MIT AITI

Accelerating Information Technology Innovation

http://aiti.mit.edu

Kenya Summer 2011

Lecture 1 – Introduction to Python

MISTI

Agenda

- About the Course
- What is Python?
- Why Python, in general?
- Why Python, for us?
- The Development Cycle
- Basic Syntax

About the Course

Course Outline

- Week 1 - Basic Python
    - Introduction to Python
    - Variable and Operators
    - Control Structures
- Week 2 - Intermediate Python
    - Data Structures
    - Functions
    - Objects
    - Inheritance
    - Exceptions

Course Outline

- Week 3 - Advanced Python
    - Regular Expressions
    - Becoming a Python Ninja
    - Useful Libraries and Functions
    - Django
- Week 4
    - Google App Engine
    - Client Interfaces (Mobile Web)
    - Start Final Project
- Weeks 5 & 6
    - Work on Final Project

Course Expectations

- Attend class every day
- Arrive to class on time
- Collaborate
- Teach others as much as you can
- Do everything you can in the labs
- Ask questions!

Course Website and Mailing List

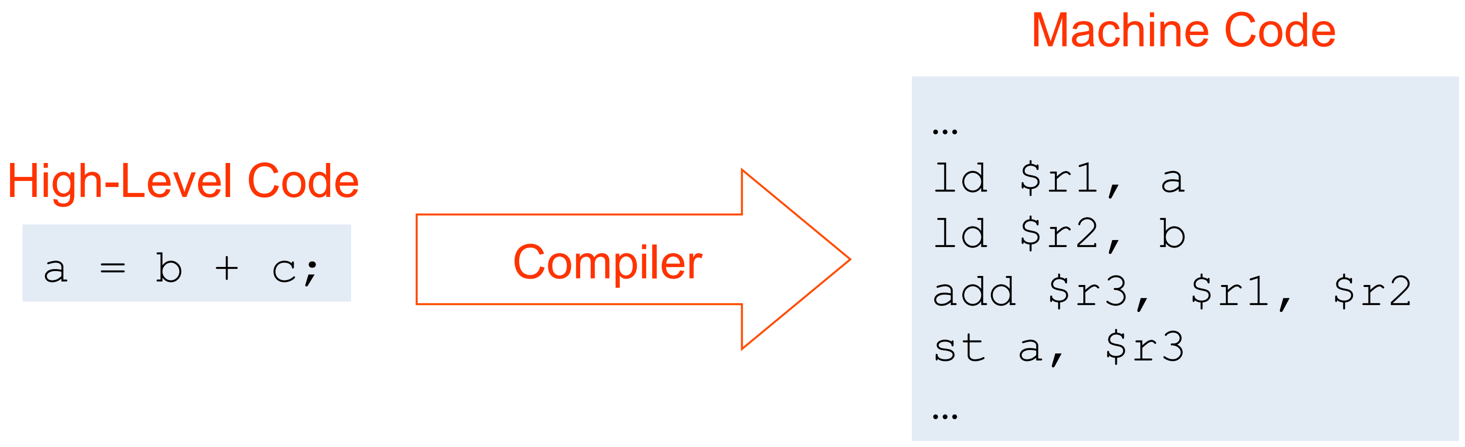
- Lectures and labs will be posted at:
  - http://aiti.mit.edu/app/materials/kenya-summer-2011/

- Official mailing list for the course is:
  - aiti-kenya-2011-summer-class@mit.edu

What is Python?

Python is…

- …*interpreted.* Languages like C/C++ require *compilers* to translate high-level code to machine code…

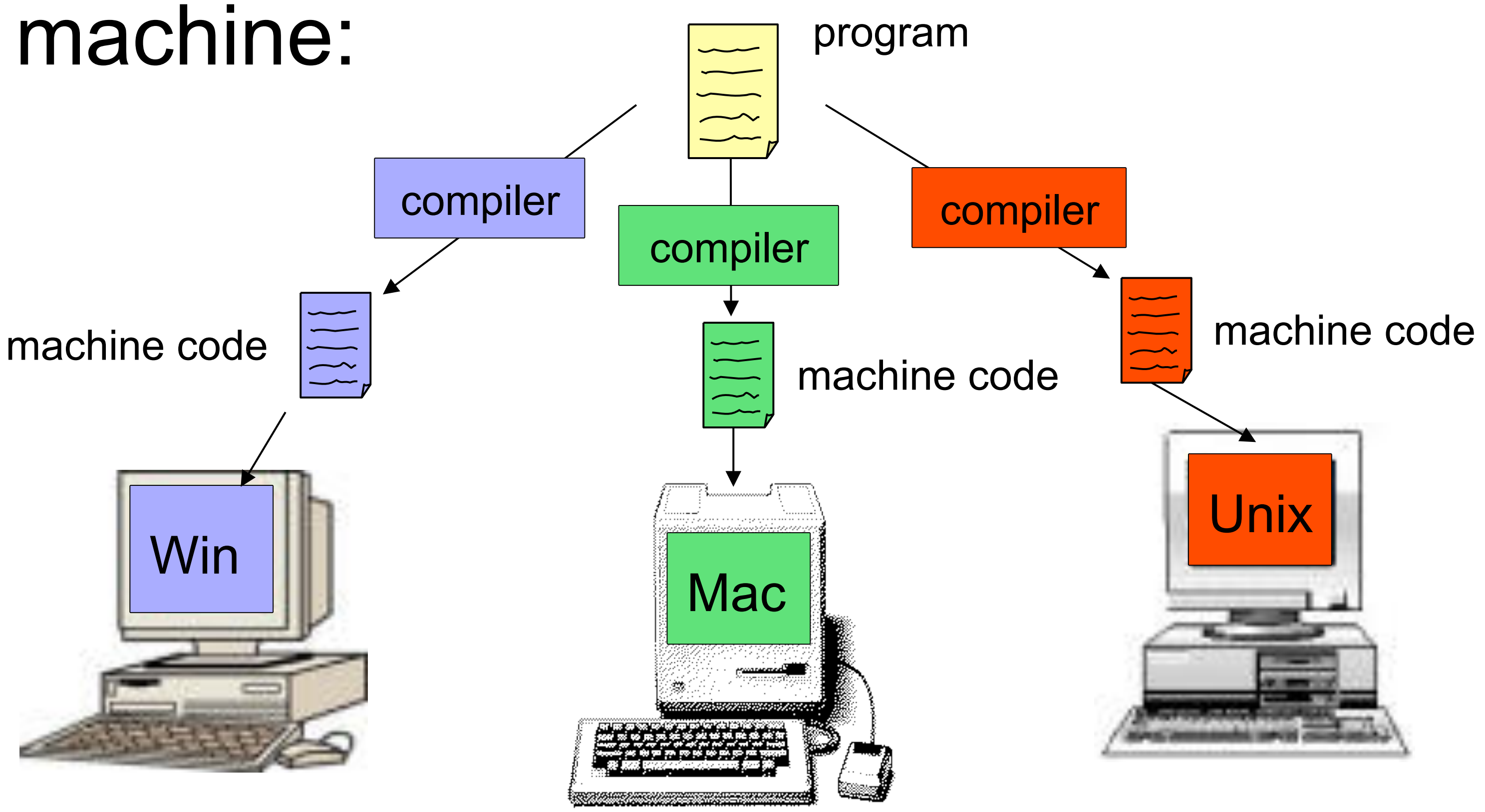


```
a = b + c;
```

```
…
ld $r1, a
ld $r2, b
add $r3, $r1, $r2
st a, $r3
…
```

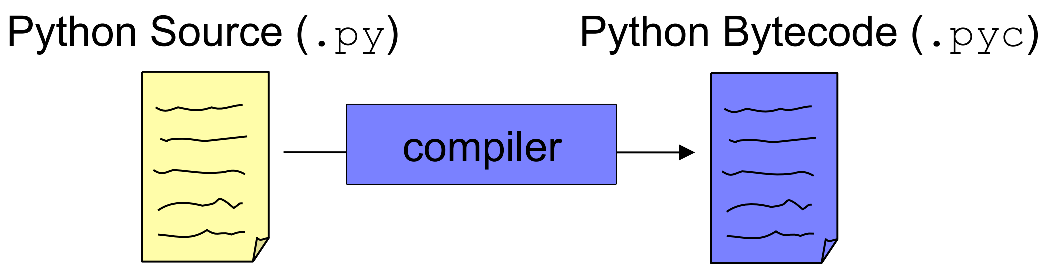
Python is…

- …which means that a program has to be compiled separately for each type of machine:



Python is…

- Python code is compiled to an intermediate format called ***bytecode***, which is understood by a *virtual machine.*

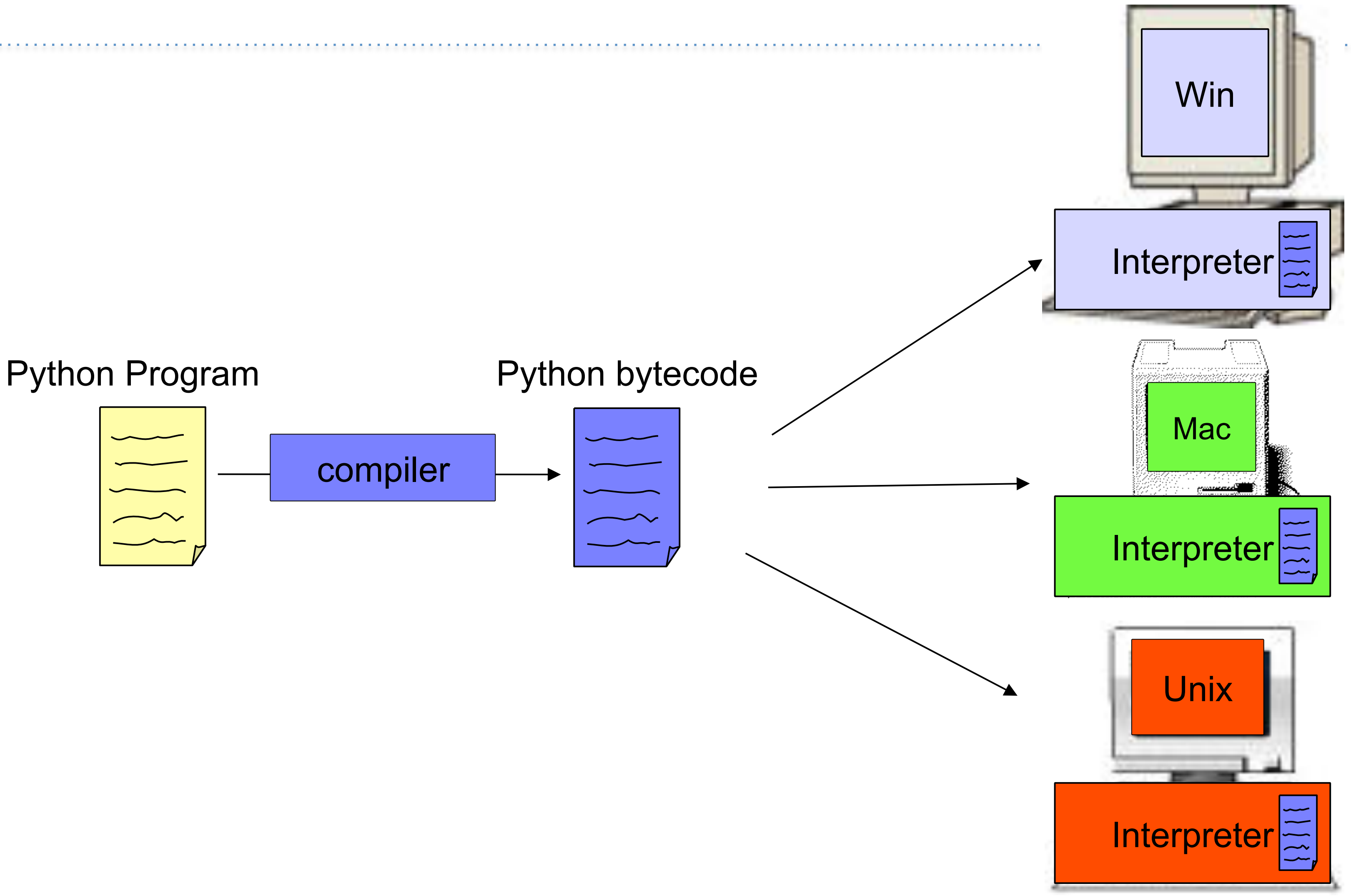


- 'Write Once, Run Anywhere'

Python is…

- This is accomplished through the use of Python virtual machines, or *interpreters,* which are built on each type of machine.
- The interpreter simulates the VM bytecode on the actual hardware, translating the VM's 'native' calls to machine code.
- This presents a standard interface to the language, allowing portability

Python is…


Python Program
compiler
Python bytecode
Win
Interpreter
Mac
Interpreter
Unix
Interpreter

Python is…

- Interestingly, implementations exist for other VMs on the same hardware:
  - Jython – compiles to Java VM bytecode
  - Iron Python – compiles to .NET bytecode

Python is…

- Dynamically typed; variable types are determined at runtime depending on what you assign to them:

```
# int
a = 1
# string
a = “a”
# list
a = [1,2,3]
# dictionary
a = {1:2,3:4}
```

Why Python?

Python because…

- Portable and architecture-agnostic
- Convenient built-in functions and data structures
- Syntax is readable and fast to write

```
if (x)
{
    if (y)
    {
        a();
    }
    b();
}
```

```
if x:
    if y:
        a()
    b()
```

Python because…

- Great for rapid prototyping
  - No separate compile step
  - No need to explicitly specify method argument types beforehand (due to dynamic typing)

Why Python, For Us?

Python for us, because…

- We want each of you to reach millions of users, and don’t want to waste time building the pipes and plumbing
- Python is supported by a number of good frameworks, led by
  - Google AppEngine
  - Django

The Development Cycle

The (Ideal) Development Cycle

- *Clearly* specify the problem:
  - Inputs, input manipulation, outputs
- Design the solution:
  - E.g what algorithms, data structures
- Implementation:
  - Coding!
- Test, test, test
  - Strongly suggest unit testing with PyUnit

The (Real) Development Cycle

- As above, but *faster*.
  - Python, as a dynamically typed, dynamic language is perfect for *rapid* prototyping
- Be prepared to throw away one (or more!) prototypes
  - Often you learn crucial things about the problem as you code which cannot be fixed without starting from scratch.

Strong Recommendations

- Use self-documenting variable names
  - e.g. “name” instead of “n”
- Use full length variable names
  - e.g. “custom_presenter” not “custpres”
- Comment everything that’s not absolutely obvious
  - Can your team member extend some part of your code?
  - Can you read your own code in 10 years?

Basic Syntax

Syntax

- Blocks are delimited with whitespace: specifically, four spaces (and no tabs)

```
if x:
    if y:
        a()
    b()
```

```
count = 0
for i in range(0:5)
    count += i
```

Syntax

- Semicolons are only used to separate multiple statements on the same line, which is discouraged:

```
if (x)
{
    a();
    b();
}
```

```
if x:
    a(); b()
```

Syntax

- Single line comments are denoted with hash (#), multiline with three quotes ”””

```
# This is a comment
foo()
```

```
"""
This is a
longer comment
"""
foo()
```

Interaction

- Python has an interactive console which is great for tinkering

```
$ python
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:13:53)
[GCC 4.5.2] on linux2
Type “help”, “copyright”, “credits” or “license” for
more information
>>> a = 1
>>> a
1
>>> type(a)
<type ‘int’>
>>>
```

- …etc

Questions?