Python Exceptions

Stéphane Vialette

LIGM, Université Paris-Est Marne-la-Vallée

October 21, 2009

Outline

Looking at some exceptions

Exceptions

```
In [1]: f
---------------------------
NameError Traceback (most recent call last)
.../<ipython console> in <module>()
NameError: name 'f' is not defined

In [2]: 1/0
---------------------------
ZeroDivisionError  Traceback (most recent call last)
.../<ipython console> in <module>()
ZeroDivisionError: integer division or modulo by zero

In [3]: for
---------------------------
   File "<ipython console>", line 1
     for
         ^
SyntaxError: invalid syntax
In [4]:
```

Looking at some exceptions

Exceptions

```
In [1]: l = []
In [2]: l[0]
---------------------------------------------------------------------------
IndexError Traceback (most recent call last)
.../<ipython console> in <module>()
IndexError: list index out of range

In [3]: d = {}
In [4]: d['some_key']
---------------------------------------------------------------------------
KeyError Traceback (most recent call last)
.../<ipython console> in <module>()
KeyError: 'some_key'

In [5]: f = open('not_a_file')
---------------------------------------------------------------------------
IOError Traceback (most recent call last)
.../<ipython console> in <module>()
IOError: [Errno 2] No such file or directory: 'not_a_file'
```

Looking at some exceptions

Exceptions

```
In [1]: class A(object):
   ...:     pass
   ...:
In [2]: a = A()
In [3]: a.f
---------------------------------------------------------------------------
AttributeError Traceback (most recent call last)
.../<ipython console> in <module>()
AttributeError: 'A' object has no attribute 'f'

In [4]: a.f()
---------------------------------------------------------------------------
AttributeError Traceback (most recent call last)
.../<ipython console> in <module>()
AttributeError: 'A' object has no attribute 'f'

In [5]:
```

Outline

try-except statement

Syntax

```
try:
    try_suite # watch for exception here
except Exception[, reason]:
    except_suite # exception-handling code
```

Example

```
In [1]: try:
   ...:     1/0
   ...: except ZeroDivisionError, e:
   ...:     print 'error:', e
   ...:
error: integer division or modulo by zero

In [2]:
```

try-except statement

Example

```
In [1]: try:
   ...:     1/0
   ...: except AttributeError, e:
   ...:     print 'error:', e
   ...:
---------------------------------------------------------------------------
ZeroDivisionError Traceback (most recent call last)
.../<ipython console> in <module>()
ZeroDivisionError: integer division or modulo by zero

In [2]:
```

try-except statement

safe_float

```
In [1]: print float(123)
123.0
In [2]: print float('axy')
---------------------------------------------------------------------------
ValueError Traceback (most recent call last)
.../<ipython console> in <module>()
ValueError: invalid literal for float(): axy
In [3]: def safe_float(obj):
   ...:     try:
   ...:         return float(obj)
   ...:     except ValueError:
   ...:         pass
In [4]: print safe_float(123)
123.0
In [5]: print safe_float('xyz')
None
```

try-except statement

safe_float

```
In [1]: def safe_float(obj):
   ...:     try:
   ...:         retval = float(obj)
   ...:     except ValueError:
   ...:         retval = None
   ...:     return retval
In [2]:
```

safe_float

```
In [1]: def safe_float(obj):
   ...:     try:
   ...:         retval = float(obj)
   ...:     except ValueError:
   ...:         retval = 'could not convert non-number to float'
   ...:     return retval
In [2]:
```

try statement with multiple excepts

safe_float

```
In [1]: def safe_float(obj):
   ...:     try:
   ...:         retval = float(obj)
   ...:     except ValueError:
   ...:         retval = 'could not convert non-number to float'
   ...:     except TypeError:
   ...:         retval = 'object type cannot be converted to float'
   ...:     return retval


In [2]: print safe_float('ayz')
could not convert non-number to float

In [3]: print safe_float(())
object type cannot be converted to float

In [4]:
```

except statement with multiple Exceptions

safe_float

```
In [1]: def safe_float(obj):
   ...:     try:
   ...:         retval = float(obj)
   ...:     except (ValueError, TypeError):
   ...:         retval = 'obj must be a number or numeric string'
   ...:     return retval
   ...:

In [2]: print safe_float('ayz')
obj must be a number or numeric string

In [3]: print safe_float(())
obj must be a number or numeric string

In [4]:
```

Catching all exceptions

safe_float

```
In [1]: def safe_float(obj):
   ...:     try:
   ...:         retval = float(obj)
   ...:     except Exception:
   ...:         retval = 'obj must be a number or numeric string'
   ...:     return retval
   ...:

In [2]: print safe_float('ayz')
obj must be a number or numeric string

In [3]: print safe_float(())
obj must be a number or numeric string

In [4]:
```

Catching all exceptions

Notes

There are exceptions that are not due to an error condition.

- `SystemExit`, and
- `KeyboardInterrupt`.

Catching all exceptions

```
try:
    try_suite
except (KeyboardInterrupt, SystemExit):
    raise # user wants to quit
except Exception:
    handle_real_errors
```

else clause

Example

```
import some_module

log = open('logfile', 'w')

try:
    some_module.some_function()
except:
    log.write("caught exception")
else:
    log.write("no exception caught")

log.close()
```

`finally` clause

Description

A `finally` clause is one where its suite or block is executed regardless of whether an exception occurred or whether it was caught (or not).

(Full) Syntax

```
# no exeption: A - C - D
# exception: A - B - D
try:
    A
except:
    B
else:
    C
finally:
    D
```

Outline

A RGB color class

Example

```
class RGBColor(object):
    def __init__(self, red, green, blue):
        self.red = red
        self.green = green
        self.blue = blue

if __name__ == '__main__':
    c = RGBColor(0, 0, 256) # oups ...
```

A RGB color class

Example

```
class RGBColorError(ValueError):
    pass
class RGBColor(object):
    def __init__(self, red, green, blue):
        RGBColor.check_color_values((red, green, blue))
        (self.red, self.green, self.blue) = (red, green, blue)

    @staticmethod
    def check_color_values(color_values):
        for color_value in color_values:
            if not 0 <= color_value <= 255:
                raise RGBColorError

if __name__ == '__main__':
    c = RGBColor(0, 0, 256) # raise RGBColorError
```