Module 16

Errors and Exceptions in Python.

- When we write a faulty code, the execution of program may stop due to the errors
- There can be some places where exceptions can be caused
- Errors are caused by wrong syntax.
- Exceptions are logic based wherein syntax is right but logic isn't. Ex. Zero division

Examples of errors

① Syntax Error

```
if :
```

SyntaxError : invalid syntax

② Name Error

```
print (a)
```

NameError : name 'a' is not defined

③ Type Error

```
"2" + 2
```

TypeError : can only concatenate str (not "int") to str

④ File not found

```
data = open ("abc.txt")
```

FileNotFoundError : [Errno 2] No such file or directory : 'abc.tx

⑤ Indentation Error

```
if True:
```

IndentationError : expected an indented block

⑥ ZeroDivision Error

```
5 / 0
```

ZeroDivisionError : division by zero

```
# Print all list of all possible errors in Python
print (dir (__builtins__))
```

['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', ...]

Try and Except

- try : This block handles the error in your code if any
- except : This block gives the output that you want to show if your code is faulty.

```
a = 5
b = 0
print (a/b)
print ("Hello Uvaraj")
```

ZeroDivisionError : division by zero

```
a = 5
b = 0

try :
    # Put the suspect code in try block
    print (a / b)

except :
    print ("There is an Error you might wanna check")

print ("Hello Uvaraj")
```

There is an Error you might wanna check
Hello Uvaraj

```
a = 5
b = 2
try :
    print (a / b)
except :
    print ("There is an Error you might wanna check")
print ("Hello Uvaraj")
```

2.5
Hello Uvaraj

Except block

- You can tell the kind of error using Exception

```
a = 5
b = 0
try:
    print("2" + 2)
except Exception as e:
    print("error..", e)
print("Hello Uvaraj")
```

error... can only concatenate str (not "int") to str
Hello Uvaraj

```
a = 5
b = 0
try:
    print(5 / 0)
except Exception as e:
    print("error..", e)
print("Hello Uvaraj")
```

error.. division by zero
Hello Uvaraj

Finally block

- This block will be executed in any case
- It is helpful when you want to de-allocate resources
- Like closing a file, or db connection

```
a = 5
b = 0
try:
    print("Open file")
    print(a / b)
    print("close file")
except Exception as e:
    print("Error:", e)
```

Not executed..
i.e. File not closed.
Will cause Memory Leak.

Open file
Error: division by zero

```
a = 5
b = 0
try:
    print ("Open file")
    print (a/b)
except Exception as e:
    print ("Error", e)
    print ("Close file")
```

✓

```
open file
Error : division by zero
close file
```

```
a = 5
b = 2
try:
    print ("open file")
    print (a/b)
except Exception as e:
    print ("Error", e)
    print ("close file")
```

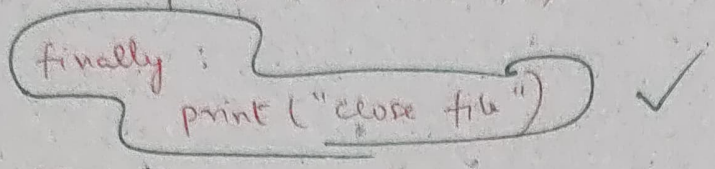
Not executed.
(i) File not closed.
Will cause Memory Leak

```
open file
2.5
```

```
a = 5
b = 2
try:
    print ("open file")
    print (a/b)
    print ("close-file")
except Exception as e:
    print ("Error", e)
    print ("close file")
```

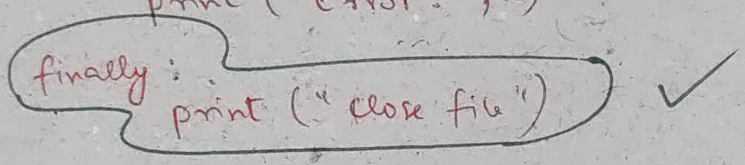
Not an ideal way
of solving this issue..

```
a = 5
b = 2
try :
    print ("open file")
    print (a / b)
except Exception as e :
    print ("Error :", e)
finally :
    print ("close file")
```



Open file
2.5
close file

```
a = 5
b = 0
try :
    print ("open file")
    print (a / b)
except Exception as e :
    print ("Error :", e)
finally :
    print ("close file")
```



Open file
Error : division by zero
close file
