Exceptions

Exceptions: Exceptions are raised when the program is syntactically correct, but the code resultedin an error. This error does not stop the execution of the program but it changes the normal flow of the program.

Exception can be done by following blocks in python:

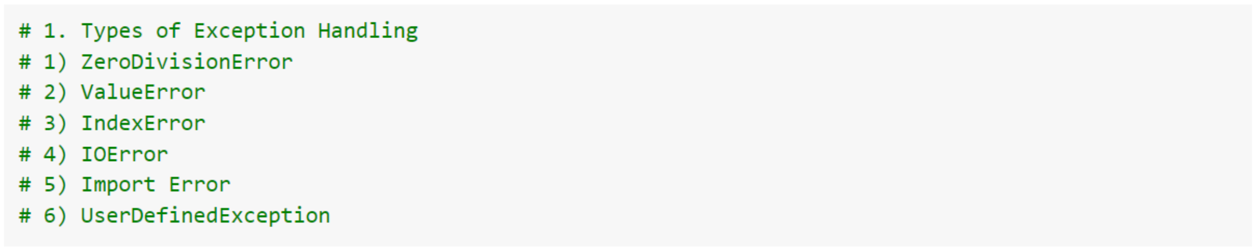
1) Try block: Statements that can raise exceptions are kept inside the try clause
2) Except block: the statements that handle the exception are written inside except clause.
3) Else block: The code enters the else block only if the try clause does not raise an exception.
4) Finally block: The final block always executes after normal termination of try block or after try block terminates due to some exception.
5) Raise statement: Programmers can also forcefully raise exceptions in a program using the raise keyword.
   Once an exception is raised, no further statement in the current block of code is executed
   - raise statement can be used to throw an exception.
   - The syntax of raise statement is:
     raise <exception name>
   - Argument is generally a string that is displayed when the exception is raised

Each and every exception has to be handled by the programmer to avoid the program fromcrashing abruptly.

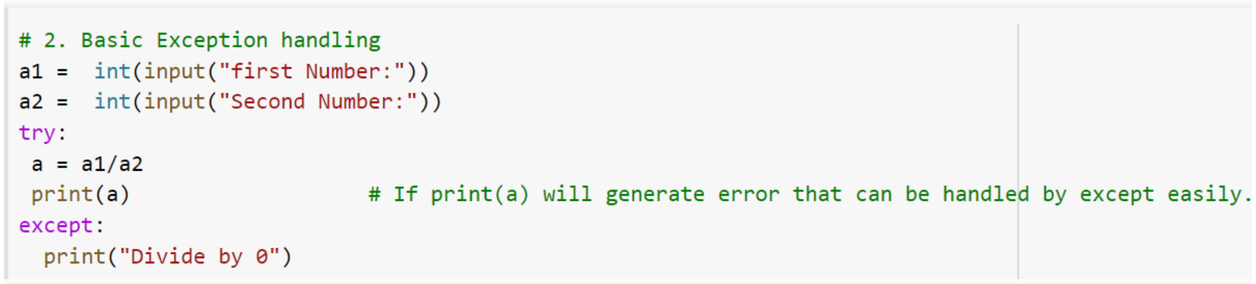
TYPES OF EXCEPTION

ImportError	Raised when the imported module is not found.
IndexError	Raised when the index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.
RuntimeError	Raised when an error does not fall under any other category.
SyntaxError	Raised by parser when syntax error is encountered.
IndentationError	Raised when there is incorrect indentation.
ValueError	Raised when a function gets an argument of correct type but improper value.
ZeroDivisionErro	Raised when the second operand of division or modulo operation is zero.

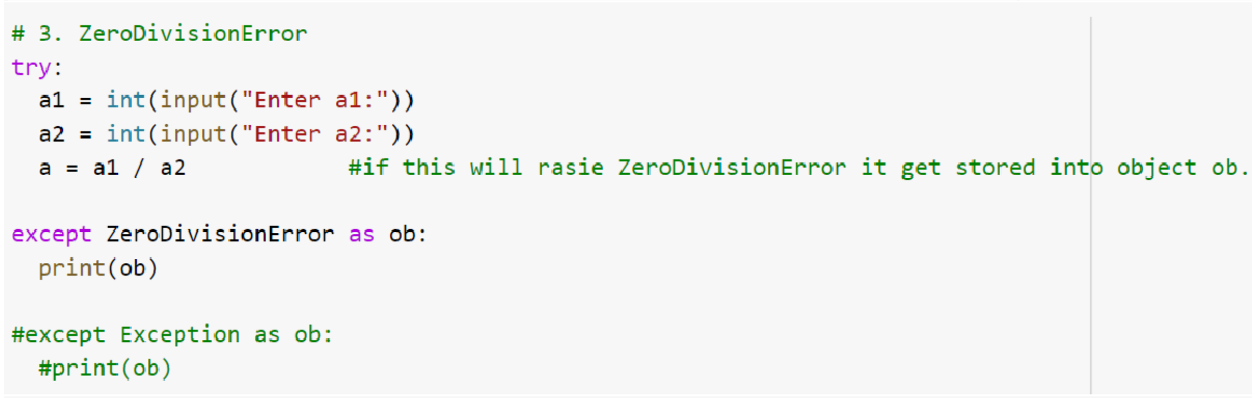
Exception



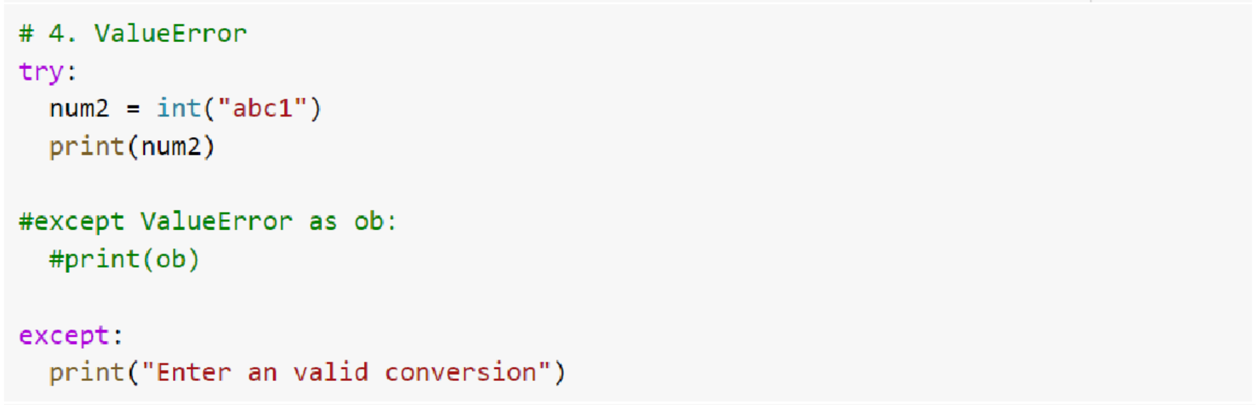
```
# 1. Types of Exception Handling
# 1) ZeroDivisionError
# 2) ValueError
# 3) IndexError
# 4) IOError
# 5) Import Error
# 6) UserDefinedException
```



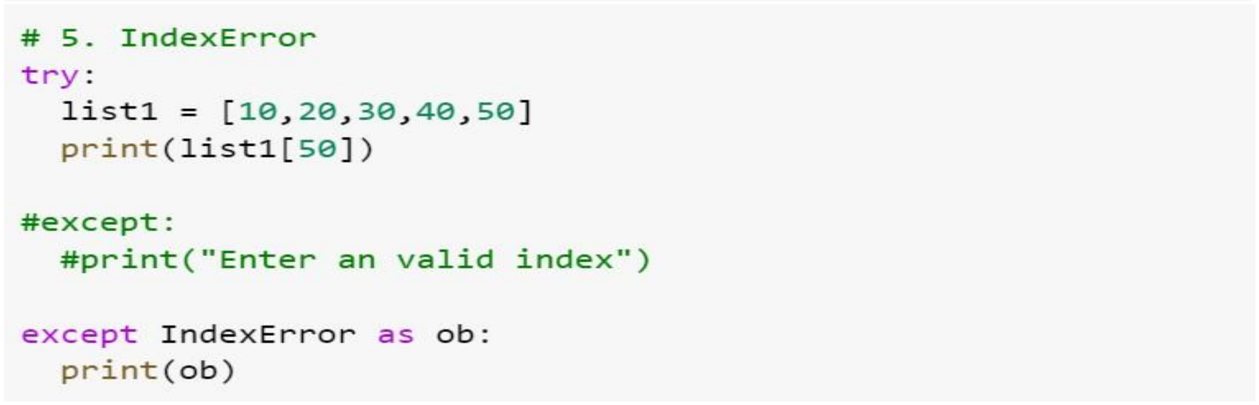
```
# 2. Basic Exception handling
a1 =  int(input("first Number:"))
a2 =  int(input("Second Number:"))
try:
 a = a1/a2
 print(a)                   # If print(a) will generate error that can be handled by except easily.
except:
  print("Divide by 0")
```



```
# 3. ZeroDivisionError
try:
  a1 = int(input("Enter a1:"))
  a2 = int(input("Enter a2:"))
  a = a1 / a2             #if this will rasie ZeroDivisionError it get stored into object ob.

except ZeroDivisionError as ob:
  print(ob)

#except Exception as ob:
  #print(ob)
```



```
# 4. ValueError
try:
  num2 = int("abc1")
  print(num2)

#except ValueError as ob:
  #print(ob)

except:
  print("Enter an valid conversion")
```



```
# 5. IndexError
try:
  list1 = [10,20,30,40,50]
  print(list1[50])

#except:
  #print("Enter an valid index")

except IndexError as ob:
  print(ob)
```

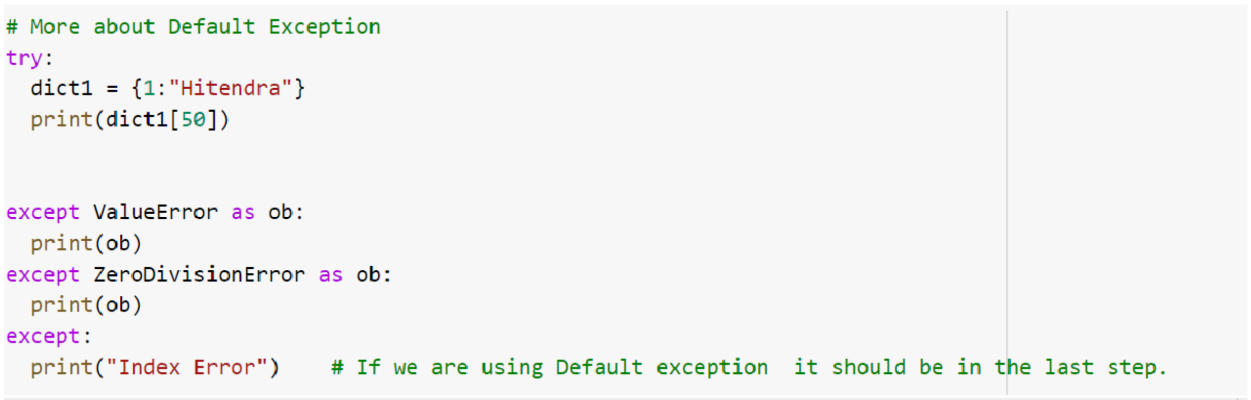
Exception

```python
# 6. IOError
try:
  file1 = open("souce.csv","r")

#except:
  #print("file not found")

#except Exception as ob:
  #print(ob)

#except Exception:
  #print("File not found")

except IOError as ob:
  print(ob)
```

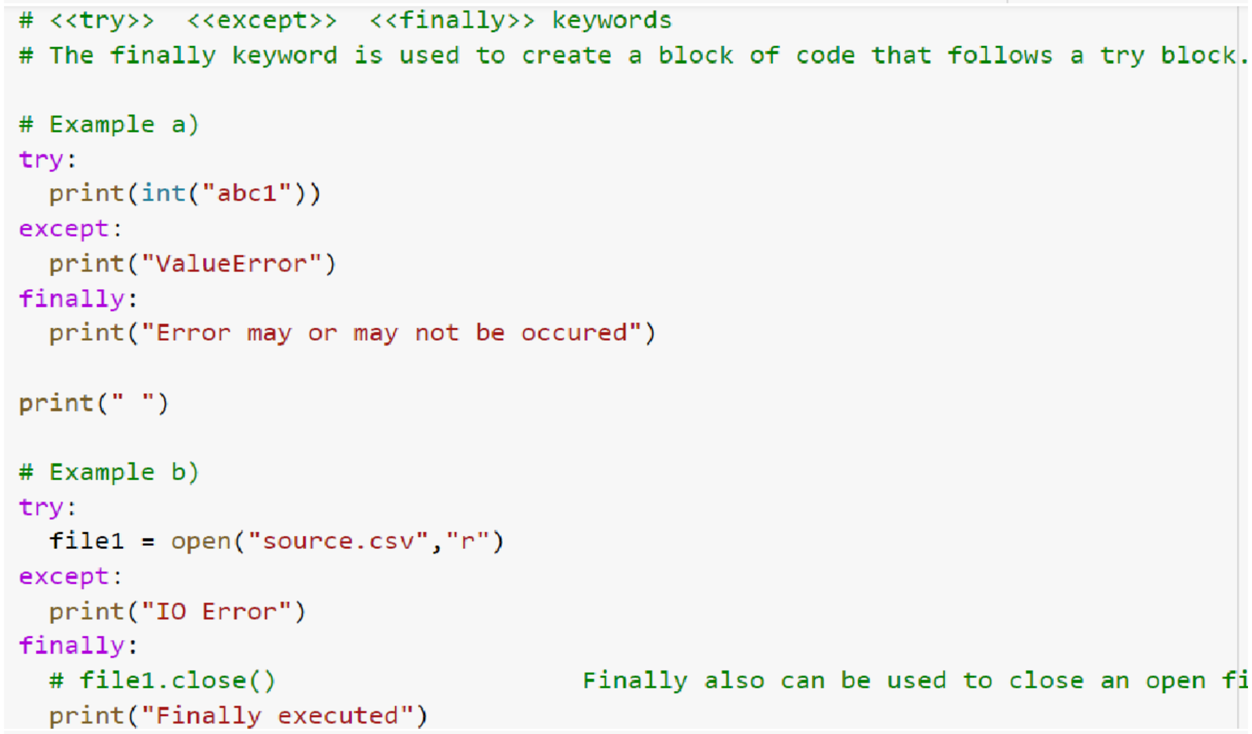
```python
# 7. ImportError
try:
  from math import pll

#except:
  #print("Library doesn't Exist")

#except Exception:
  #print("Library doesn't exist")

#except Exception as ob:
  #print(ob)

except ImportError as ob:
  print(ob)
```

```python
# We also can handel two or more exception at a time
try:
  a1 = int(input())
  a2 = int(input())
  a = a1/a2
  print(a)                       # if print(a) generates an error than call ZeroDivisionError
  print(int("abc1"))             # than print(int("ab1")) doesn't be executed

except (ZeroDivisionError , ValueError) as ob:  # Handling two more Exception under single try:10
  print(ob)
```

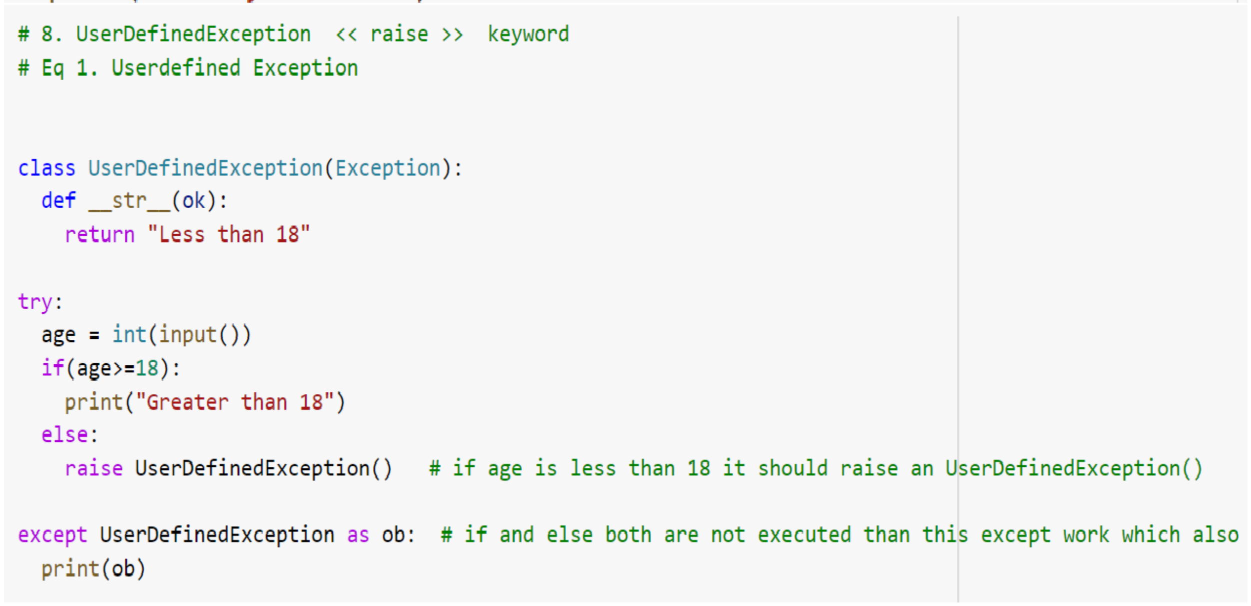
Exception



```
# More about Default Exception
try:
  dict1 = {1:"Hitendra"}
  print(dict1[50])


except ValueError as ob:
  print(ob)
except ZeroDivisionError as ob:
  print(ob)
except:
  print("Index Error")    # If we are using Default exception  it should be in the last step.
```



```
# <<try>>  <<except>>  <<finally>> keywords
# The finally keyword is used to create a block of code that follows a try block.

# Example a)
try:
  print(int("abc1"))
except:
  print("ValueError")
finally:
  print("Error may or may not be occured")

print(" ")

# Example b)
try:
  file1 = open("source.csv","r")
except:
  print("IO Error")
finally:
  # file1.close()                 Finally also can be used to close an open fi
  print("Finally executed")
```



```
# 8. UserDefinedException  << raise >>  keyword
# Eq 1. Userdefined Exception


class UserDefinedException(Exception):
  def __str__(ok):
    return "Less than 18"

try:
  age = int(input())
  if(age>=18):
    print("Greater than 18")
  else:
    raise UserDefinedException()   # if age is less than 18 it should raise an UserDefinedException()

except UserDefinedException as ob:  # if and else both are not executed than this except work which also
  print(ob)
```