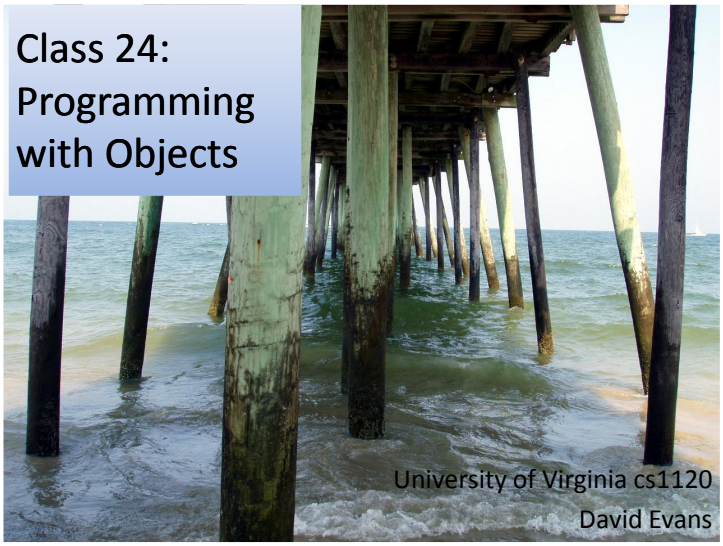
Class 24: Programming with Objects



University of Virginia cs1120

David Evans

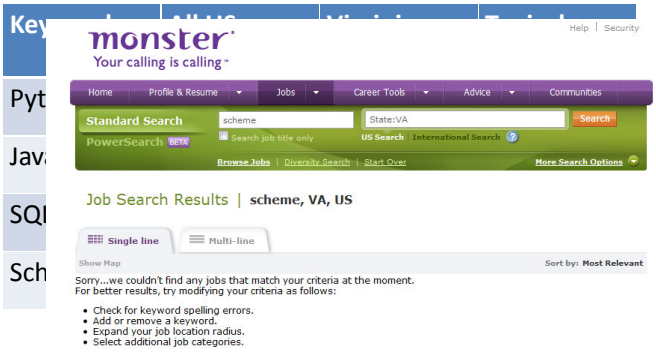
Menu

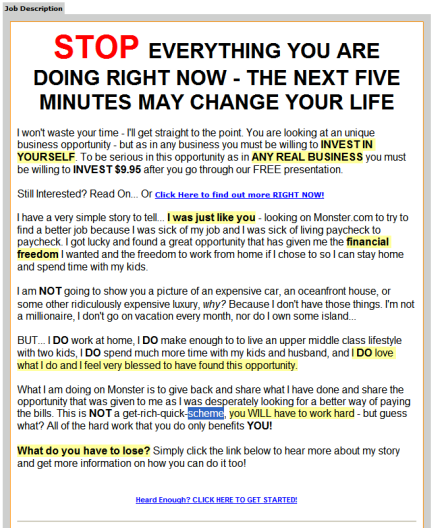
- Python
- Programming with Objects
- Inheritance

Why learn Python?

Reason 1: Vocational Skill

Job listings at monster.com (20 October 2009)





Job Description

STOP EVERYTHING YOU ARE DOING RIGHT NOW - THE NEXT FIVE MINUTES MAY CHANGE YOUR LIFE

I won't waste your time - I'll get straight to the point. You are looking at an unique business opportunity - but as in any business you must be willing to **INVEST IN YOURSELF**. To be serious in this opportunity as in **ANY REAL BUSINESS** you must be willing to **INVEST $9.95** after you go through our FREE presentation.

Still Interested? Read On... Or **Click Here to find out more RIGHT NOW!**

I have a very simple story to tell... **I was just like you** - looking on Monster.com to try to find a better job because I was sick of my job and I was sick of living paycheck to paycheck. I got lucky and found a great opportunity that has given me the **financial freedom** I wanted and the freedom to work from home if I chose to so I can stay home and spend time with my kids.

I am **NOT** going to show you a picture of an expensive car, an oceanfront house, or some other ridiculously expensive luxury, *why?* Because I don't have those things. I'm not a millionaire, I don't go on vacation every month, nor do I own some island...

BUT... I **DO** work at home, I **DO** make enough to to live an upper middle class lifestyle with two kids, I **DO** spend much more time with my kids and husband, and I **DO** love what I do and I feel very blessed to have found this opportunity.

What I am doing on Monster is to give back and share what I have done and share the opportunity that was given to me as I was desperately looking for a better way of paying the bills. This is **NOT** a get-rich-quick-scheme, you WILL have to work hard - but guess what? All of the hard work that you do only benefits **YOU!**

What do you have to lose? Simply click the link below to hear more about my story and get more information on how you can do it too!

Heard Enough? CLICK HERE TO GET STARTED!

"Scheme" Jobs

Reason 1: Vocational Skill

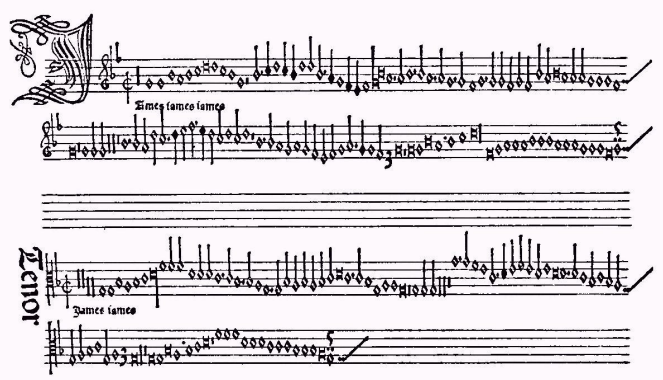
Job listings at monster.com (20 October 2009)

Keyword	All US	Virginia Only	Typical Salary
Python			
Java			
SQL			
Scheme	83	0	$100-999K

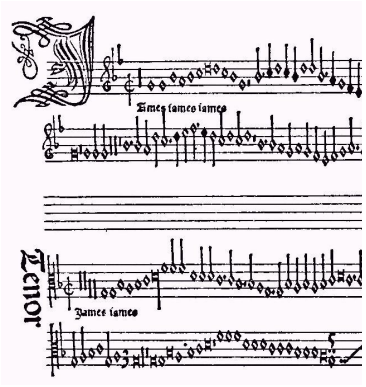
Reason 2: Expanding Minds

Languages change the way we think.

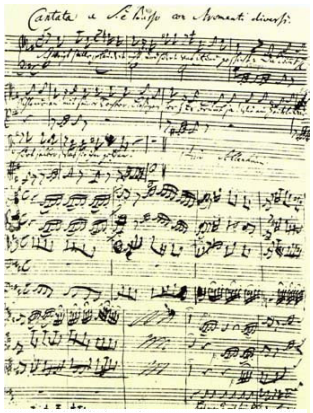
The more languages you know, the more different ways you have of expressing ideas and for thinking about (and solving) problems.



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



"Jamais Jamais Jamais" from *Harmonice Musices Odhecaton A.* (1501)



J S Bach, "Coffee Cantata", BWV 211 (1732)
www.npj.com/homepage/teritowe/jsbhand.html

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).

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).

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

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.

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***) :** $Block_{\text{Consequent}}$
else: $Block_{\text{Alternate}}$

Evaluate *Expression*. If it evaluates to a true value, evaluate the $Block_{\text{Consequent}}$; otherwise, evaluate the $Block_{\text{Alternate}}$.

Similar to (if *Expression* (begin $Block_{\text{Consequent}}$) (begin $Block_{\text{Alternate}}$))
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

```
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

ClassDefinition ::= **class** *Name***:**
FunctionDefinitions

```
class Dog:
  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

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

spot = Dog()

AssignmentStatement ::= *Variable* **=** *Expression*

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*.

Python Procedures

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

FunctionDefinition ::= **def** *Name* **(** *Parameters* **):** *Block*
Parameters ::= ε | *SomeParameters*
SomeParameters ::= ***Name*** **|** ***Name,*** *SomeParameters*
Block ::= *Statement*
Block ::= **<newline>** ***indented***(*Statements*)
Statements ::= *Statement* **<newline>** *MoreStatments*
MoreStatements ::= ε | *Statement* **<newline>** *MoreStatments*

Some Python Procedures

FunctionDefinition ::= **def** *Name* **(** *Parameters* **):** *Block*
Parameters ::= ε | *SomeParameters*
SomeParameters ::= ***Name*** **|** ***Name,*** *SomeParameters*
Block ::= *Statement*
Block ::= **<newline>** ***indented***(*Statements*)
Statements ::= *Statement* **<newline>** *MoreStatments*
MoreStatements ::= ε | *Statement* **<newline>** *MoreStatments*

```
def square(x):
    return x * x
```

```
def bigger(a,b):
    if a > b:
        return a
    else:
        return b
```

Whitespace Matters!

```
def bigger(a,b):
    if a > b:
        return a
    else:
        return b
```

```
def bigger(a,b):
    if a > b:
        return a
   else:
        return b
```

File "<pyshell#1>", line 4
else:

^
IndentationError: unindent does not match any outer indentation level

Python requires you to format your code structurally!

Barking: Invoking Methods

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

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

ApplicationStatement ::= *Name* **(** *Arguments* **)**
Arguments ::= ε | *MoreArguments*
MoreArguments ::= *Argument* **,** *MoreArguments*
MoreArguments ::= *Argument*
Argument ::= *Expression*

<obj>.<method>(<arguments>)
Invoke method on obj. The obj will be the first (self) parameter to the method.

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!**