Excepciones

Excepciones

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ZeroDivisionError: integer division or modulo by zero

>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'spam' is not defined


>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: cannot concatenate 'str' and 'int' objects
```

Excepciones

```
>>> def division(a, b):
...     return a / b
...

>>> division(2,2)
1
>>> division(2,0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in division
ZeroDivisionError: integer division or modulo by zero
```

Excepciones - Tratamiento Básico

```
>>> def division(a, b):
...     try:
...       resultado = a / b
...     except:
...       resultado = None
...     return resultado
...
>>> division(1,0)
>>>
```

Excepciones - Tratamiento Básico

```
>>> def division(a, b):
...     try:
...         resultado = a / b
...     except ZeroDivisionError:
...         resultado = 0
...     except TypeError:
...         resultado = int(a) / int(b)
...     return resultado
...
>>> division(1,0)
0
```

Excepciones - Más de un error

```
import sys

try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except OSError as err:
    print("OS error: {0}".format(err))
except ValueError:
    print("Could not convert data to an integer.")
except:
    print("Unexpected error:", sys.exc_info()[0])
    raise
```

Excepciones - Multiple Catch

```
import sys

try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except (OSError, ValueError) as err:
    err_type = type(err)
    print("Error de tipo".format(err_type))
except:
    print("Unexpected error:", sys.exc_info()[0])
    raise
```

Excepciones - Cleanup

```
import sys


try:
    f = open('myfile.txt')
except OSError as err:
    print("OS error: {0}".format(err))
except:
    print("Unexpected error:", sys.exc_info()[0])
    raise
else:
    s = f.readline()
finally:
    if not f.closed:
        f.close()
```

Excepciones - Cleanup

```
import sys


try:
    f = open('myfile.txt')
except OSError as err:
    print("OS error: {0}".format(err))
except:
    print("Unexpected error:", sys.exc_info()[0])
    raise
else:
    s = f.readline()
finally:
    if not f.closed:
        f.close()
```

Excepciones - Cleanup (SIEMPRE)

```
>>> def division(a, b):
... try:
...     return a / b
... except ZeroDivisionError:
...     return 0
... finally:
...     print("esto se ejecuta siempre")
...
>>> division(1, 0)
esto se ejecuta siempre
0
```

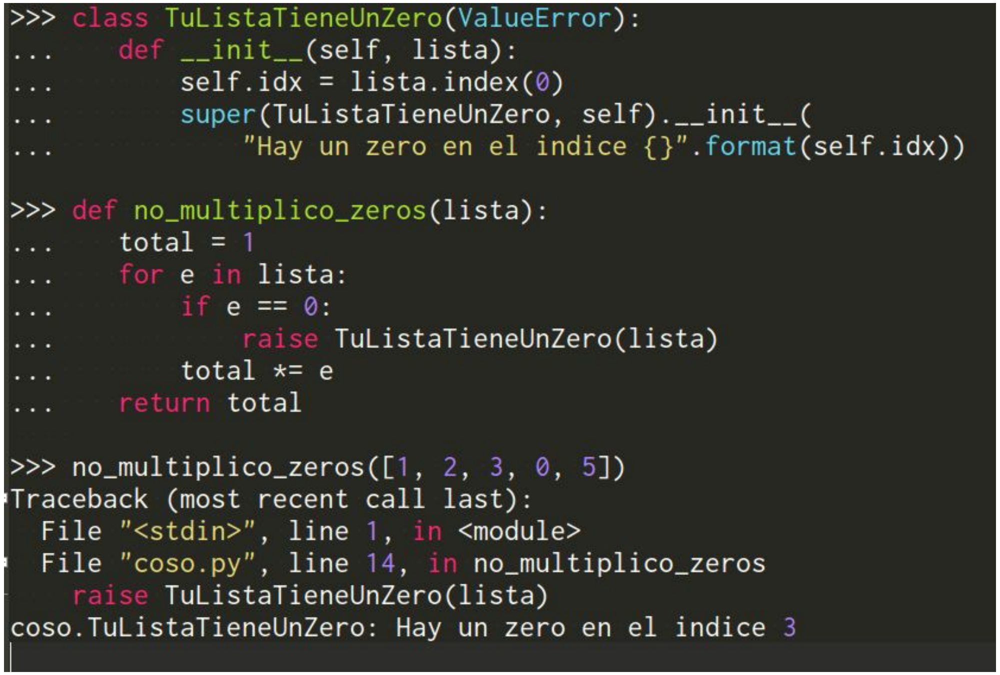
Excepciones - Lanzando Exceptions

```
>>> def no_multiplico_zeros(lista):
...     total = 1
...     for e in lista:
...         if e == 0:
...             raise ValueError("Dije que no multiplico zeros!")
...         total *= e
...     return total

>>> no_multiplico_zeros([1, 2, 3, 4, 5])
120

>>> no_multiplico_zeros([1, 2, 3, 0, 5])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "coso.py", line 5, in no_multiplico_zeros
    raise ValueError("Dije que no multiplico zeros!")
ValueError: Dije que no multiplico zeros!
```

Excepciones - Personalizando

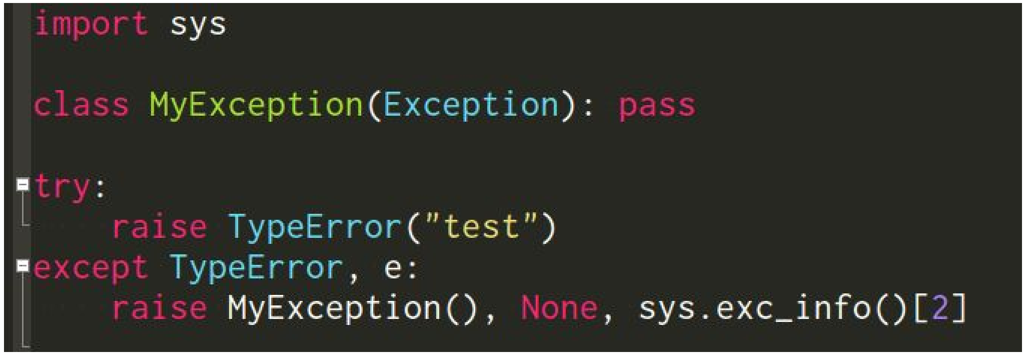
```
class MiError(Exception):
    pass

try:
    MiError("CLANK!")
except MiError:
    print("exploto todo macho!")
```

Excepciones - Personalizando



```
>>> class TuListaTieneUnZero(ValueError):
...     def __init__(self, lista):
...         self.idx = lista.index(0)
...         super(TuListaTieneUnZero, self).__init__(
...             "Hay un zero en el indice {}".format(self.idx))

>>> def no_multiplico_zeros(lista):
...     total = 1
...     for e in lista:
...         if e == 0:
...             raise TuListaTieneUnZero(lista)
...         total *= e
...     return total

>>> no_multiplico_zeros([1, 2, 3, 0, 5])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "coso.py", line 14, in no_multiplico_zeros
    raise TuListaTieneUnZero(lista)
coso.TuListaTieneUnZero: Hay un zero en el indice 3
```

Excepciones - Traceback



```
import sys

class MyException(Exception): pass

try:
    raise TypeError("test")
except TypeError, e:
    raise MyException(), None, sys.exc_info()[2]
```

- **Discusion Completa (Py3 details:** http://stackoverflow.com/questions/1350671/inner-exception-with-traceback-in-python
- **Modulo traceback:** https://docs.python.org/2/library/traceback.html

Archivos Rapido

Archivos - Contextos

```
with open(“archivo.txt”) as fp:
        fp.read()
        fp.seek(0)
        fp.tell()
```