Why learn Python?

Reason 1: Vocational Skill

Job listings at monster.com (20 October 2009)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>All US</th>
<th>Virginia Only</th>
<th>Typical Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scheme</td>
<td>83</td>
<td>0</td>
<td>$100-999K</td>
</tr>
</tbody>
</table>

"Scheme" Jobs
Reason 2: Expanding Minds

Languages change the way we think.

The more languages you know, the more different ways you have of expressing ideas and for thinking about (and solving) problems.

“Jamais Jamais Jamais” from Harmonice Musices Odhecaton A. Printed by Ottaviano Dei Petrucci in 1501 (first music with movable type).

Reason 3: Deepening Understanding

By seeing how the same concepts we encountered in Scheme are implemented by a different language, you will understand those concepts better (especially procedures, assignment, data abstraction).

“Jamais Jamais Jamais” from Harmonice Musices Odhecaton A. (1501)

Reason 4: Building Confidence

By learning Python (mostly) on your own, the next time you encounter a problem that is best solved using a language you don’t know, you will be confident you can learn it (rather than trying to use the wrong tool to solve the problem).

Reason 5: Fun

Programming in Python is fun (possibly even more fun than programming in Scheme!)

Especially because:
- It is an elegant and simple language
- Most programs mean what you think they mean
- It is dynamic and interactive
- It can be used to easily build web applications
- It is named after Monty Python’s Flying Circus
- It was designed by someone named Guido.

This is also important for taking cs2110 this Spring: assumes you can learn Java on your own.
Python

A universal programming language
— Everything you can compute in Scheme you can compute in Python, and vice versa
— Chapter 11/PS7: implement a Scheme interpreter in Python
— Chapter 12: more formal definition of a universal PL

Imperative Language
— Designed to support a programming where most of the work is done using assignment statements: \( x = e \)

Object-Oriented Language
— All data are objects
— Built in support for classes, methods, inheritance

Learning New Languages

Syntax: Where the {,%;!,$, etc. all go
If you can understand a BNF grammar, this is easy
(Okay, it still takes some getting used to a new syntax…)

Semantics: What does it mean
Learning the evaluation rules
Harder, but most programming languages have very similar evaluation rules (but the subtle differences can cause lots of problems)

Style: What are the idioms and customs of experienced programmers in that language?
Takes many years to learn - need it to be a "professional" Python programmer, but not to make a useful program

Python If

**Instruction** ::= **if** \((Expression)\) : **Block\_Consequent**
**else**: **Block\_Alternate**

Evaluate \(Expression\). If it evaluates to a true value, evaluate the **Block\_Consequent**; otherwise, evaluate the **Block\_Alternate**.

Similar to \((if\ (Expression)\ (begin\ \text{Block\_Consequent})\ (begin\ \text{Block\_Alternate}))\)

Differences:
Indenting and new lines matter!
Changing the indentation changes meaning of code
What is a "true value":
**Scheme**: anything that is not false.
**Python**: anything that is not False, None, 0, an empty string or container

If Example

```python
if []:
    print "Empty is true!"
else:
    print "Empty is false!"
```

Empty is false!

Learning Python

• We will introduce (usually informally) Python constructs in class as we use them, example code in PS6
• The “Schemer’s Guide to Python” is an introduction to Python: covers the most important constructs you need for PS6, etc.
• Course book: Chapter 11 introduces Python
  — Read ahead Section 11.1
• On-line Python documentation

Making Objects

```python
\textbf{class} Dog:
\textbf{def} bark(self):
    print “wuff wuff wuff wuff”
```

In Washington, it’s dog eat dog. In academia, it’s exactly the opposite.
Robert Reich
Making a Dog

```python
class Dog:
    def bark(self):
        print "wuff wuff wuff wuff"

spot = Dog()
```

Python assignments are like both define and set! If the Variable name is not yet defined, it creates a new place. The value in the named place is initialized to the value of the Expression.

Some Python Procedures

```python
def square(x):
    return x * x
def bigger(a,b):
    if a > b:
        return a
    else:
        return b
```

Def bigger(a,b):
If a > b:
    return a
else:
    return b

Whitespace Matters!

```python
File "<pyshell#1>", line 4
else:
   return b

^  File "<pyshell#1>", line 4
IndentationError: unindent does not match any outer indentation level

Python requires you to format your code structurally!
```

Barking: Invoking Methods

```python
class Dog:
    def bark(self):
        print "wuff wuff wuff wuff"

spot = Dog()
spot.bark("Hello")
wuff wuff wuff wuff
```

Object Lingo

"Apply a procedure" :: "Invoke a method"

We apply a procedure to parameters.

We invoke a method on an object, and pass in parameters.
Dogs with Names

class Dog:
    def __init__(self, n):
        self.name = n

spot = Dog("Spot")
spot.name
Spot
bo = Dog("Bo")
bo.name
Bo

__init__ is a constructor
It creates a new object of the type.
It is called when Dog(n) is evaluated.

Summary

• An object packages state and procedures.
• A class provides procedures for making and manipulating a type of object.
• The procedures for manipulating objects are called methods. We invoke a method on an object.
• Friday: Inheritance
• Monday: Excursion on Exponential Growth
  — Please ready Tyson essay before Monday!