# Exception Handling

Examples of when things go wrong:

- Scraping a web page when the server is down
- Trying to add an integer and a `None` type object
- Opening a file that doesn't exist
- A database connection dies
- User interrupts program with `Ctrl + C`

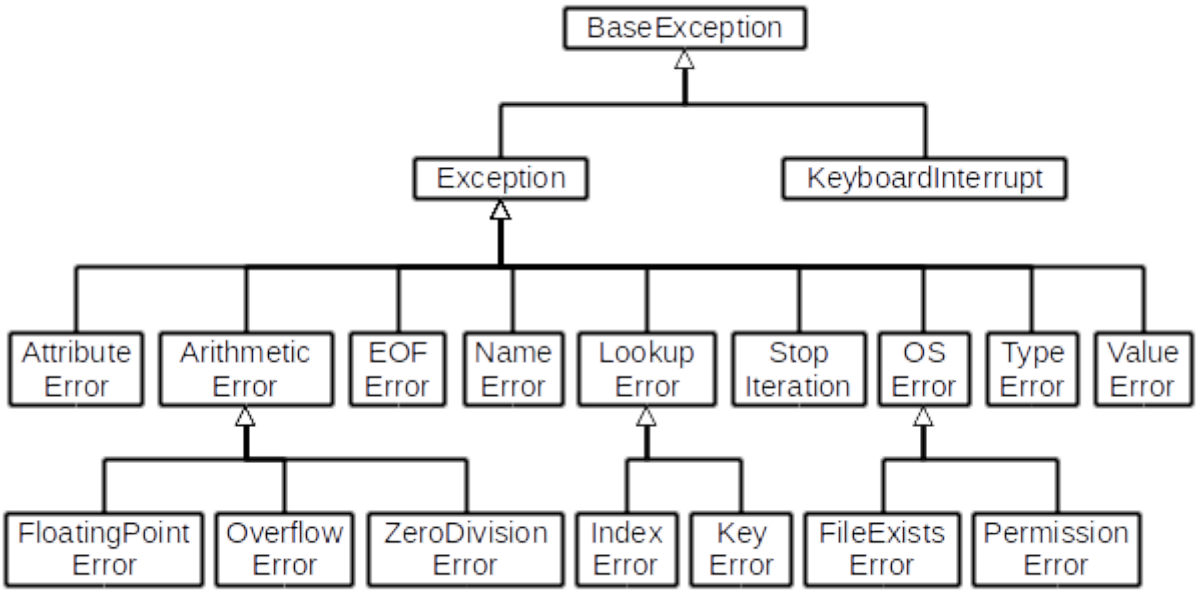
What should Python do in these circumstances?

- Break! Sometimes it's gone so wrong, it shouldn't continue
- Keep going

Execption handling helps you "keep going"

## Exceptions in Python

What could go wrong?




## Handling an Exception

In [ ]:
```
try:
    # this always gets entered
    something_that_might_go_wrong()

except SomeException as e:
    # only triggered with SomeException
    print "Encountered an Exception:", e

except (DifferentException, YetAnotherException) as de:
    # catch for two different exceptions that should be handled the same
    print "Encountered:", de

else:
    # optional block
    print "Nothing went wrong, we're all good!"

finally:
    # optional clean-up block
    something_we_would_do_either_way()
```

## Example: Divide by Zero

In [ ]:
```
divisor = 0

try:
    answer = 5 / divisor
except ZeroDivisionError as zde:
    print "Cannot divide by zero! ->", zde
else:
    print "We made it! Answer =", answer
finally:
    print "This prints no matter what"
```

## Raising Exceptions

In [ ]:
```
def calculate_months_alive(years_old):
    if years_old < 0:
        raise Exception("Cannot have negative age!")
    return years_old * 12

# calculate_months_alive(-10)
```

## Shortcut for Raising Input Validation Exceptions (`assert`)

Format:

```
assert <condition>, <message if condition is False>
```

Will raise an `AssertionError`.

In [ ]:
```
def calculate_months_alive(years_old):
    assert years_old > 0, "Cannot have negative age!"
    return years_old * 12

# calculate_months_alive(-10)
```

## Custom Exceptions

In [ ]:
```
class NegativeAgeError(Exception):
    pass

def calculate_months_alive(years_old):
    if years_old < 0:
        raise NegativeAgeError("Cannot have negative age!")
    return years_old * 12

# now as if in a script
try:
    months = calculate_months_alive(-10)
except NegativeAgeError:
    months = 0
    print "NegativeAgeError: Negative age, setting months to `0`!"
finally:
    print "Person is", months, "months old"
```

## Debugging Exceptions

- `traceback` module
- `pdb` module

In [1]:
```
import traceback

numerator = 100
denominators = [1, 8, 0, 3, 2, 12]

for denom in denominators:
    try:
        answer = numerator / denom
    except ZeroDivisionError as zde:
        full_stack_trace = traceback.format_exc()  # <-- get the full tr
ace!
        print full_stack_trace
        import pdb; pdb.set_trace()                # <-- drop into a deb
ugging shell
    else:
        print "We made it! Answer =", answer
```

```
We made it! Answer = 100
We made it! Answer = 12
Traceback (most recent call last):
  File "<ipython-input-1-03fb296e20ad>", line 8, in <module>
    answer = numerator / denom
ZeroDivisionError: integer division or modulo by zero

> <ipython-input-1-03fb296e20ad>(6)<module>()
-> for denom in denominators:
(Pdb) numerator
100
(Pdb) denom
0
(Pdb) numerator / denom
*** ZeroDivisionError: integer division or modulo by zero
(Pdb) p "Ah I see"
'Ah I see'
(Pdb) c
We made it! Answer = 33
We made it! Answer = 50
We made it! Answer = 8
```

## Lab: Coding Excercises

Fill in the method definitions in the file `excercises/exceptions.py`.

Make sure you can pass tests with:

```
$ py.test tests/test_exceptions.py::ExceptionExcercises::<function_name>  #
 test single function
$ py.test tests/test_exceptions.py::ExceptionExcercises                   #
 test all at once
```