The current topic: Python

- ✓ Introduction
  - ✓ reasons for studying languages
  - ✓ language classifications
  - ✓ simple syntax specification
- Object-oriented programming: Python
- Types and values
- Syntax and semantics
- Functional programming: Scheme
- Exceptions
- Logic programming: Prolog

Announcements

- The course information sheet is now available.
- Office hours will be M1:30-2:30 and W11-12 in SF3207.
- Make sure you use an ECF email address when signing up for a bulletin board account.

Python

- A scripting language.
  - Script: A chunk of text that gets executed without any fuss.
    - origins: “shell scripts” in Unix, and later batch files in DOS.
- Though called “scripting languages”, Python and Perl are full-scale programming languages. (We won’t cover Perl here; like Python it's often used on web servers and for general system administration tasks.)
- An evolving language.
  - In this course, we'll use version 2.5 (more precisely, version 2.5.2, the latest release).
- An object-oriented language.
  - similar in some ways to Java and C++, but with important and interesting differences.

Python

- No separate compilation phase.
  - At runtime, your program is first converted into *byte code* and then the byte code is run by the Python Virtual Machine.
  - The byte code is platform-independent.
  - "Byte code" and "virtual machine" should remind you of Java. The key difference is that with Python, this is all dealt with in the background.
  - The lack of compilation to machine code (as happens with C/C++) can lead to slower performance, but it's not as bad as you might expect:
    - Many of Python's built-in functions are implemented in C.
- Can make developers more productive.
  - Syntax is simple.
  - Proper indentation is required. This contributes towards simpler syntax since it eliminates the need to explicitly mark the beginning and end of a block (like the body of a loop).
  - The result is readable code.
- Makes programming fun!

Python Trivia

- Released in 1991 by Guido van Rossum.
- Named after Monty Python!
- Python is used by:
  - Google
  - YouTube
  - BitTorrent
  - Industrial Light & Magic
  - Pixar
  - NASA
  - ...and others!
  - Source: Mark Lutz, *Learning Python*, O'Reilly, 2008.

Features

- non-C-like syntax
  - It's good to learn a language that looks different.
- multiple inheritance
  - like C++, but unlike Java
- garbage collection
  - like Java, but unlike C++
- exceptions
- built-in lists, tuples, and dictionaries
  - dictionaries are like maps in C++ and Java

Running Python

- You can run individual commands within the Python interpreter.
- You can run entire programs from the command-line or within an IDE
  - IDEs include IDLE, and Eclipse with the Pydev plugin.
- On ECF, the default installation of Python (in `/usr/bin/python`) is rather old (version 2.3). I've installed the latest version (2.5.2) in a public area of my home directory (that is, `~ajuma/share/bin/python2.5`). **Your assignments will be tested using version 2.5.2.** The easiest way to run this version on ECF is to add `~ajuma/share/bin` to your PATH environment variable; then you can just use the command `python2.5` to run the correct version of Python.

Running Python programs

- Say your program is in the file `prog.py`.
  - The ".py" extension isn't always required, but is a good practice to follow.
- Option 1: Use the following command.
  ```
  python2.5 prog.py
  ```
- Option 2: Make `prog.py` an executable script. This works on Unix-like systems (e.g. Linux, Mac OS X, Cygwin).
  - Make the following the first line of the file:
    ```
    #!/u/prof/ajuma/share/bin/python2.5
    ```
    (On non-ECF machines, use the actual path to python instead.)
  - Set the file's permissions, making the file executable by its owner (you):
    ```
    chmod 700 prog.py
    ```
  - You can now run the program using the command:
    ```
    ./prog.py
    ```

Variables

- In Python, variables are just names for values.
- Variables are not declared.
  - Unlike C or Java.
  - Variables are created by their first use:
    ```
    lis = [0, 1]   #creates a variable called lis
    ```
    - Observe that "#" is used for comments.
- ***Variables* have no types, but *values* do have types.**
  - Python is *dynamically typed* (since types don't need to be declared).
  - Python is *strongly typed:*
    - Types are checked and type restrictions are enforced when performing operations.
  - Example:
    ```
    x = 123
    x = "hello"  #x's value was first an int, now it's a string
    y = 456
    x + y    #This produces a TypeError.
    ```


Variables

- Variables cannot be used before they receive values.
  - There is no default value.
  - Example:
    ```
    z+2  #This produces a NameError.
    ```


Strings

- Use single or double quotation marks for string literals:
  ```
  'hi' + "there"  #this produces 'hithere'
  ```
  - Note that the + operator is used for concatenation.
- Converting a value to a string: use the `str()` function:
  ```
  str(13)   #produces the string '13'
  ```
- Getting multiple copies of a string: use the * operator:
  ```
  'ha'*3    #produces 'hahaha'
  ```
- Converting a string to an int or a float:
  ```
  int('543')    #produces the int 543
  float('4324.4343')   #produces the float 4324.4343
  ```


Numbers

- Three types:
  - Integers
    - e.g. 34, -43, 0
    - automatic conversion to "long integers" (which can grow as large as needed) when the value overflows a standard integer.
      ```
      2**100  #correctly calculates 2^100, no overflow error.
      ```
    - "long integers" are displayed with a L at the end: e.g. 1099511627776L
  - Floating-point numbers
    - e.g. 43.3, 10.0, -3.3
  - Complex numbers:
    - e.g. 43 + 6j, -1j, 43.43 + 1.3j
    - Note that the letter j is used in place of i (the "imaginary" part).

Booleans

- Values are `True` and `False` (capitalized). These are of type bool.
- As in C, all values have a boolean meaning:
  - 0, None, empty structures are false.
    - Here, structures include lists, dictionaries, strings.
  - Everything else is true.
- Operators: `and, or, not`.
- `and` and `or` are short-circuit and return the last value evaluated (as opposed to returning the value `True` or `False`):

```
'a' or 'b'   #value is 'a'
'' or 'z'    #value is z
'a' and 0    #value is 0
'a' and 'b'  #value is 'b'
type('a' and 0)   # <type 'int'>
```


Comparison operators

- Operators are: >, <, >=, <=, ==, !=
  - Same as C and Java.
- Results are `True` or `False`.
- Can be chained:

```
-1 < 0 < 1   # True
```


Basic input/output

- Output to stdout using `print`.
  - Unless the statement ends with a comma, the output ends with a newline character.

```
print "hello"               #output ends with a newline
print "hello no new line",  #output does not end with a
                           #newline
print "hello", "world", 123, "csc"  #commas cause items to
                                    #be separated by spaces
print "hello"+"world"+str(123)+"csc"  #no spaces
```

- Read from stdin using `raw_input()`.
  - Optionally, you can pass a prompt to raw_input().

```
name = raw_input("Please enter your name:")
```


If statements and while loops

- The symbol ":" and indenting are used to denote the body of the statement.

```
a = 3
if a < 0:
    print "less"
elif a == 0:     #Observe it's *not* "else if".
    print "equal"
    print "other stuff"
else:
    print "greater"

while a > 0:
    print a
    a -= 1   #There is no ++ or -- operator.
```

Exercises

- These are good things to ask about in tutorial (after you've tried working on them).

1. Add ~ajuma/share/bin to your PATH environment variable, and make sure you can run python2.5.
2. Write a simple "Hello World" program, and run it both of the ways described on slide 8.
3. Modify your program to ask the user for their name, and then output "Hello <user's name>".
4. Write a program that asks the user to input an integer n, and then outputs the first n Fibonacci numbers. The Fibonacci sequence is defined as follows:
   $F_1 = 1$, $F_2 = 1$, $F_k = F_{k-2} + F_{k-1}$ for $k > 2$.
   So the first ten Fibonacci numbers are 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.