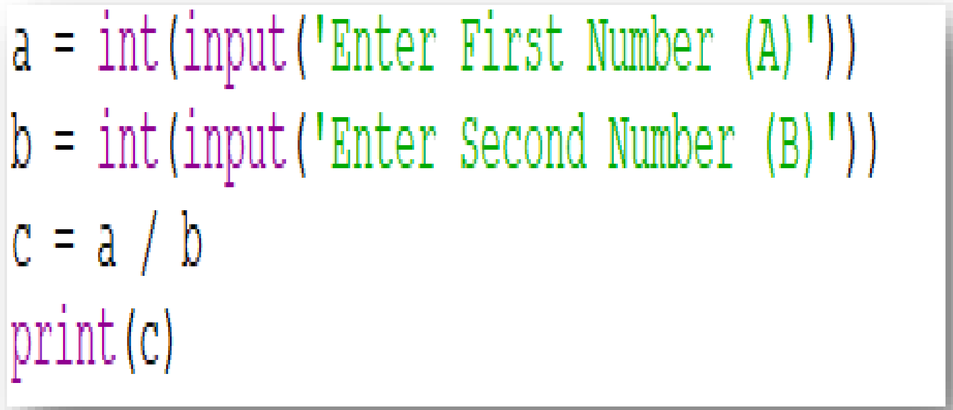
- try block lets you test a block of code for errors.
- We put all those codes inside the try block which may raise runtime errors.
- When try block encounters any runtime error the control is passed to appropriate except block.

except block

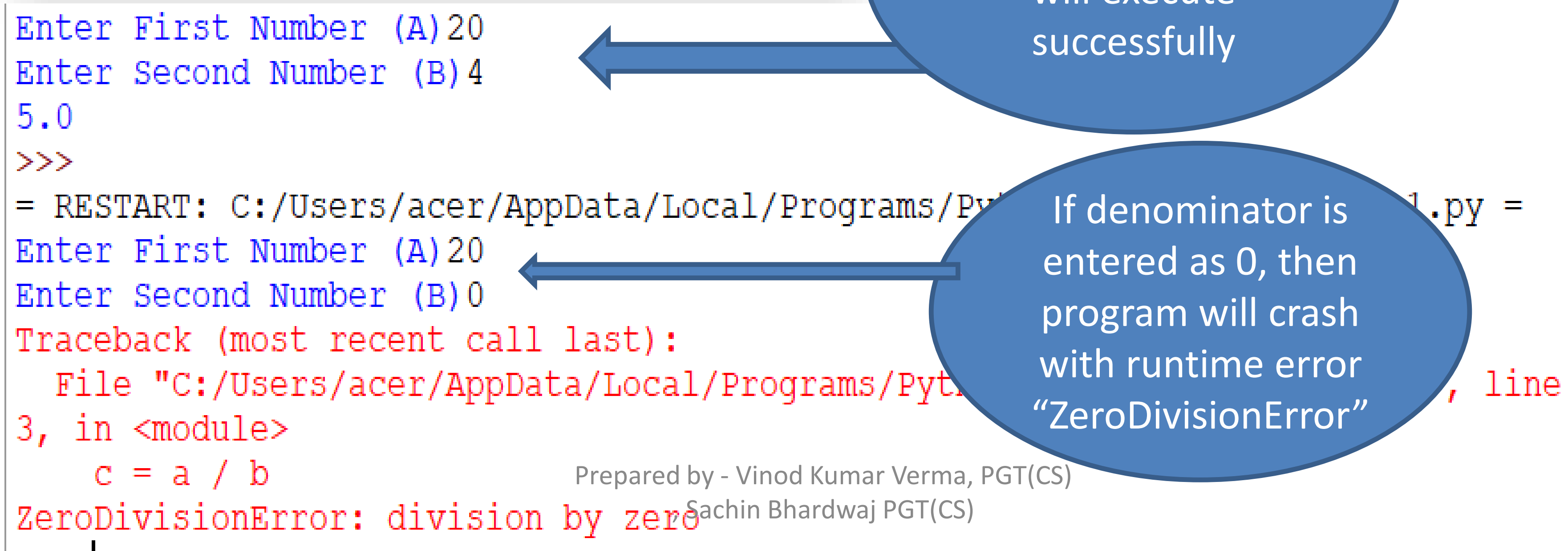
- Except block lets you handle runtime errors
- When any runtime error occurs in try block the control is transferred to except block to handle and maintain the normal execution of program
- Default "**except**" block can handle any type of runtime errors however we can pass the name of runtime error with except block to handle specific type of runtime error.

Example – 1 (Handling Divide by Zero)

Without try..except



```python
a = int(input('Enter First Number (A)'))
b = int(input('Enter Second Number (B)'))
c = a / b
print(c)
```



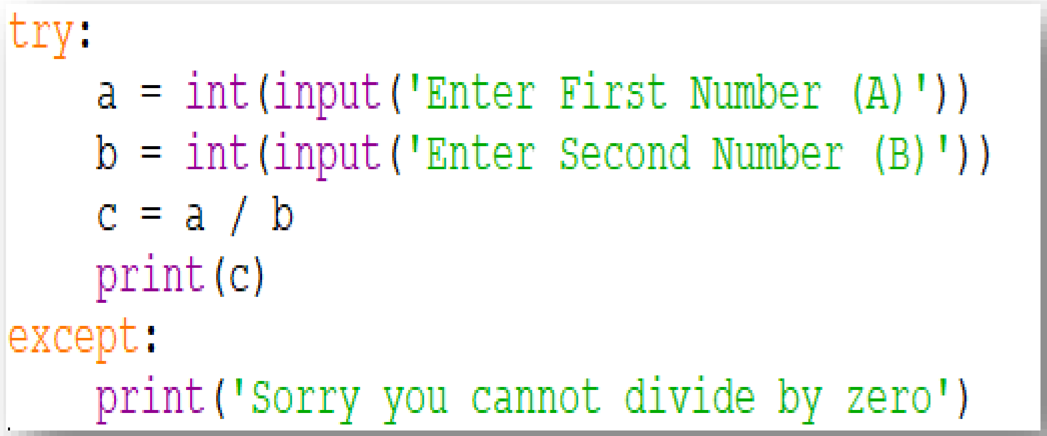
```
Enter First Number (A)20
Enter Second Number (B)4
5.0
>>>
= RESTART: C:/Users/acer/AppData/Local/Programs/Py                  1.py =
Enter First Number (A)20
Enter Second Number (B)0
Traceback (most recent call last):
  File "C:/Users/acer/AppData/Local/Programs/Pyt                    , line
3, in <module>
    c = a / b
ZeroDivisionError: division by zero
```

When we input valid data, program will execute successfully

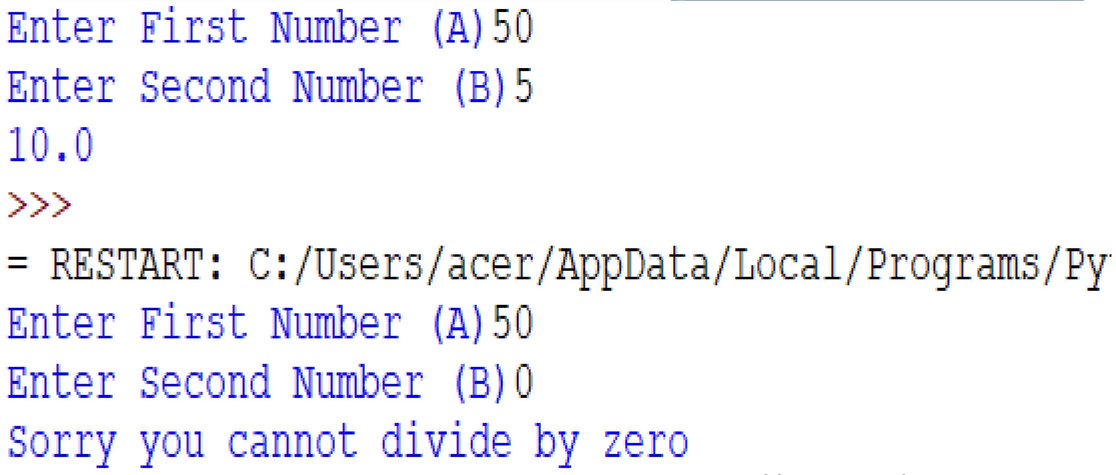
If denominator is entered as 0, then program will crash with runtime error “ZeroDivisionError”

Example – 2 (Handling Divide by Zero)

With try..except



```
try:
    a = int(input('Enter First Number (A)'))
    b = int(input('Enter Second Number (B)'))
    c = a / b
    print(c)
except:
    print('Sorry you cannot divide by zero')
```



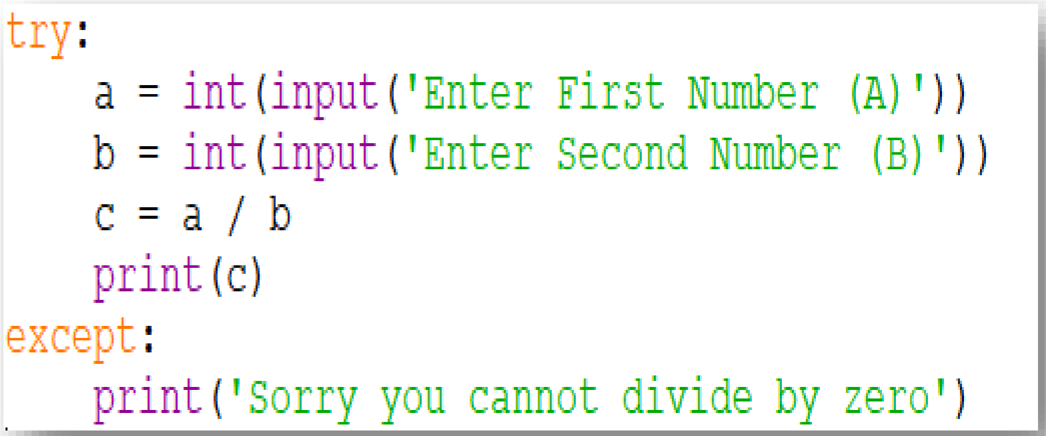
```
Enter First Number (A)50
Enter Second Number (B)5
10.0
>>>
= RESTART: C:/Users/acer/AppData/Local/Programs/Py
Enter First Number (A)50
Enter Second Number (B)0
Sorry you cannot divide by zero
```

When we input valid data, program will execute successfully

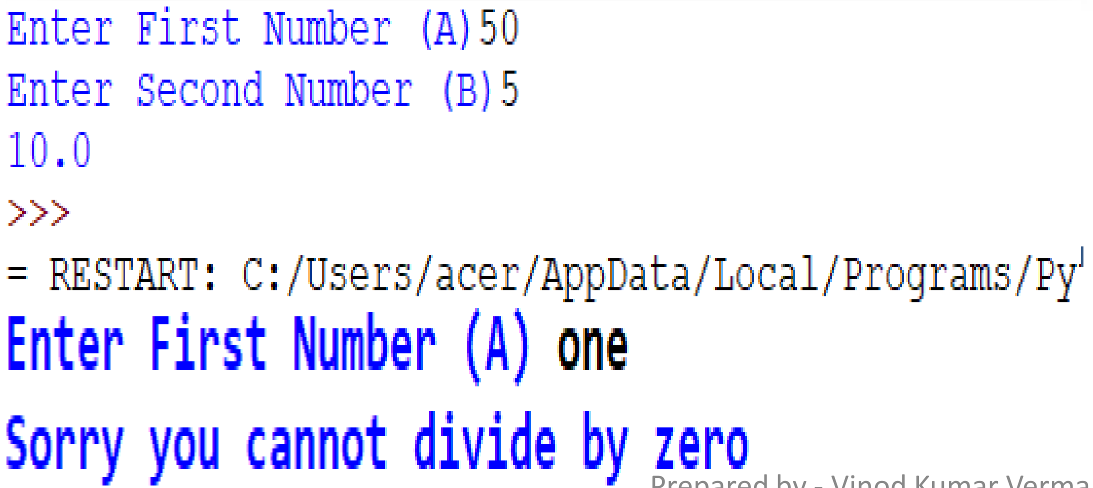
If denominator is entered as 0, then program will not abnormally terminate and error is handled gracefully

Example – 2 (Handling Divide by Zero)

With try..except



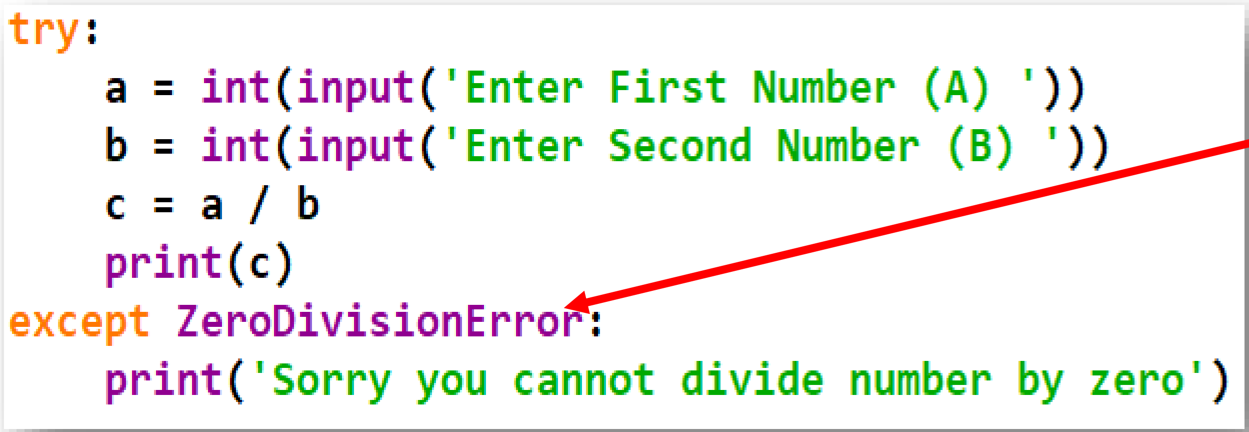
```
try:
    a = int(input('Enter First Number (A)'))
    b = int(input('Enter Second Number (B)'))
    c = a / b
    print(c)
except:
    print('Sorry you cannot divide by zero')
```



```
Enter First Number (A)50
Enter Second Number (B)5
10.0
>>>
= RESTART: C:/Users/acer/AppData/Local/Programs/Py
Enter First Number (A) one
Sorry you cannot divide by zero
```

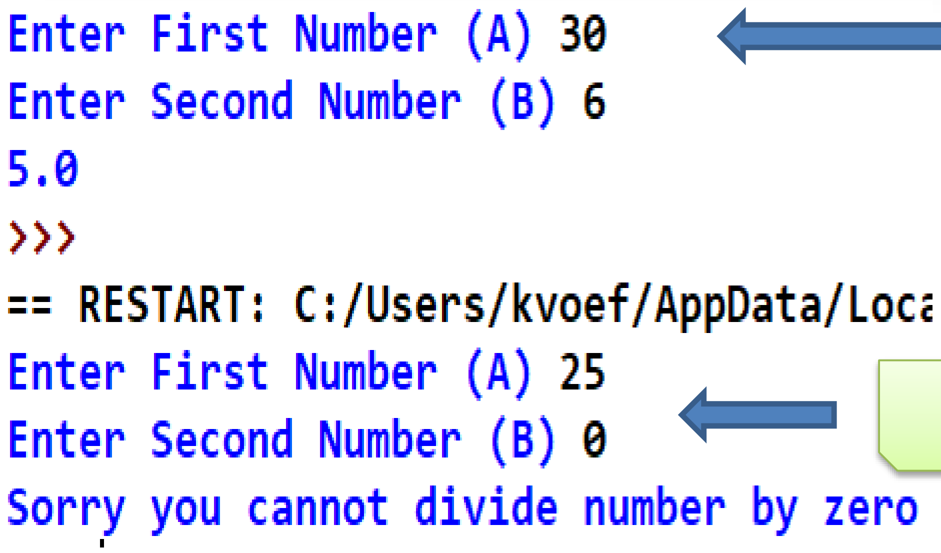
However, here we used **except block** which can handle any type of runtime errors, so if try block raises any other runtime error like ValueError when we input invalid data in variable a or b like string, the same message 'Sorry you cannot divide by zero' will be printed

Example – 3 (Handling specific Exception)



```python
try:
    a = int(input('Enter First Number (A) '))
    b = int(input('Enter Second Number (B) '))
    c = a / b
    print(c)
except ZeroDivisionError:
    print('Sorry you cannot divide number by zero')
```

Name of specific exception is given

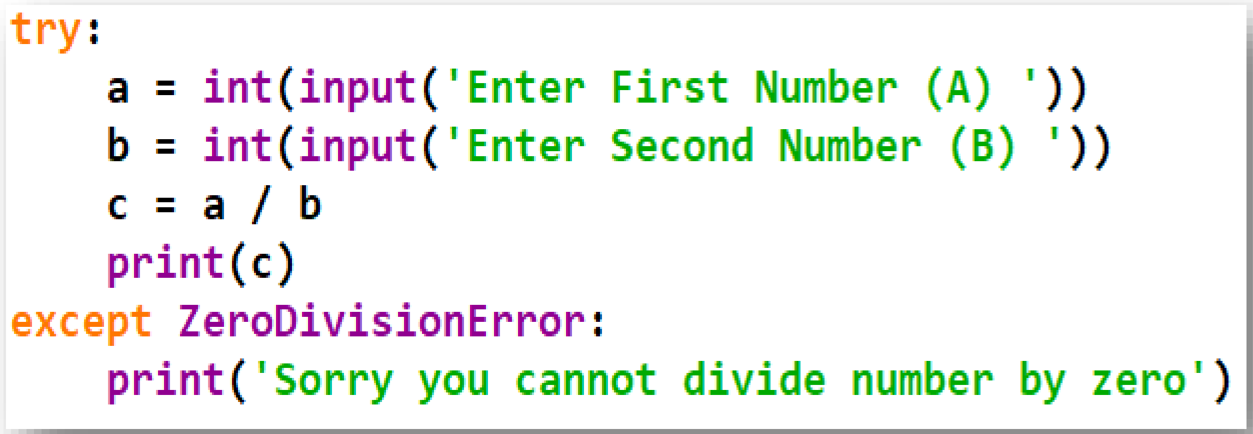


```
Enter First Number (A) 30
Enter Second Number (B) 6
5.0
>>>
== RESTART: C:/Users/kvoef/AppData/Loca
Enter First Number (A) 25
Enter Second Number (B) 0
Sorry you cannot divide number by zero
```

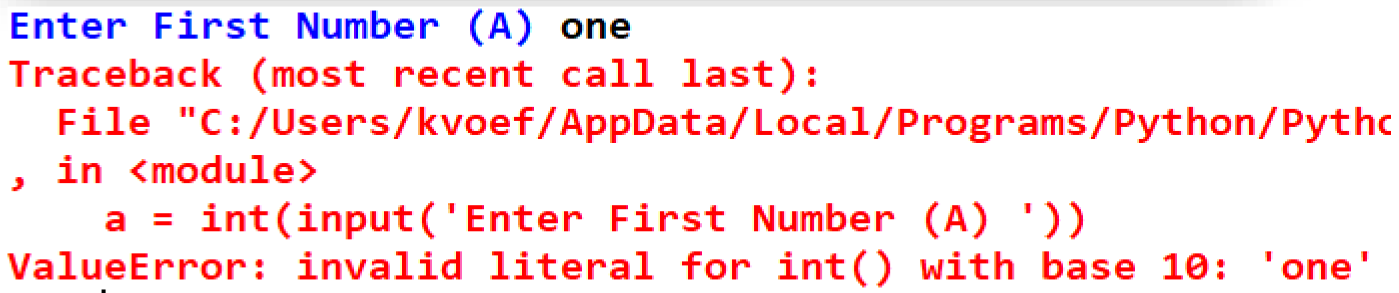
Successful execution with valid data

Divide by 0 error is handled successfully

Example – 3 (Handling specific Exception)



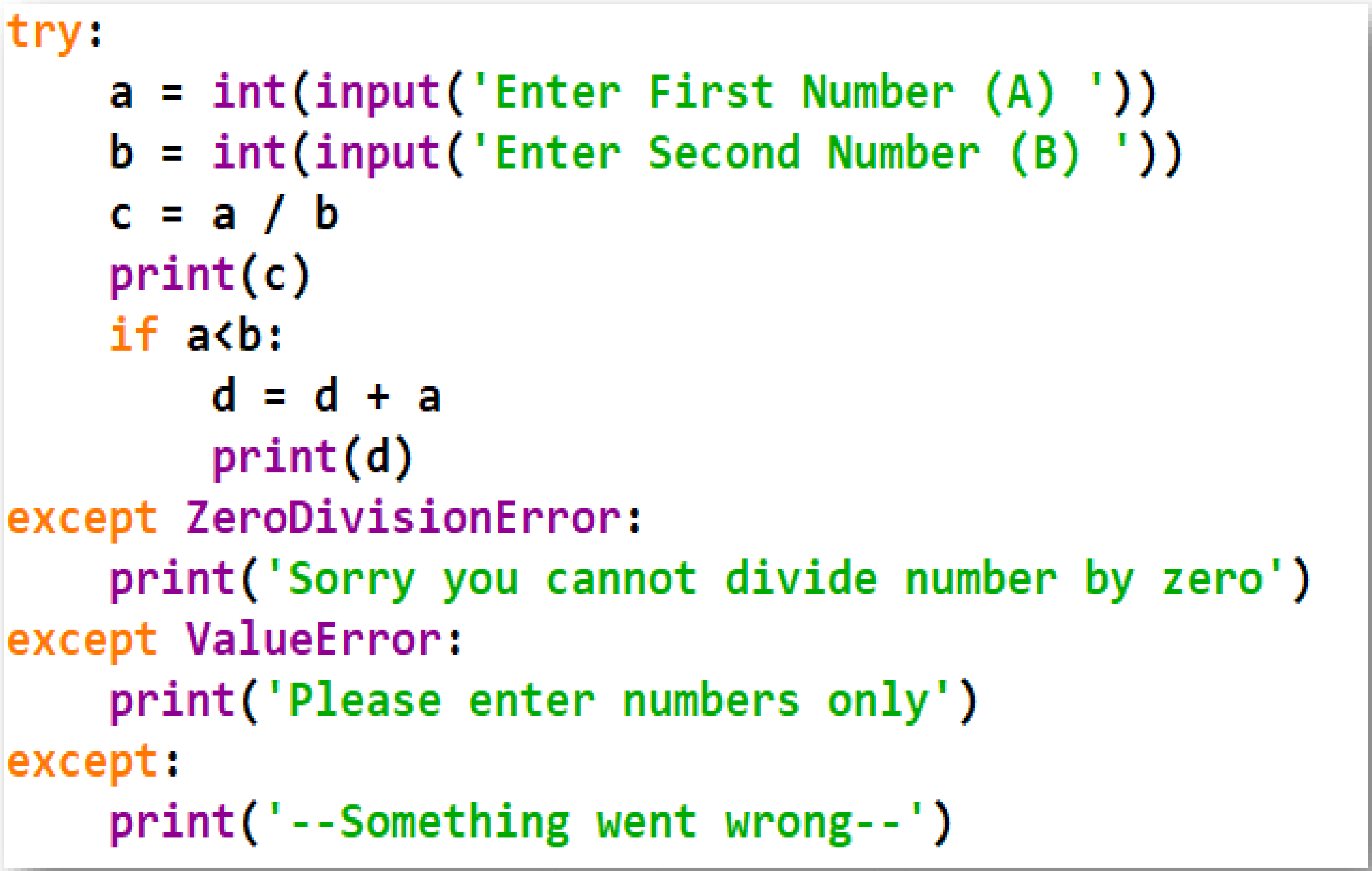
```
try:
    a = int(input('Enter First Number (A) '))
    b = int(input('Enter Second Number (B) '))
    c = a / b
    print(c)
except ZeroDivisionError:
    print('Sorry you cannot divide number by zero')
```



```
Enter First Number (A) one
Traceback (most recent call last):
  File "C:/Users/kvoef/AppData/Local/Programs/Python/Pythc
, in <module>
    a = int(input('Enter First Number (A) '))
ValueError: invalid literal for int() with base 10: 'one'
```

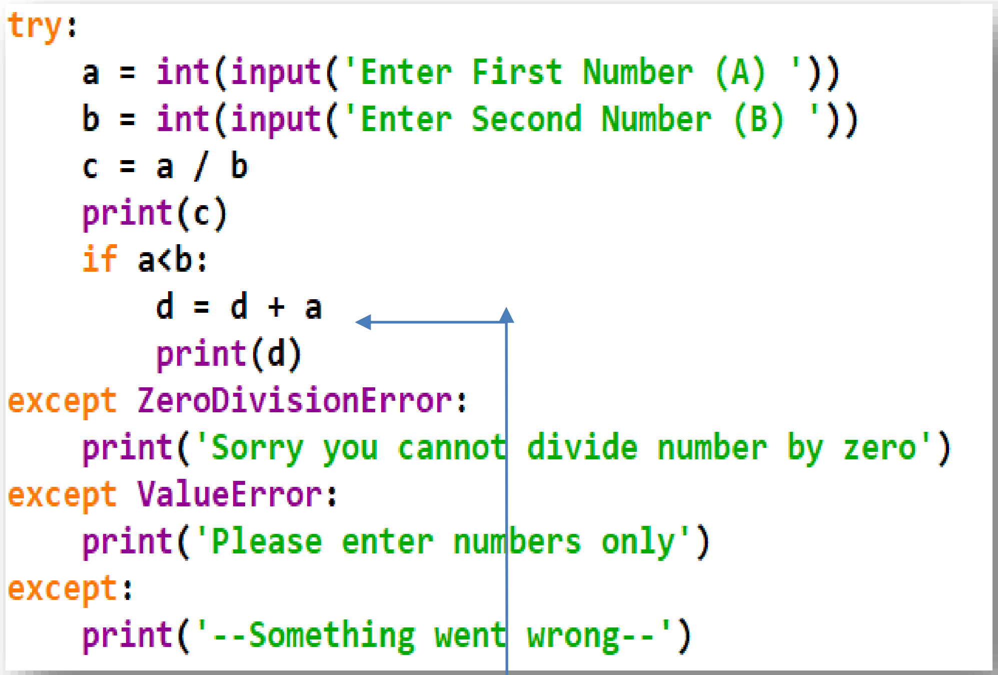
As only ZeroDivisionError exception is handle here, if any other runtime error occurs like if we input string data in place of int data, then program will crash with ValueError Exception

Example – 4 (Multiple Exception Handling)



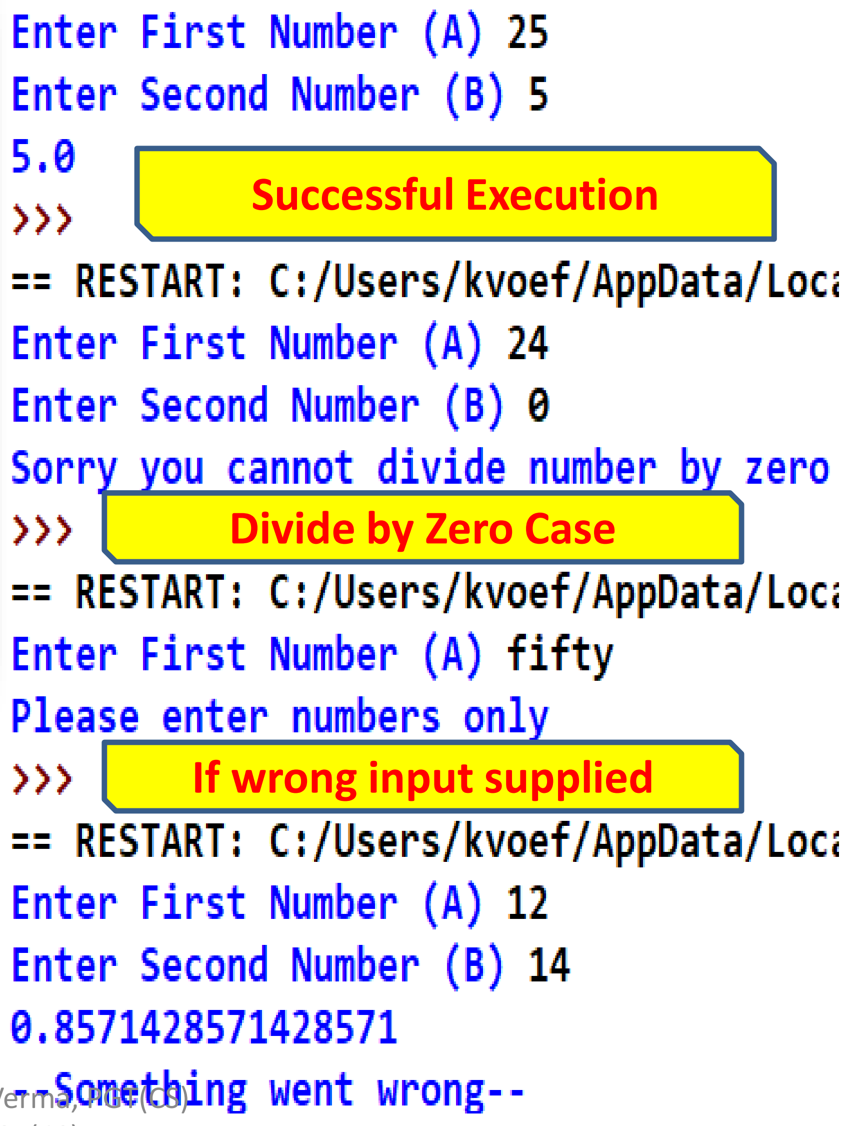
```python
try:
    a = int(input('Enter First Number (A) '))
    b = int(input('Enter Second Number (B) '))
    c = a / b
    print(c)
    if a<b:
        d = d + a
        print(d)
except ZeroDivisionError:
    print('Sorry you cannot divide number by zero')
except ValueError:
    print('Please enter numbers only')
except:
    print('--Something went wrong--')
```

Example – 4 (Multiple Exception Handling)



```
try:
    a = int(input('Enter First Number (A) '))
    b = int(input('Enter Second Number (B) '))
    c = a / b
    print(c)
    if a<b:
        d = d + a
        print(d)
except ZeroDivisionError:
    print('Sorry you cannot divide number by zero')
except ValueError:
    print('Please enter numbers only')
except:
    print('--Something went wrong--')
```

Occurs because variable d is not defined and assigned, NameError Exception will occur and will be handled by except block



```
Enter First Number (A) 25
Enter Second Number (B) 5
5.0
>>>
== RESTART: C:/Users/kvoef/AppData/Loca
Enter First Number (A) 24
Enter Second Number (B) 0
Sorry you cannot divide number by zero
>>>
== RESTART: C:/Users/kvoef/AppData/Loca
Enter First Number (A) fifty
Please enter numbers only
>>>
== RESTART: C:/Users/kvoef/AppData/Loca
Enter First Number (A) 12
Enter Second Number (B) 14
0.8571428571428571
--Something went wrong--
```

Successful Execution

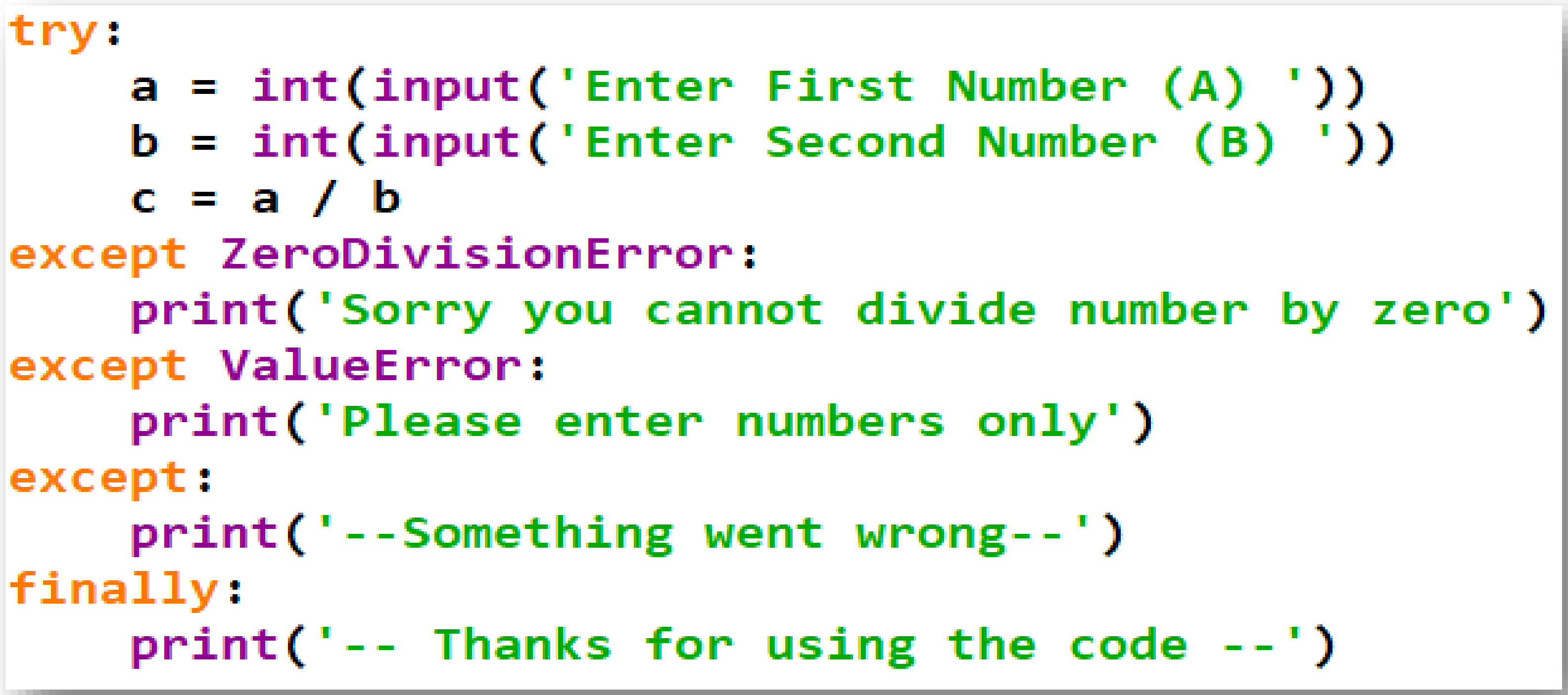
Divide by Zero Case

If wrong input supplied

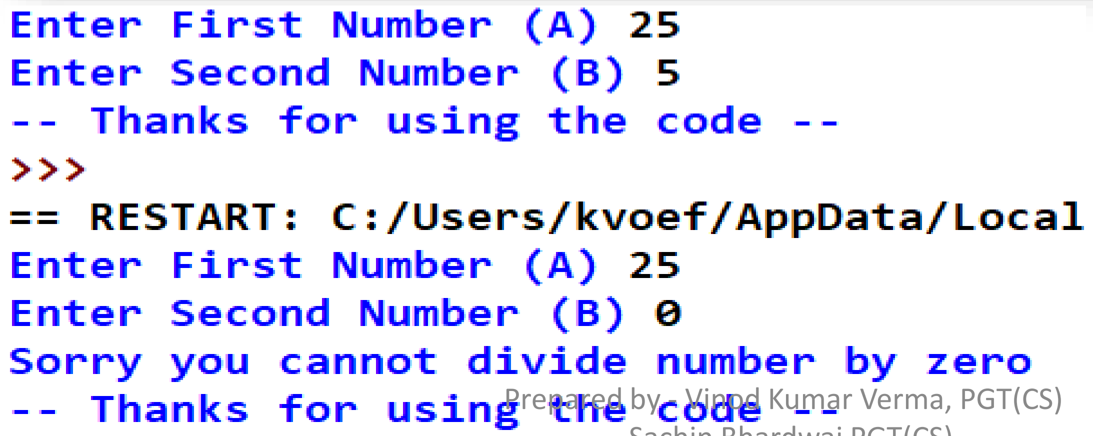
finally block

- The 'finally' block of python is always executed whether an exception in code occurred or not.
- It allows us to perform clean-up actions like releasing resources or closing files, irrespective of the presence of exceptions.

Example (with **finally** block)



```
try:
    a = int(input('Enter First Number (A) '))
    b = int(input('Enter Second Number (B) '))
    c = a / b
except ZeroDivisionError:
    print('Sorry you cannot divide number by zero')
except ValueError:
    print('Please enter numbers only')
except:
    print('--Something went wrong--')
finally:
    print('-- Thanks for using the code --')
```



```
Enter First Number (A) 25
Enter Second Number (B) 5
-- Thanks for using the code --
>>>
== RESTART: C:/Users/kvoef/AppData/Local
Enter First Number (A) 25
Enter Second Number (B) 0
Sorry you cannot divide number by zero
-- Thanks for using the code --
```

Here we can see that code of finally block executed in both cases, whether exception occurred on not

Points to remember

- try block must be followed by either **except** or **finally** block
- except block is not mandatory with try block, we can write either except or finally. However to handle error for successful execution of program except block is must.
- While handling multiple exception the except block must be the last block otherwise all other exception given below except will become unreachable. Syntax error will appear with message 'default except must be last'