Introduction:
Python

Today's agenda

- What is Python?
- Why Python?
- The Development Cycle
- Basic Syntax

Python is…

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

High-Level Code

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

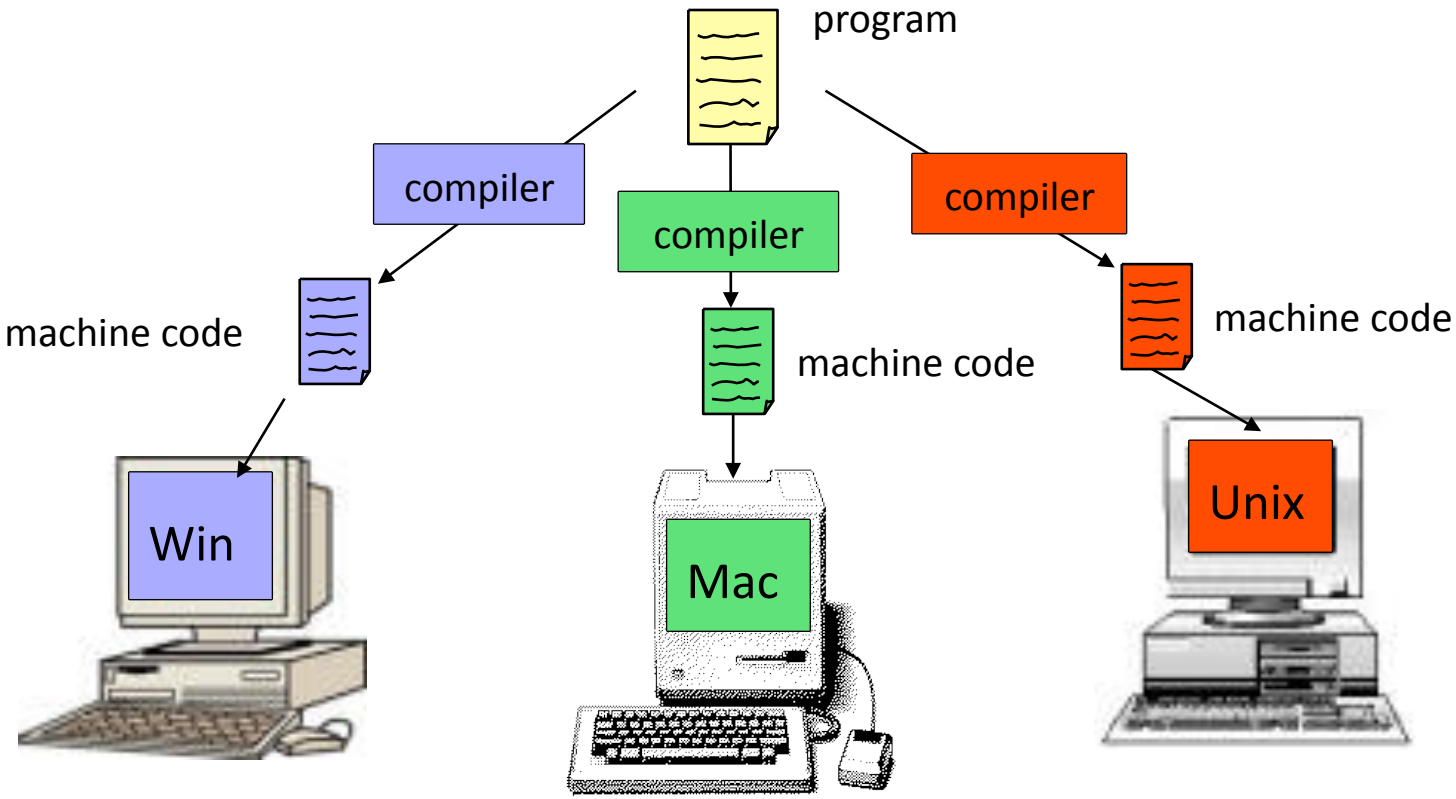
Compiler

Machine Code

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

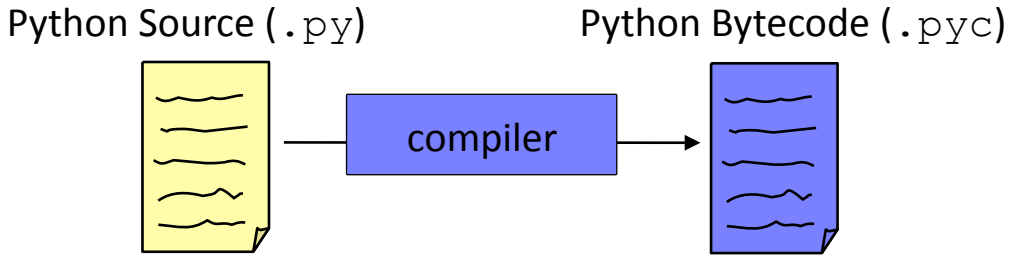
Python is…

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



Python is…

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

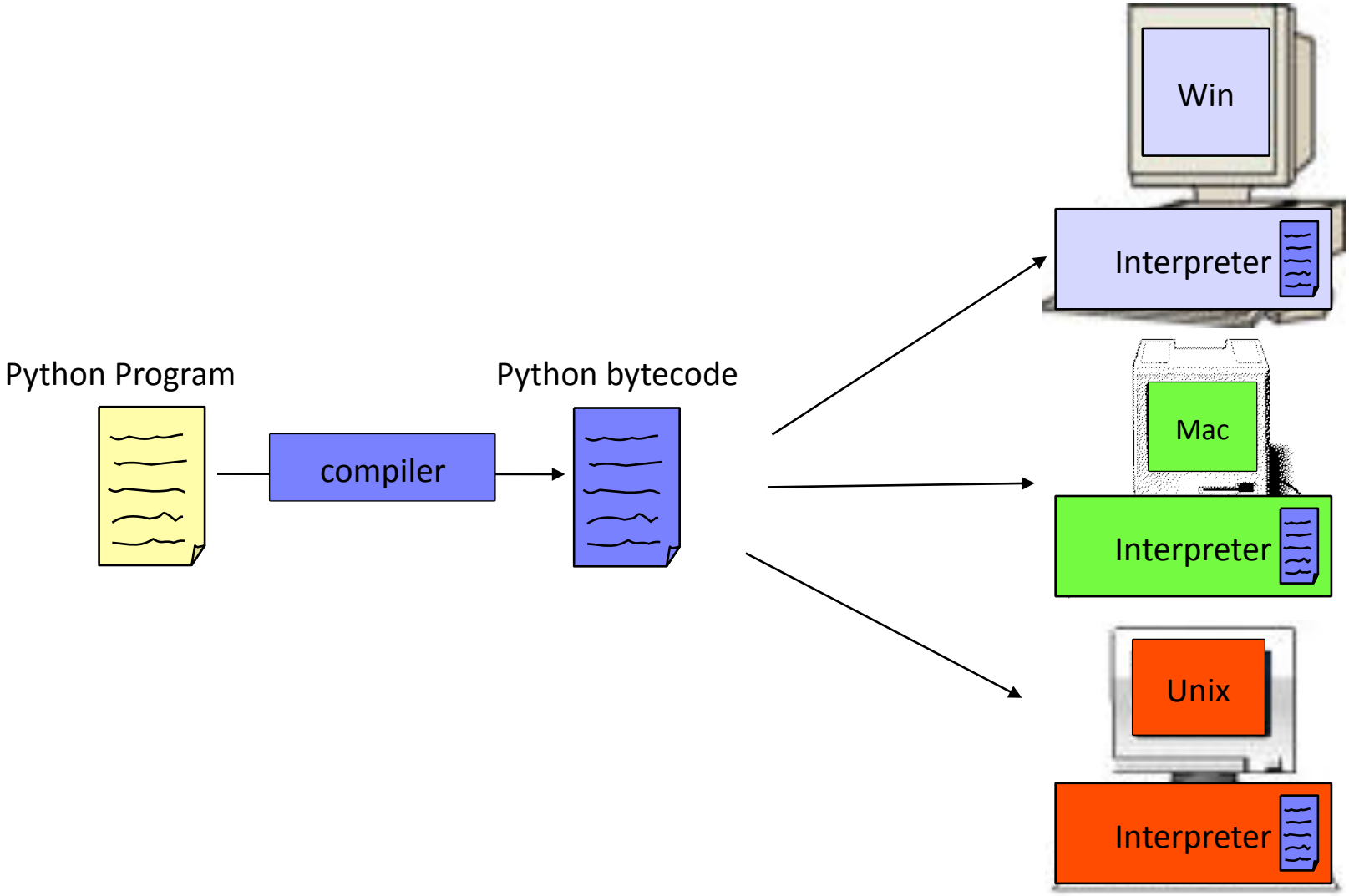


- This model is similar to Java's and is designed to allow you to 'Write Once, Run Anywhere'

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…

- 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}
```

Today's agenda

- What is Python?
- Why Python?
- The Development Cycle
- Basic Syntax

Python because…

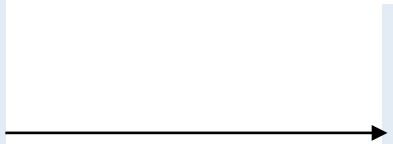
- Python's interpreted nature makes it portable and architecture-agnostic; if a machine runs Python, it'll run your code.

- Python, like Java, includes many convenient built-in functions and data structures which are already optimized for its virtual machine.

Python because…

- Python's syntax is designed to be readable and fast to write. In addition to dynamic typing, whitespace is used as a block delimiter, and semicolons are not usually necessary:

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



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

Python because…

- Useful data structures (dictionaries, tuples, lists, etc.) are built-in and do not need to be separately imported
- Lack of a separate compile step speeds rapid prototyping and debugging
- Dynamic typing speeds up development – no need to explicitly specify method argument types beforehand

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

Today's agenda

- What is Python?
- Why Python?
- The Development Cycle
- Basic Syntax

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 camelcase for class names
  - e.g. “CustomPresenter” not “custpres”
  - More style tips at http://www.python.org/dev/peps/pep-0008/
- Comment everything that’s not absolutely obvious
  - Can you read your own code in 10 years?

Today's agenda

- What is Python?
- Why Python?
- The Development Cycle
- Basic Syntax

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

Syntax

- As mentioned before, blocks are delimited with whitespace: use spaces, not tabs (most python editors will convert tabs to spaces for you)

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

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

Syntax

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

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

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