There are 2 stages where error may happen in a program

- During compilation -> Syntax Error
- During execution -> Exceptions

Syntax Error

- Something in the program is not written according to the program grammar.
- Error is raised by the interpreter/compiler
- You can solve it by rectifying the program

```
# Examples of syntax error
print 'hello world'
```

```
  File "<ipython-input-3-4655b84ba7b7>", line 2
    print 'hello world'
                      ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print('hello world')?
```

Other examples of syntax error

- Leaving symbols like colon,brackets
- Misspelling a keyword
- Incorrect indentation
- empty if/else/loops/class/functions

```
a = 5
if a==3
  print('hello')
```

```
  File "<ipython-input-68-efc58c10458d>", line 2
    if a==3
           ^
SyntaxError: invalid syntax
```

```
a = 5
iff a==3:
  print('hello')
```

```
  File "<ipython-input-69-d1e6fae154d5>", line 2
    iff a==3:
        ^
SyntaxError: invalid syntax
```

```
a = 5
if a==3:
print('hello')
```

```
  File "<ipython-input-70-ccc702dc036c>", line 3
    print('hello')
        ^
IndentationError: expected an indented block
```

```
# IndexError
# The IndexError is thrown when trying to access an item at an invalid index.
L = [1,2,3]
L[100]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-71-c90668d2b194> in <module>
      2 # The IndexError is thrown when trying to access an item at an invalid index.
      3 L = [1,2,3]
----> 4 L[100]

IndexError: list index out of range
```

```
# ModuleNotFoundError
# The ModuleNotFoundError is thrown when a module could not be found.
import mathi
math.floor(5.3)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-73-cbdaf00191df> in <module>
      1 # ModuleNotFoundError
      2 # The ModuleNotFoundError is thrown when a module could not be found.
----> 3 import mathi
      4 math.floor(5.3)

ModuleNotFoundError: No module named 'mathi'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

OPEN EXAMPLES

```
# KeyError
# The KeyError is thrown when a key is not found

d = {'name':'nitish'}
d['age']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-74-453afa1c9765> in <module>
      3
      4 d = {'name':'nitish'}
----> 5 d['age']

KeyError: 'age'
```

```
# TypeError
# The TypeError is thrown when an operation or function is applied to an object of an inappropriate type.
1 + 'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-78-2a3eb3f5bb0a> in <module>
      1 # TypeError
      2 # The TypeError is thrown when an operation or function is applied to an object
of an inappropriate type.
----> 3 1 + 'a'

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

```
# ValueError
# The ValueError is thrown when a function's argument is of an inappropriate type.
int('a')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-76-e419d2a084b4> in <module>
      1 # ValueError
      2 # The ValueError is thrown when a function's argument is of an inappropriate
type.
----> 3 int('a')

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

```
# NameError
# The NameError is thrown when an object could not be found.
print(k)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-79-e3e8aaa4ec45> in <module>
      1 # NameError
      2 # The NameError is thrown when an object could not be found.
----> 3 print(k)

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

```
# AttributeError
L = [1,2,3]
L.upper()

# Stacktrace
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-80-dd5a29625ddc> in <module>
      1 # AttributeError
      2 L = [1,2,3]
----> 3 L.upper()

AttributeError: 'list' object has no attribute 'upper'
```

Exceptions

If things go wrong during the execution of the program(runtime). It generally happens when something unforeseen has happened.

- Exceptions are raised by python runtime
- You have to takle is on the fly

Examples

- Memory overflow
- Divide by 0 -> logical error
- Database error

```
# Why is it important to handle exceptions
# how to handle exceptions
# -> Try except block
```

```
# let's create a file
with open('sample.txt','w') as f:
  f.write('hello world')
```

```
# try catch demo
try:
  with open('sample1.txt','r') as f:
    print(f.read())
except:
  print('sorry file not found')
```

```
    sorry file not found
```

```
# catching specific exception
try:
  m=5
  f = open('sample1.txt','r')
  print(f.read())
  print(m)
  print(5/2)
  L = [1,2,3]
  L[100]
except FileNotFoundError:
  print('file not found')
except NameError:
  print('variable not defined')
except ZeroDivisionError:
  print("can't divide by 0")
except Exception as e:
  print(e)
```

```
    [Errno 2] No such file or directory: 'sample1.txt'
```

```
# else
try:
  f = open('sample1.txt','r')
except FileNotFoundError:
  print('file nai mili')
except Exception:
  print('kuch to lafda hai')
else:
  print(f.read())
```

```
    file nai mili
```

```
# finally
# else
try:
  f = open('sample1.txt','r')
except FileNotFoundError:
  print('file nai mili')
except Exception:
  print('kuch to lafda hai')
else:
  print(f.read())
finally:
  print('ye to print hoga hi')
```

```
    file nai mili
    ye to print hoga hi
```

```
# raise Exception
# In Python programming, exceptions are raised when errors occur at runtime.
# We can also manually raise exceptions using the raise keyword.

# We can optionally pass values to the exception to clarify why that exception was raised
```

```
raise ZeroDivisionError('aise hi try kar raha hu')
# Java
# try -> try
# except -> catch
# raise -> throw
```

```
    ---------------------------------------------------------------------------
    ZeroDivisionError                         Traceback (most recent call last)
    <ipython-input-106-5a07d7d89433> in <module>
    ----> 1 raise ZeroDivisionError('aise hi try kar raha hu')

    ZeroDivisionError: aise hi try kar raha hu
```

```
class Bank:

  def __init__(self,balance):
    self.balance = balance

  def withdraw(self,amount):
    if amount < 0:
      raise Exception('amount cannot be -ve')
    if self.balance < amount:
      raise Exception('paise nai hai tere paas')
    self.balance = self.balance - amount

obj = Bank(10000)
try:
  obj.withdraw(15000)
except Exception as e:
  print(e)
else:
  print(obj.balance)
```

```
    paise nai hai tere paas
```

```
class MyException(Exception):
  def __init__(self,message):
    print(message)

class Bank:

  def __init__(self,balance):
    self.balance = balance

  def withdraw(self,amount):
    if amount < 0:
      raise MyException('amount cannot be -ve')
    if self.balance < amount:
      raise MyException('paise nai hai tere paas')
    self.balance = self.balance - amount

obj = Bank(10000)
try:
  obj.withdraw(5000)
except MyException as e:
  pass
else:
  print(obj.balance)
```

```
    5000
```

```
# creating custom exceptions
# exception hierarchy in python
```

```
# simple example
```

```
class SecurityError(Exception):

  def __init__(self,message):
    print(message)

  def logout(self):
    print('logout')

class Google:

  def __init__(self,name,email,password,device):
    self.name = name
    self.email = email
    self.password = password
    self.device = device

  def login(self,email,password,device):
    if device != self.device:
      raise SecurityError('bhai teri to lag gayi')
    if email == self.email and password == self.password:
      print('welcome')
    else:
      print('login error')



obj = Google('nitish','nitish@gmail.com','1234','android')

try:
  obj.login('nitish@gmail.com','1234','windows')
except SecurityError as e:
  e.logout()
else:
  print(obj.name)
finally:
  print('database connection closed')
```