Python Exception Handling

Python has many built-in exceptions that are raised when your program encounters an error (something in the program goes wrong).

For example, let us consider a program where we have a function A that calls function B, which in turn calls function C. If an exception occurs in function C but is not handled in C, the exception passes to B and then to A.

If never handled, an error message is displayed and our program comes to a sudden unexpected halt.

Catching Exceptions in Python

In Python, exceptions can be handled using a 'try' statement

The critical operation which can raise an exception is placed inside the 'try' clause. The code that handles the exceptions is written in the 'except' clause.

In [41]:
```
x=0
y=5
print(y/x)
```

```
---------------------------------------------------------------------
----
ZeroDivisionError                         Traceback (most recent call l
ast)
<ipython-input-41-45e77966e675> in <module>
```

```
      1 x=0
      2 y=5
----> 3 print(y/x)

ZeroDivisionError: division by zero
```

Types of Built-in Exceptions :

EOFError

> Raised when the input() function hits end-of-file condition.
>
> ***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.
>
> ***NameError***
>
> Raised when a variable is not found in local or global scope.
>
> ***SyntaxError***
>
> Raised by parser when syntax error is encountered.
>
> ***IndentationError***

Raised when there is incorrect indentation.

TabError

Raised when indentation consists of inconsistent tabs and spaces.

ValueError

Raised when a function gets an argument of correct type but improper value.

ZeroDivisionError

Raised when the second operand of division or modulo operation is zero.

In [65]:
```
if True:
    print('Hi')
```

```
Hi
```

Catching Specific Exceptions in Python

Syntax:

```
try:
    do something
   pass

except ValueError:
    handle ValueError exception
   pass

except (TypeError, ZeroDivisionError):
```

```
                handle multiple exceptions
                TypeError and ZeroDivisionError
                   pass

            except:
                handle all other exceptions
               pass
```

Zero division error

In [66]:
```
x=0
y=5
w='char'
try :
    print(y/x)

except ZeroDivisionError as zde:
    print(zde)
except NameError as ne:
    print("name error",ne)
except TypeError as te:
    print("Error Occured",te)
except ValueError as ve:
    print(ve)
```

```
division by zero
```

Name error

In [67]:
```
x=0
y=5
w='char'
try :
```

```
    print(y/q)

except ZeroDivisionError as zde:
    print("zero division",zde)
except NameError as ne:
    print("Name Error: ",ne)
except TypeError as te:
    print("Error Occured",te)
except ValueError as ve:
    print(ve)
```

```
Name Error:  name 'q' is not defined
```

Type Error

In [70]:
```
x=0
y=5
w='char'
try :
    print(y/w)

except ZeroDivisionError as zde:
    print("zero division",zde)
except NameError as ne:
    print("Name Error: ",ne)
except TypeError as te:
    print("Type Error: ",te)
except ValueError as ve:
    print('Value Error: ',ve)

print('Hello')
```

```
Type Error:  unsupported operand type(s) for /: 'int' and 'str'
Hello
```

Value Error

In [71]:
```
print(5/0)
print('hello')
```

```
---------------------------------------------------------------------
----
ZeroDivisionError                         Traceback (most recent call l
ast)
<ipython-input-71-8341eb64b54d> in <module>
----> 1 print(5/0)
      2 print('hello')

ZeroDivisionError: division by zero
```

In [72]:
```
w='char'
try :
    x = int(w)

except ZeroDivisionError as zde:
    print("zero division",zde)
except NameError as ne:
    print("Name Error: ",ne)
except TypeError as te:
    print("Type Error:",te)
except ValueError as ve:
    print('Value Error: ',ve)
```

```
Value Error:  invalid literal for int() with base 10: 'char'
```

In [ ]:
```

```

Raise Error:

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

In [76]:
```
a=-1
try :
    if a<=0:
        raise Exception("a={}".format(a),"is must be greater than 1")
except Exception as ve:
    print(ve)
```

```
('a=-1', 'is must be greater than 1')
```

Python try with else clause

In some situations, you might want to run a certain block of code if, the code block inside try ran without any errors. For these cases, you can use the optional else keyword with the try statement.

If try block doesn't contain any exception then the else block will be executed.

Else is implied on except block... i.e., if except breaks only then else run.

In [78]:
```
try:
    num = int(input("Enter a number: "))
    assert num % 2 == 0
except:
    print("Not an even number!")
else:
    reciprocal = 1/num
    print(reciprocal)
```

```
Enter a number: 4
0.25
```

Python try...finally

The try statement in Python can have an optional finally clause. This clause is executed no

matter what, and is generally used to release external resources.

In [80]:
```
try:
    num = int(input("Enter a number: "))
    assert num % 2 == 0
except:
    print('In the except block ,',end = ' ')
    print("Not an even number!")
else:
    reciprocal = 1/num
    print(reciprocal)
finally:
    print("end of try block finally!!")
```

```
Enter a number: 4
0.25
end of try block finally!!
```

In [ ]:
```
```

In [43]:
```
s = 'siddhesh'
s[15]
```

```
---------------------------------------------------------------------
----
IndexError                                Traceback (most recent call l
ast)
<ipython-input-43-ee80ddae77d7> in <module>
      1 s = 'siddhesh'
----> 2 s[15]

IndexError: string index out of range
```

In [44]:
```
d = dict()
d['sid']
```

```
---------------------------------------------------------------------
----
```

```
KeyError                                  Traceback (most recent call l
ast)
<ipython-input-44-6a53e849fe9f> in <module>
      1 d = dict()
----> 2 d['sid']

KeyError: 'sid'
```

In [45]:
```
d
```

Out[45]:
```
{}
```

In [47]:
```
if True:
    pass
print('hi')
```

```
hi
```

In [ ]:
```

```