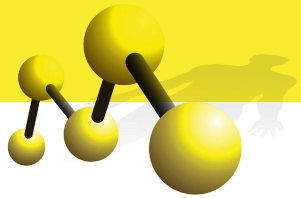
Defining Your Own Types

Luis Pedro Coelho

Programming for Scientists

October 22, 2012

Modules

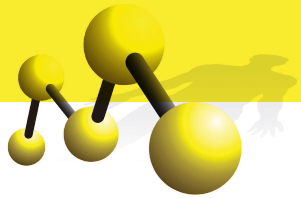


We have already seen this

```
import module
```

what is happening exactly?

Modules



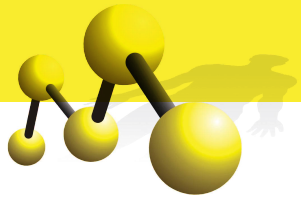
module.py

```
def hello():
    print 'Hello'
```

main.py

```
import module
module.hello()
```

Modules



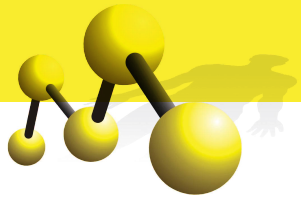
module.py

```
def hello():
    print 'Hello'
```

main.py

```
import module as mod
mod.hello()
```

Modules



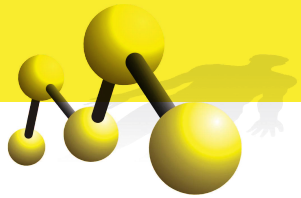
module.py

```
def hello():
    print 'Hello'
```

main.py

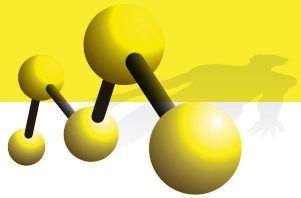
```
from module import hello
hello()
```

Standard Library



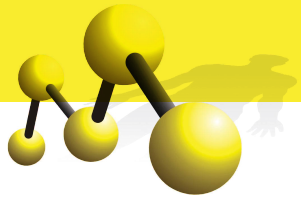
```
import datetime
print datetime.datetime.now()
```

Non-Standard Library



```
import numpy
```

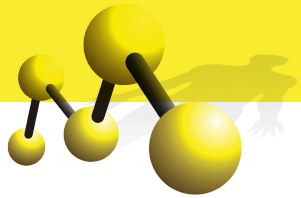
User-Defined Types



Built-in Types

1. lists
2. dictionaries
3. strings
4. ...

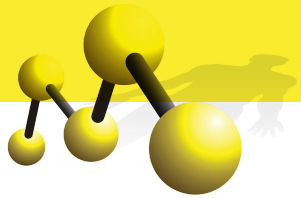
Type



What's a Type

1. A domain of values
2. A set of methods (functions)

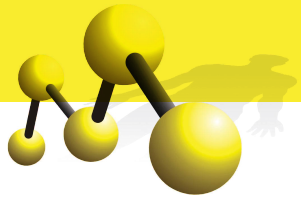
Examples of Types



List

1. Domain: lists
2. Functions: L.append(e),L.insert(idx,e), ...
3. Operators: L[0], 'Rita' in L

Examples of Types



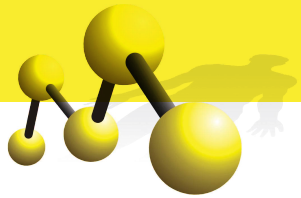
List

1. Domain: lists
2. Functions: L.append(e),L.insert(idx,e), ...
3. Operators: L[0], 'Rita' in L

Integer

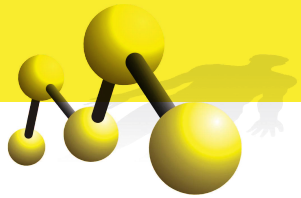
1. Domain: $\ldots, -2, 1, 0, 1, 2, \ldots$
2. Operators: A + B,...

User-defined Types



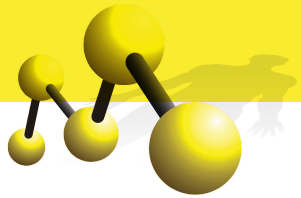
Object-oriented programming languages allow us to define new types.

FastQ Example



- DNA (RNA) sequence
- Quality (integer value) for each position

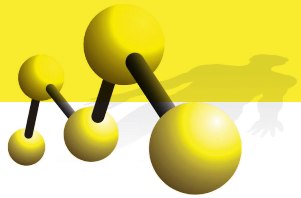
FastQ Example



```
def mean(xs):
    return sum(xs)/float(len(xs))
class FastQSequence(object):
    def __init__(self, seq, quals):
        if len(seq) != len(quals):
            print 'OOOOOOOOOPS!'
        self.seq = seq
        self.quals = quals

    def averageq(self):
        return mean(self.quals)
```

FastQ Example

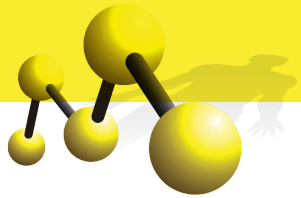


```
class NAME(object):
    def __init__(self, ...):
        ...

    def METHODNAME(self, . . .):
        . . .
```

Note: it is a double underscore!

FastQ Example

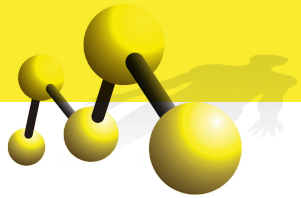


```
def mean(xs):
    return sum(xs)/float(len(xs))
class FastQSequence(object):
    def __init__(self, seq, quals):
        if len(seq) != len(quals):
            print 'OOOOOOOOOOPS!'
        self.seq = seq
        self.quals = quals

    def averageq(self):
        return mean(self.quals)

s = FastQSequence('ATTA', [23, 32, 20, 21])
print s.averageq()
```

Exercise



Take the previous class and a method called minq which returns the minimum quality.