Python Class 3: Errors, Exceptions and Testing

Betül Demirkaya

August 6, 2015

Errors

Exceptions

Testing

Break, Continue and Else

Types of Errors

- Syntax error
  - Errors related to language structure.
- Runtime error
  - Errors during the execution of program.
  - eg. TypeError, NameError
- Semantic error
  - The program will run successfully but the output is not what you expect.
  - You'll need to run a test.

Debugging Tips

Make sure:

- You are not using a reserved/keyword.
  List of keywords:
  https://docs.python.org/2.5/ref/keywords.html
- You have `:` after `for`, `while`, etc.
- Parentheses and quotations are closed properly.
- You use $=$ and $==$ correctly.
- Indentation is correct.

Exceptions

- `raise:  #to create exceptions or errors`
- `pass:  #to continue execution without doing anything`
- `try:  #tries executing the following`

```
  ....
except TypeError:
  ... # runs if a Type Error was raised
except:
  ... # runs for other errors or exceptions
else:
  ... # runs if there was no exception/error
finally:
  ... # always runs!
```

- You can create your own exceptions using classes.

Sample Test

```
import unittest #You need this module
import myscript #This is the script you want to test


class mytest(unittest.TestCase):

  def test_one(self):
    self.assertEqual("result I need", myscript.myfunction(myinput))

  def test_two(self)
    thing1=myscript.myfunction(myinput1)
    thing2=myscript.myfunction(myinput2)
    self.assertNotEqual(thing1, thing2)

if __name__ == '__main__': #Add this if you want to run the test with this script.
  unittest.main()
```

Break, Continue and Else

- These statements can be handy using `while` or `for` loops.
- `break #stops the loop`
- `continue # moves on to the next iteration`
- `else #executed only if all iterations are completed`