Python 101

Sentencia with, expresiones regulares, manipulación de errores

Manipulación de errores

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

Manipulación de errores

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

Manipulación de errores

```
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: Can't convert 'int' object to str implicitly
```

Manipulación de errores

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

Manipulación de errores - Orden de ejecución

La cláusula “try” es ejecutada, junto con su contenido.
Si lo que está dentro de la sentencia no genera excepción, entonces la cláusula “except” nunca se ejecuta y el “try” finaliza.
Si ocurre algún tipo de excepción durante la ejecución del “try”, todo lo que no llegó a ejecutarse dentro del try se omite y el programa prosigue por la cláusula “except” siempre y cuando el tipo (type) de la excepción sea el mismo. Luego ejecuta su contenido y prosigue por la siguiente por fuera del “scope”.
Si la excepción no es manejada de manera adecuada (por ejemplo, no encuentra un tipo que se corresponda con el error) el programa se interrumpe imprimiendo el mensaje.

Manipulación de errores

Una sentencia “try” puede tener más de un tipo de sentencia “except” asociada. Se pude especificar de manera explícita diferentes tipos de excepción.
Como mucho uno solo será ejecutado, que será la que se corresponda al error generado en el “try”. Sin embargo, una sentencia “except” puede tener asociada múltiples tipos de excepción.

Manipulación de errores

```
... Try:
        ....
... except (TypeError, NameError):
...     pass
... except (RuntimeError):
...     pass
```

Manipulación de errores

```
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
```

Manipulación de errores - else statement

The **try** ... **except** statement has an optional **else** clause, which, when present, must follow all **except** clauses. It **is** useful **for** code that must be executed **if** the **try** clause does **not raise** an exception. **For** example:

```
for arg in sys.argv[1:]:
    try:
        f = open(arg, 'r')
    except IOError:
        print('cannot open', arg)
    else:
        print(arg, 'has', len(f.readlines()), 'lines')
        f.close()
```

Manipulación de errores - finally statement

```
>>> try:
...     raise KeyboardInterrupt
... finally:
...     print('Goodbye, world!')
...
Goodbye, world!
KeyboardInterrupt
Traceback (most recent call last):
  File "<stdin>", line 2, in ?
```

https://docs.python.org/2/library/functions.html#open

Manipulación de errores

```
class Error(Exception):
    """Base class for exceptions in this module."""
    pass

class InputError(Error):
    """Exception raised for errors in the input.

    Attributes:
        expression -- input expression in which the error occurred
        message -- explanation of the error
    """

    def __init__(self, expression, message):
        self.expression = expression
        self.message = message
```

Predefined Clean-up Actions

https://docs.python.org/3/tutorial/errors.html

Sentencias compuestos

https://docs.python.org/3/reference/compound_stmts.html#with

https://docs.python.org/3/whatsnew/2.6.html#pep-343-the-with-statement

http://effbot.org/zone/python-with-statement.htm

Expresiones regulares

Vamos a la consola!