The Python Language (Part 3)

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - A subset of Python...
  - That is appropriate for COS 333...
  - Through example programs

Agenda

- **Modules**
- Packages
- Object-oriented programming

Modules

- ***Module***
  - A .py file that is designed to be included into a ***client*** .py file

Modules

- See **euclid.py**, **euclidclient3.py**

```
$ python euclidclient3.py
Enter the first integer:  8
Enter the second integer:  12
gcd: 4
lcm: 24
$ python euclidclient3.py
Enter the first integer:  8
Enter the second integer:  0
gcd: 8
lcm(i,j) is undefined if i or j is 0
$ python euclidclient3.py
Enter the first integer:  0
Enter the second integer:  0
gcd(i,j) is undefined if i and j are 0
$
```

Modules

- Building and running

```
$ python euclidclient3.py
```

  - Automatically compiles/interprets euclid.py

Agenda

- Modules
- **Packages**
- Object-oriented programming

Packages

- ***Package***
  - A named group of modules (and other packages)
    - A *module* is stored in a *file*
    - A *package* is stored in a *directory*

Packages

- See **intmath/__init__.py**
  - Declares `intmath` as a package
- See **intmath/euclid.py**
  - A module in the `intmath` package
- See **intmath/fibonacci.py**
  - A module in the `intmath` package

Packages

- See **euclidclient4.py**

```
$ python euclidclient4.py
Enter the first integer:  8
Enter the second integer:  12
gcd: 4
lcm: 24
$ python euclidclient4.py
Enter the first integer:  8
Enter the second integer:  0
gcd: 8
lcm(i,j) is undefined if i or j is 0
$ python euclidclient4.py
Enter the first integer:  0
Enter the second integer:  0
gcd(i,j) is undefined if i and j are 0
$
```

Packages

- See **fibclient.py**

```
$ python fibclient.py
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
fib(6) = 8
fib(7) = 13
fib(8) = 21
fib(9) = 34
$
```

Agenda

- Modules
- Packages
- **Object-oriented programming**

Object-Oriented Programming

- See **fractionprelim.py**, **fractionprelimclient.py**

```
$ python fractionprelimclient.py
Numerator 1:  1
Denominator 1:  2
Numerator 2:  3
Denominator 2:  4
frac1: 1/2
frac2: 3/4
frac1 hashcode: -3550055125485641917
frac1 does not equal frac2
frac1 is less than frac2
frac1 is less than or equal to frac2
-frac1: -1/2
frac1 + frac2: 5/4
frac1 - frac2: -1/4
frac1 * frac2: 3/8
frac1 / frac2: 2/3
$
```

Object-Oriented Programming

- What is the effect of this code?

```
f = fraction.Fraction(3, 4)
f.num = 6
```

Aside: Name Mangling

- Incidentally:
  - Use of leading **double** underscores causes ***name mangling***
    - Example: In Fraction, compiler turns `__num` into `_Fraction__num`

Summary

- We have covered these aspects of Python:
  - Modules
  - Packages
  - Object-oriented programming
- See also:
  - **Appendix**: Duck Typing

Appendix: Duck Typing

Duck Typing

- See **euclidstrong.py**
  - Which is better, euclid.py or euclidstrong.py?

Duck Typing

- Observation:
  - Python uses ***duck typing***

"When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."

-- James Whitcomb Riley

Duck Typing

- **Style 1**: Don't validate parameter types
  - Validating parameter types is constraining and slow
  - **So euclid.py is better**
- **Style 2**: Validate parameter types
  - Validating parameter types is safe
  - **So euclidstrong.py is better**
- We'll use Style 1

Duck Typing

- Commentary
  - **Small** projects:
    - Maybe need not validate parameter types
  - **Large** projects:
    - Maybe should validate parameter types

Duck Typing

- Commentary
  - But if you feel the need to validate parameter types, then why are you using Python???

Duck Typing

Language	Object references have types?	Objects have types?	Language classification
C	yes	no	**weakly** typed
Java	yes	yes	**strongly** typed
Python	no	yes	**dynamically** typed