

Python

Cam Allen

cam@cs.duke.edu

Based on slides by Zhenyu Zhou, Richard Guo



What is Python?

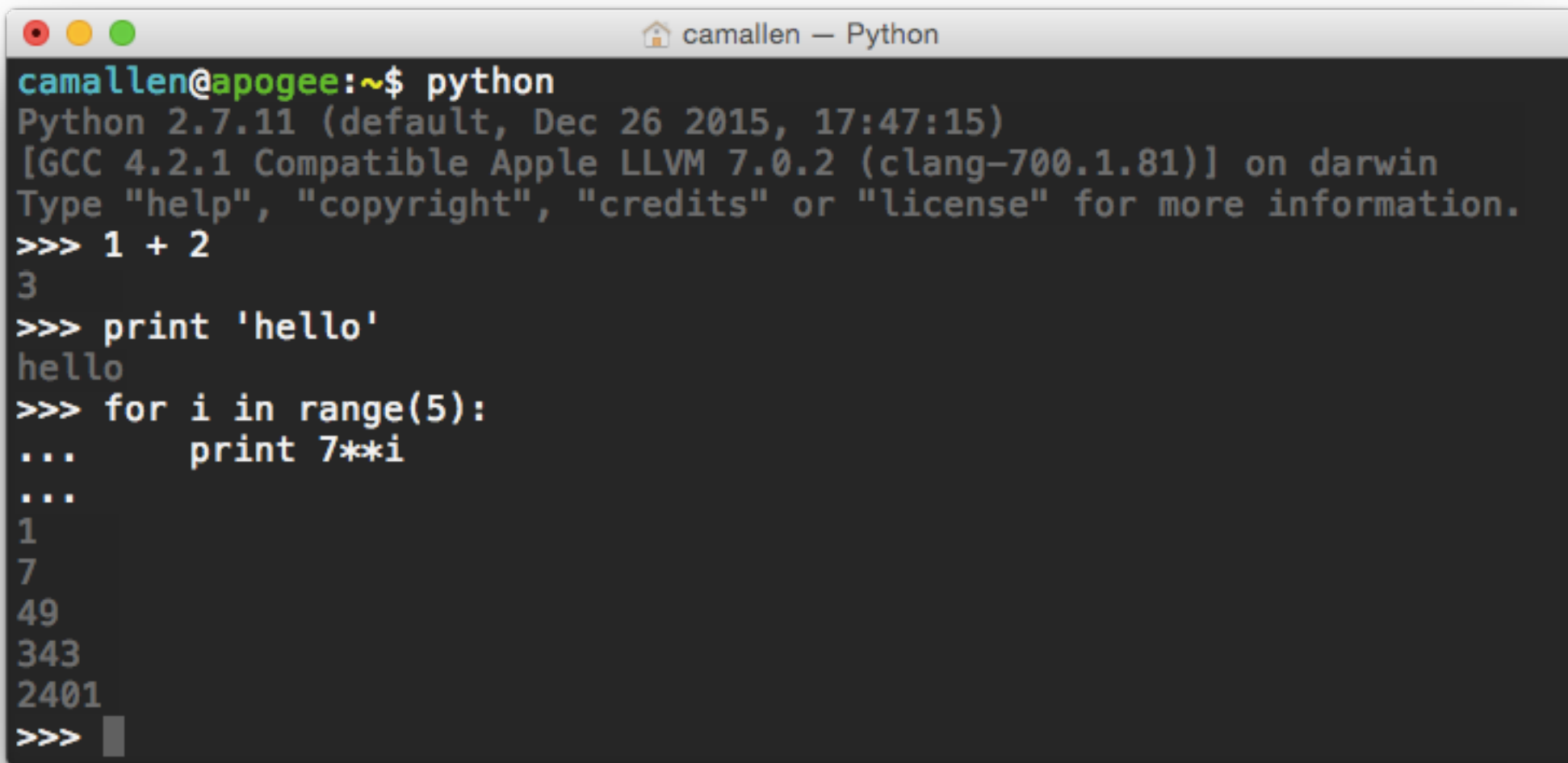


Language Principles

- **Beautiful** is better than **ugly**
- **Explicit** is better than **implicit**
- **Simple** is better than **complex**
- **Complex** is better than **complicated**
- **Readability** counts

—The Zen of Python

The Interpreter

A screenshot of a Python interpreter window on a Mac. The window has a title bar with three colored buttons (red, yellow, green) on the left and a title "camallen — Python" on the right. The main area is a dark gray terminal with white text. The text shows the user running the 'python' command, which starts the Python 2.7.11 interpreter. The user then enters several commands: '1 + 2', 'print 'hello'', and a for loop that prints powers of 7. The output shows the results of these commands.

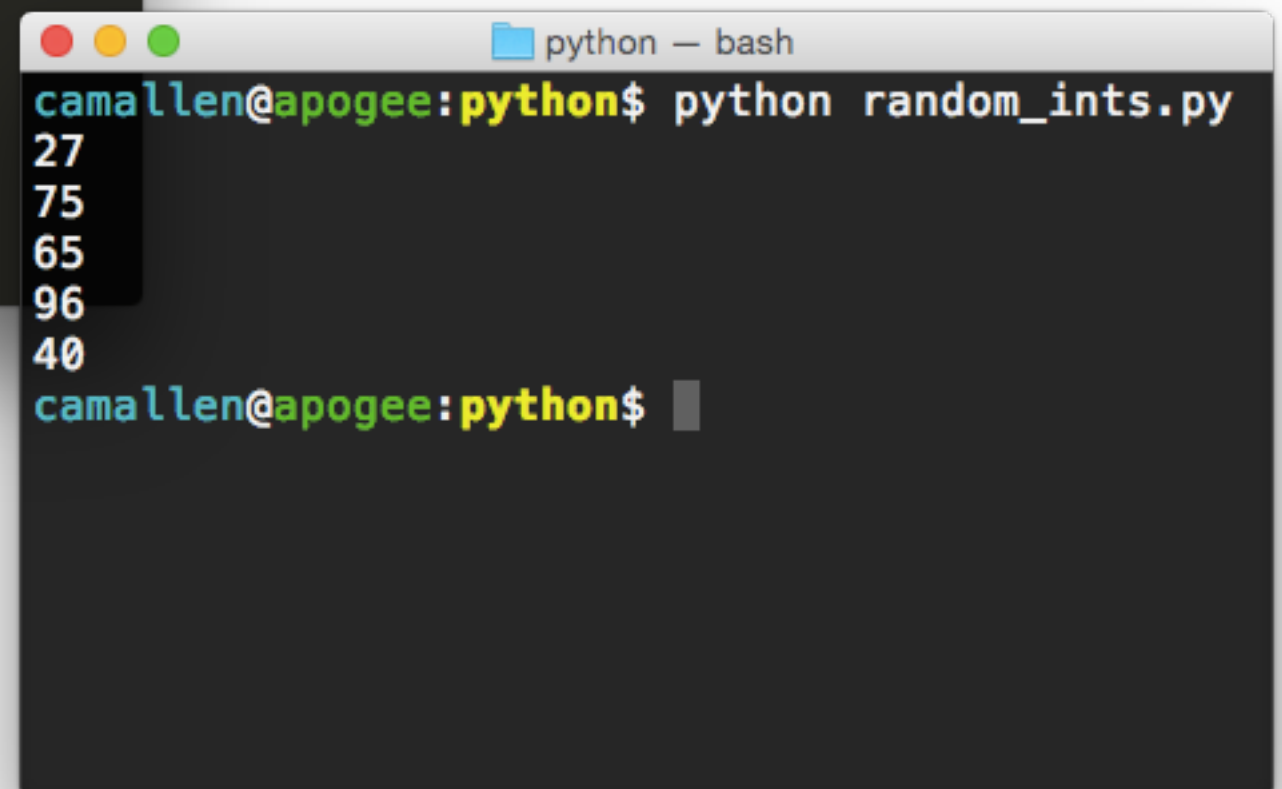
```
camallen@apogee:~$ python
Python 2.7.11 (default, Dec 26 2015, 17:47:15)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 2
3
>>> print 'hello'
hello
>>> for i in range(5):
...     print 7**i
...
1
7
49
343
2401
>>>
```

Running Scripts



A screenshot of a code editor window titled 'random_ints.py' with a status bar indicating 'UNREGISTERED'. The editor shows a Python script with the following code:

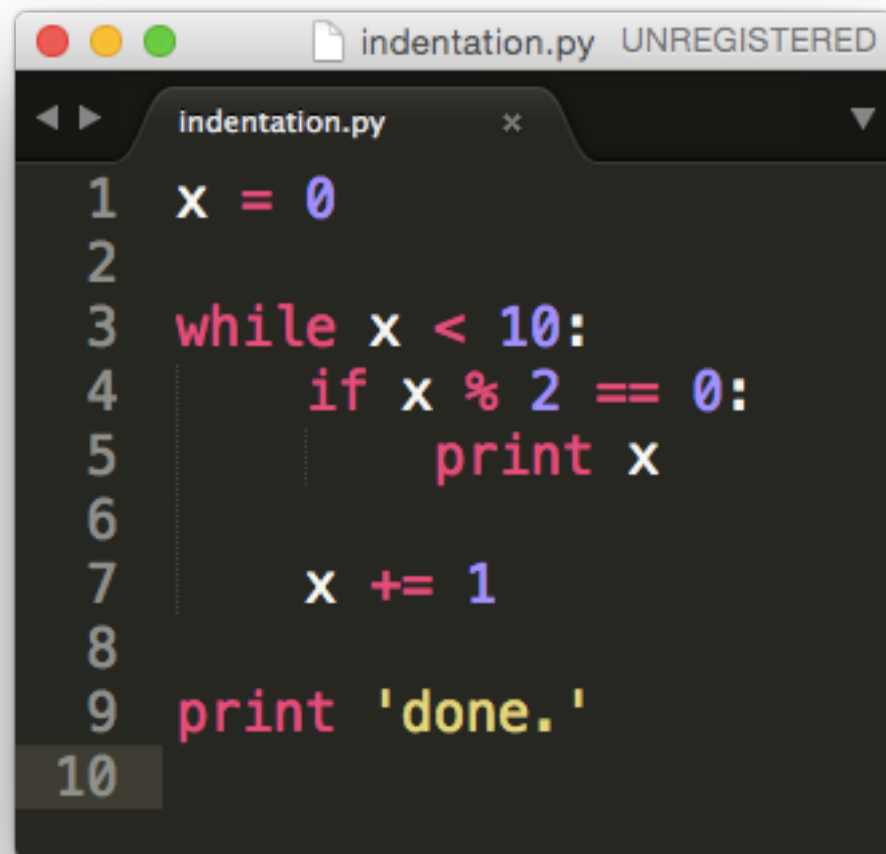
```
1 import random
2
3 for i in range(5):
4     print(random.randint(10,99))
5
```



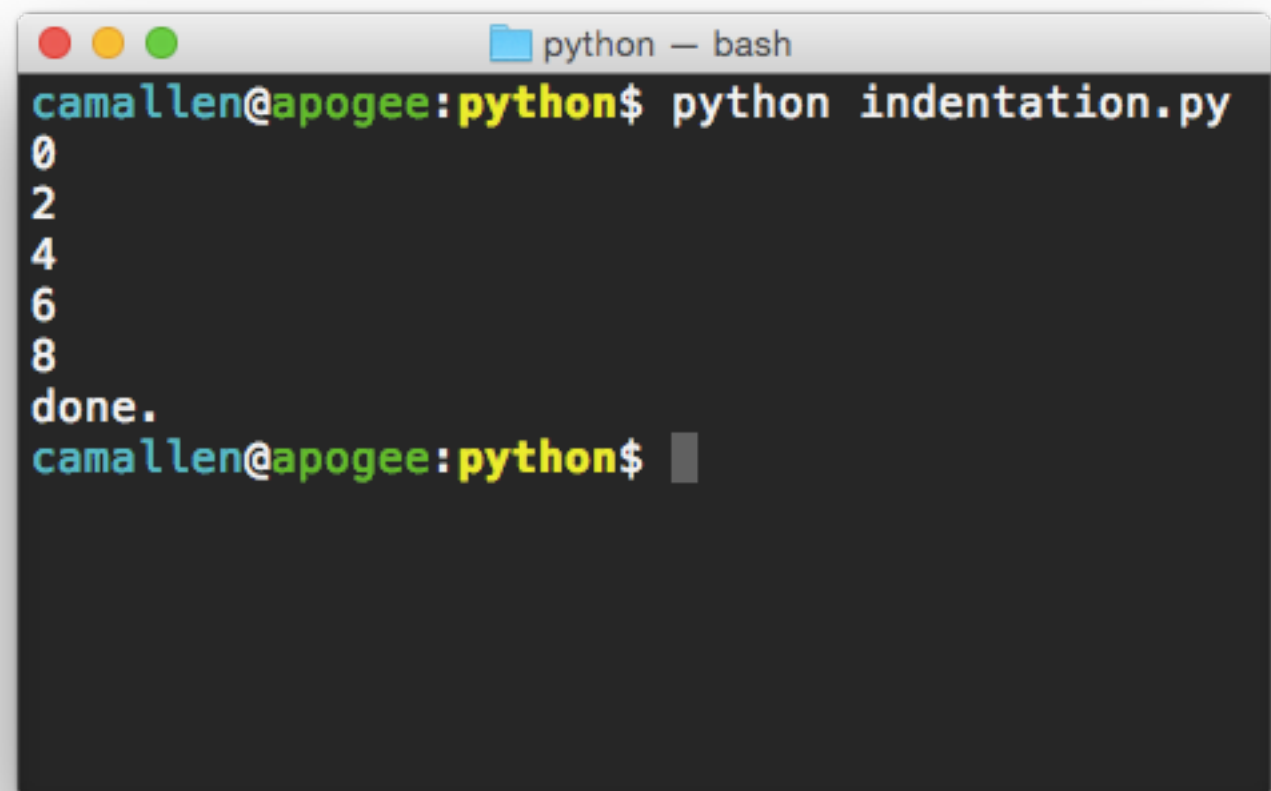
A screenshot of a terminal window titled 'python — bash'. The prompt is 'camallen@apogee:python\$'. The command 'python random_ints.py' has been executed, resulting in five lines of output: 27, 75, 65, 96, and 40. The prompt is now 'camallen@apogee:python\$'.

```
camallen@apogee:python$ python random_ints.py
27
75
65
96
40
camallen@apogee:python$
```

Indentation



```
1 x = 0
2
3 while x < 10:
4     if x % 2 == 0:
5         print x
6
7     x += 1
8
9 print 'done.'
10
```



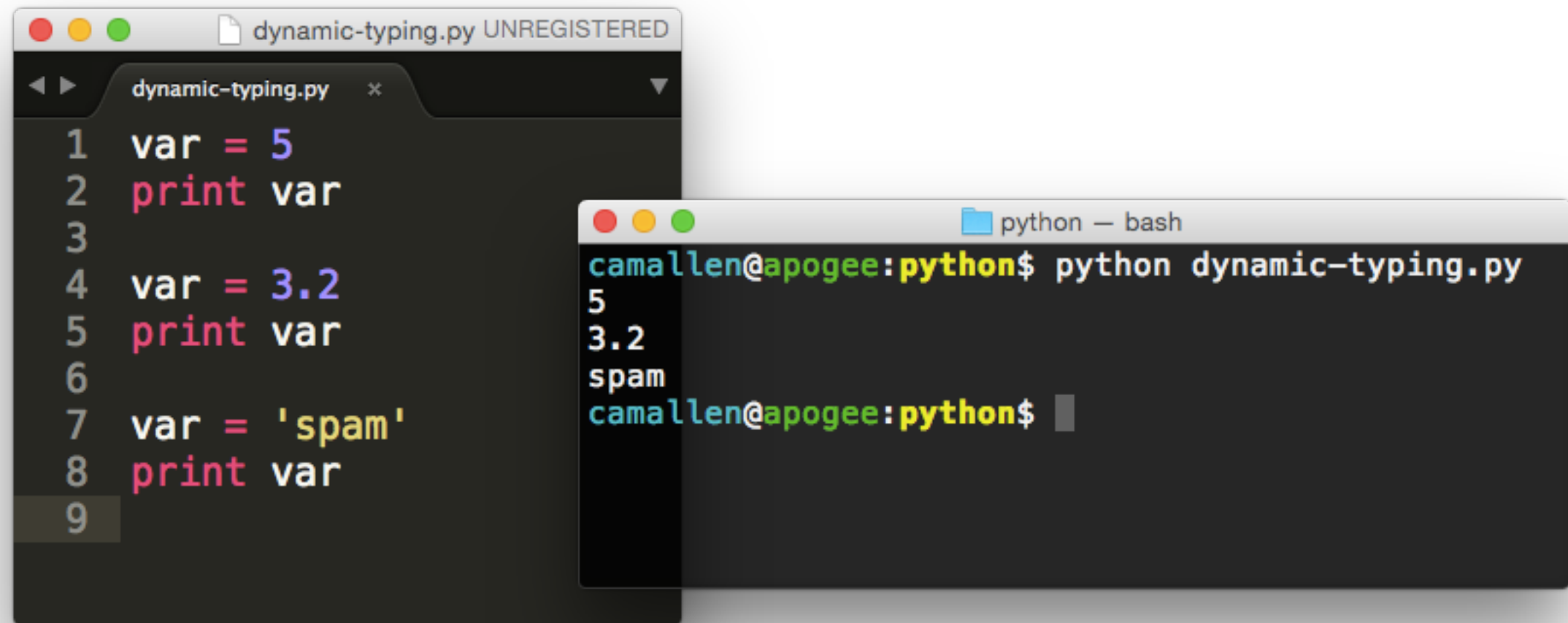
```
python — bash
camallen@apogee:python$ python indentation.py
0
2
4
6
8
done.
camallen@apogee:python$
```

Indentation Errors

```
indentation.py UNREGISTERED
1 x = 0
2
3 while x < 10:
4     if x % 2 == 0:
5         print x
6
7     x += 1
8
9 print 'done.'
10
```

```
python — bash
camallen@apogee:python$ python indentation.py
File "indentation.py", line 7
    x += 1
    ^
IndentationError: unindent does not match any
outer indentation level
camallen@apogee:python$
```

Dynamic Typing



The image shows a code editor window titled 'dynamic-typing.py UNREGISTERED' and a terminal window titled 'python — bash'. The code editor contains a Python script with three assignments and prints. The terminal shows the output of running the script, demonstrating that the variable 'var' can hold different types of values (integer, float, and string) without any explicit type declarations.

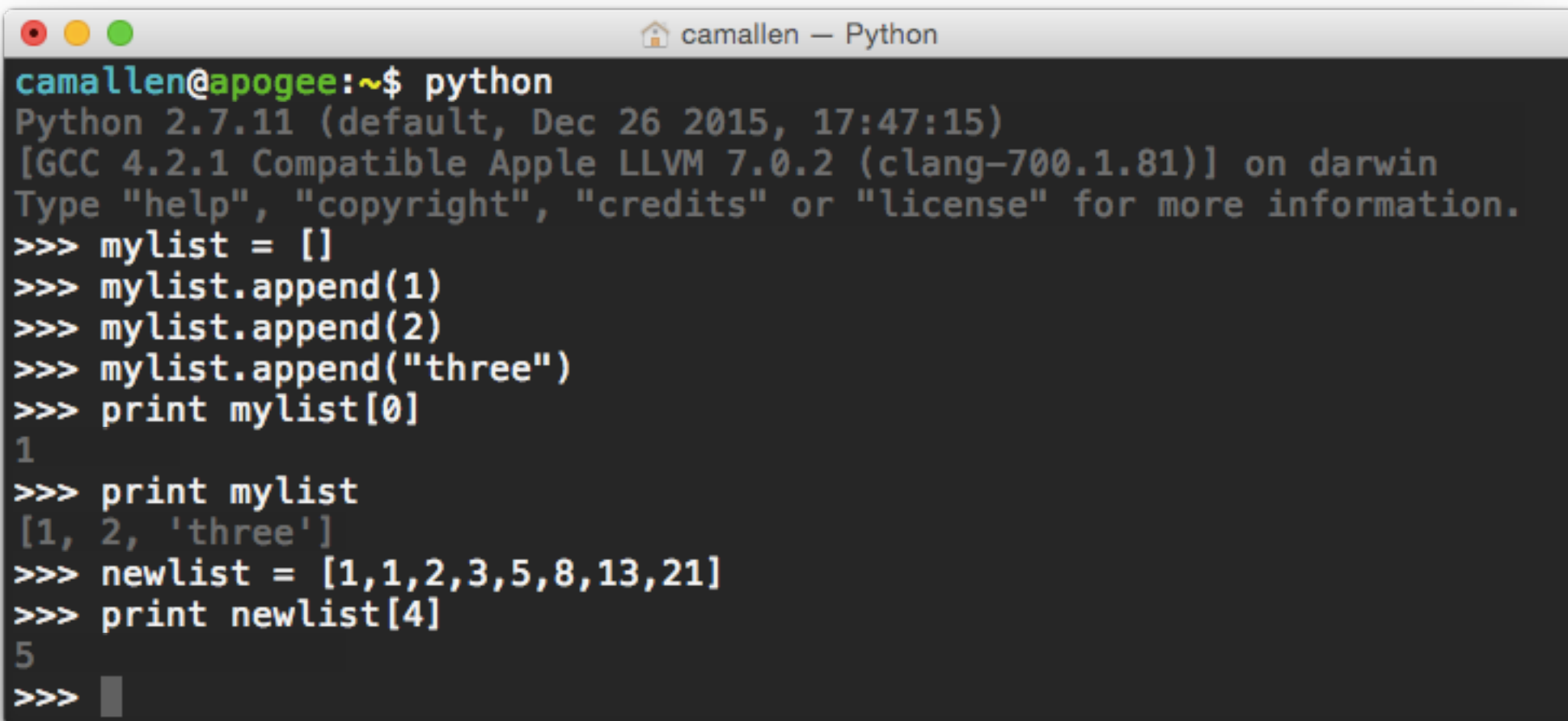
```
1 var = 5
2 print var
3
4 var = 3.2
5 print var
6
7 var = 'spam'
8 print var
9
```

```
python — bash
camallen@apogee:python$ python dynamic-typing.py
5
3.2
spam
camallen@apogee:python$
```


Strings

```
camallen@apogee:~$ python
Python 2.7.11 (default, Dec 26 2015, 17:47:15)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> mystring = 'spam and eggs'
>>> mystring[0:4]
'spam'
>>> mystring.find('and')
5
>>> mystring.split(' ')
['spam', 'and', 'eggs']
>>> mystring.upper()
'SPAM AND EGGS'
>>> print "1 + 2 = {}".format(1+2)
1 + 2 = 3
>>> █
```

Lists

A terminal window titled 'camallen — Python' with standard macOS window controls (red, yellow, green buttons). The terminal shows a Python 2.7.11 prompt where a list named 'mylist' is created and populated with the values 1, 2, and 'three'. The first element is printed, then the entire list is printed. A second list named 'newlist' is created with the values [1, 1, 2, 3, 5, 8, 13, 21], and its fifth element (index 4) is printed. The prompt is ready for further input.

```
camallen@apogee:~$ python
Python 2.7.11 (default, Dec 26 2015, 17:47:15)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> mylist = []
>>> mylist.append(1)
>>> mylist.append(2)
>>> mylist.append("three")
>>> print mylist[0]
1
>>> print mylist
[1, 2, 'three']
>>> newlist = [1,1,2,3,5,8,13,21]
>>> print newlist[4]
5
>>> █
```

Tuples

```
camallen@apogee:~$ python
Python 2.7.11 (default, Dec 26 2015, 17:47:15)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> mytuple = 'seven', 49, 343, 2401
>>> mytuple[0]
'seven'
>>> print mytuple
('seven', 49, 343, 2401)
>>> newtuple = mytuple, (1, 2, 3, 4, 5)
>>> print newtuple
(('seven', 49, 343, 2401), (1, 2, 3, 4, 5))
>>> mytuple[0] = 'eight'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> lists = (['a', 'b', 'c'], [1, 2, 3])
>>> lists[0].append('d')
>>> print lists
(['a', 'b', 'c', 'd'], [1, 2, 3])
>>> x, y = 5, 10
>>> x, y = y, x
>>> print (x, y)
(10, 5)
>>>
```

Sequence Types

Type	Example
String	<code>s = "Don't touch that dial!"</code>
List	<code>L = [1, 2, 3, 4, 5]</code>
Tuple	<code>t = ('Check', 1, 2)</code>
(more)	<code>...</code>

Sequence Types

Operation	Result
<code>x in s</code>	<code>True</code> if an item of <code>s</code> is equal to <code>x</code> , else <code>False</code>
<code>x not in s</code>	<code>False</code> if an item of <code>s</code> is equal to <code>x</code> , else <code>True</code>
<code>s + t</code>	Concatenation of <code>s</code> and <code>t</code>
<code>s * n, n * s</code>	Equivalent to adding <code>s</code> to itself <code>n</code> times
<code>s[i]</code>	The <i>i</i> th item of <code>s</code> , starting with index 0
<code>s[i:j]</code>	Slice of <code>s</code> from <i>i</i> to <i>j</i>
<code>s[i:j:k]</code>	Slice of <code>s</code> from <i>i</i> to <i>j</i> , with step <i>k</i>
<code>len(s)</code>	Length of <code>s</code>
<code>min(s)</code>	Smallest item of <code>s</code>
<code>max(s)</code>	Largest item of <code>s</code>
<code>s.index(x)</code>	Index of the first occurrence of <code>x</code> in <code>s</code>
<code>s.count(x)</code>	Total number of occurrences of <code>x</code> in <code>s</code>

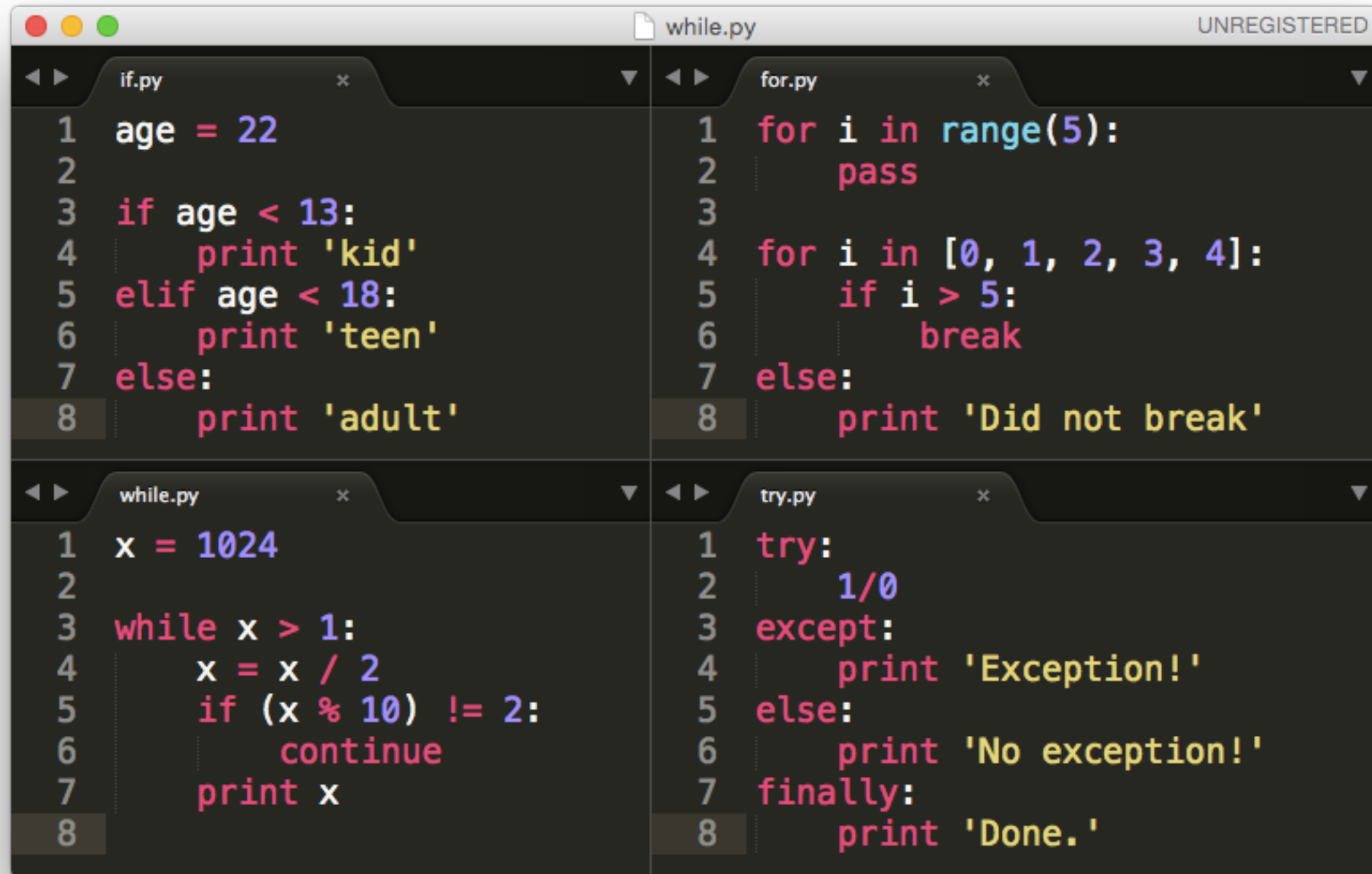
Dictionaries

```
camallen@apogee:~$ python
Python 2.7.11 (default, Dec 26 2015, 17:47:15)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> email = {'george': 'gdk@cs.duke.edu', 'cam': 'cam@cs.duke.edu'}
>>> email['ron'] = 'parr@cs.duke.edu'
>>> email
{'ron': 'parr@cs.duke.edu', 'george': 'gdk@cs.duke.edu', 'cam': 'cam@cs.duke.edu'}
>>> email['george']
'gdk@cs.duke.edu'
>>> del email['george']
>>> email
{'ron': 'parr@cs.duke.edu', 'cam': 'cam@cs.duke.edu'}
>>> email.keys()
['ron', 'cam']
>>> for key, val in email.iteritems():
...     print "{}: {}".format(key, val)
...
ron: parr@cs.duke.edu
cam: cam@cs.duke.edu
>>> 'cam' in email
True
>>>
```

Quick Recap

- Python: popular language, for good reasons
- Interactive mode and script mode
- Language basics
- Next up: control flow, functions, classes, modules

Control Flow Statements



The image shows a code editor window titled 'while.py' with a status bar indicating 'UNREGISTERED'. It contains four tabs, each showing a different Python control flow statement:

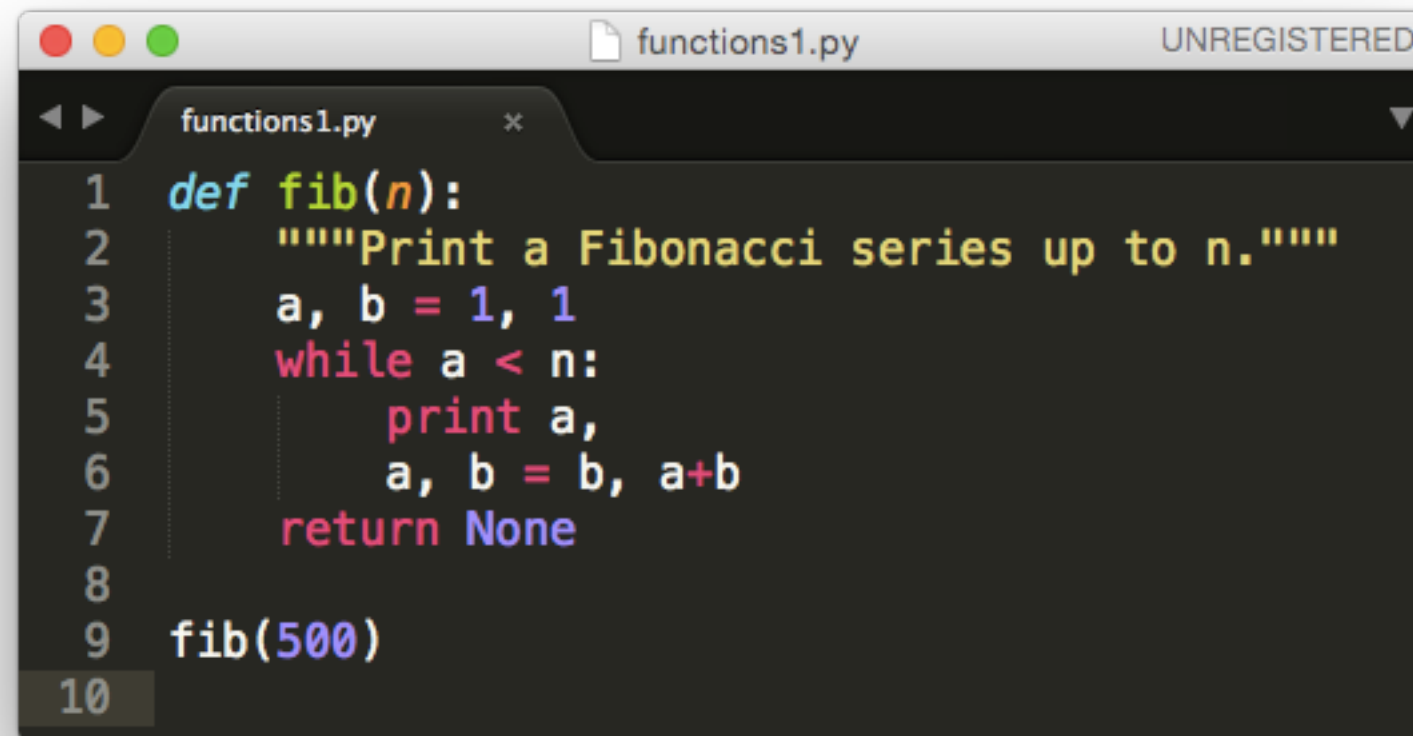
```
if.py
1 age = 22
2
3 if age < 13:
4     print 'kid'
5 elif age < 18:
6     print 'teen'
7 else:
8     print 'adult'
```

```
for.py
1 for i in range(5):
2     pass
3
4 for i in [0, 1, 2, 3, 4]:
5     if i > 5:
6         break
7 else:
8     print 'Did not break'
```

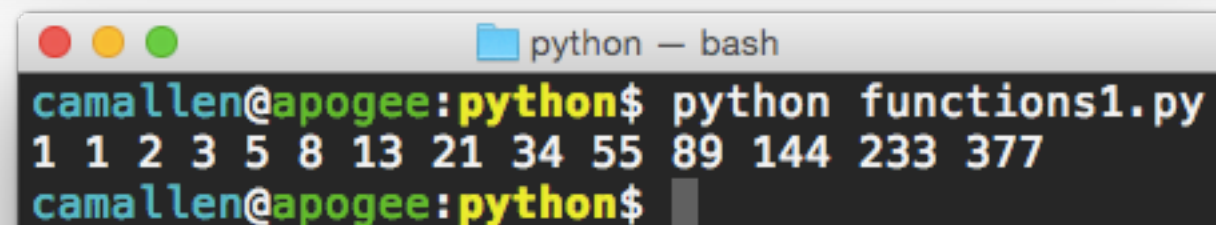
```
while.py
1 x = 1024
2
3 while x > 1:
4     x = x / 2
5     if (x % 10) != 2:
6         continue
7     print x
8
```

```
try.py
1 try:
2     1/0
3 except:
4     print 'Exception!'
5 else:
6     print 'No exception!'
7 finally:
8     print 'Done.'
```


Functions

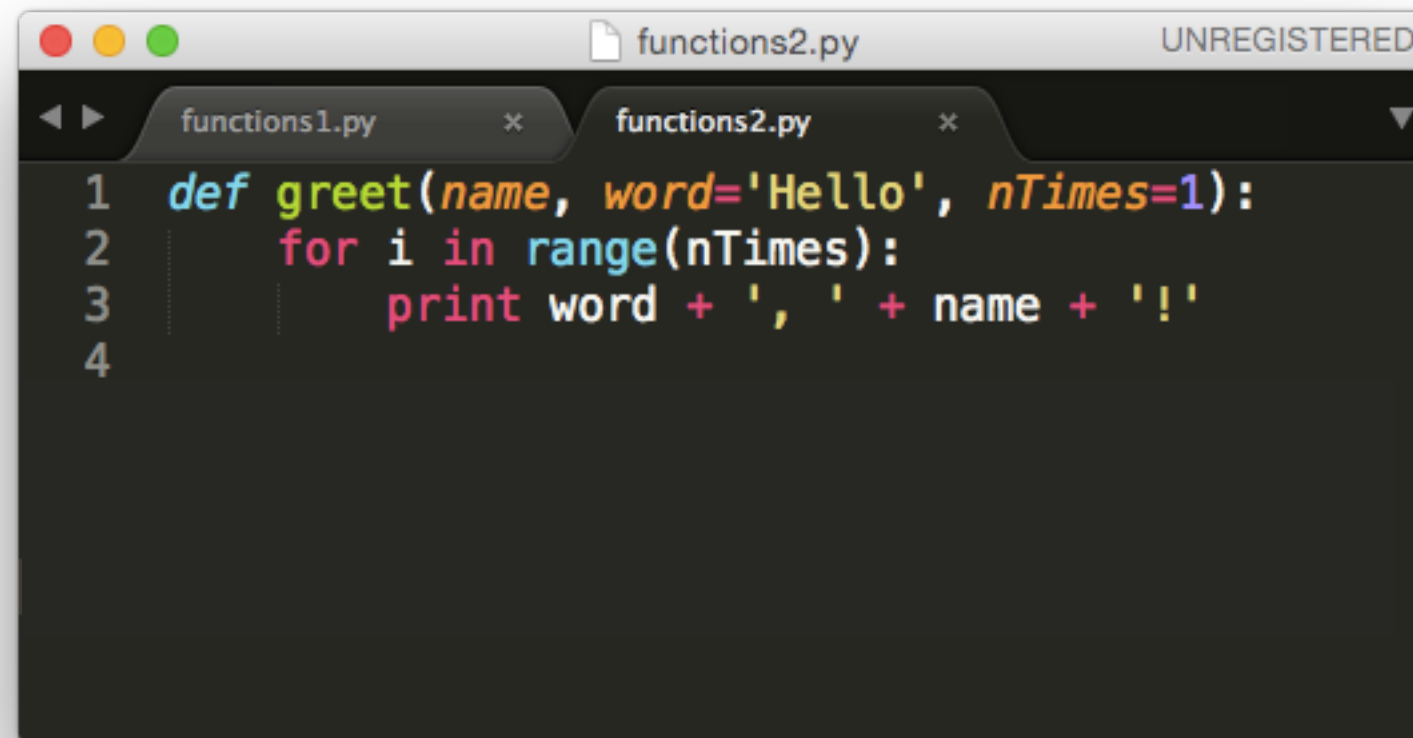


```
1 def fib(n):
2     """Print a Fibonacci series up to n."""
3     a, b = 1, 1
4     while a < n:
5         print a,
6         a, b = b, a+b
7     return None
8
9 fib(500)
10
```



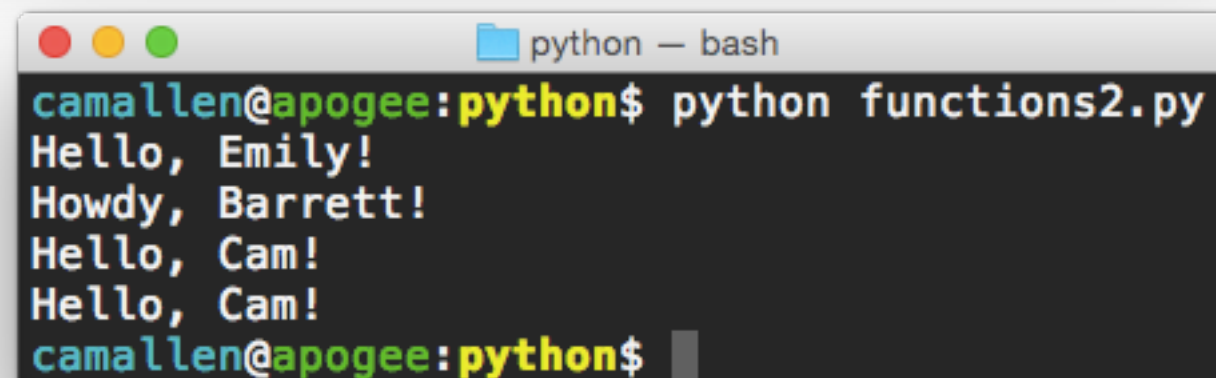
```
python — bash
camallen@apogee:python$ python functions1.py
1 1 2 3 5 8 13 21 34 55 89 144 233 377
camallen@apogee:python$
```

Default Arguments



A screenshot of a code editor window titled 'functions2.py' with a status bar indicating 'UNREGISTERED'. The editor shows two tabs: 'functions1.py' and 'functions2.py'. The code in 'functions2.py' is as follows:

```
1 def greet(name, word='Hello', nTimes=1):
2     for i in range(nTimes):
3         print word + ', ' + name + '!'
4
```



A screenshot of a terminal window titled 'python — bash'. The prompt is 'camallen@apogee:python\$'. The command 'python functions2.py' has been executed, resulting in the following output:

```
camallen@apogee:python$ python functions2.py
Hello, Emily!
Howdy, Barrett!
Hello, Cam!
Hello, Cam!
camallen@apogee:python$
```

Classes

```
classes.py UNREGISTERED
1 import math
2
3 class Vector2:
4     def __init__(self, x, y):
5         self.x = x
6         self.y = y
7
8     def len(self):
9         return math.sqrt(self.x**2 +
10                          self.y**2)
11
12     _DoNotTouch = 10
13
14
15 v = Vector2(3, 4)
16 print "({},{}):".format(v.x, v.y), \
17       "len = {}".format(v.len())
18
```

```
python — bash
camallen@apogee:python$ python classes.py
(3,4): len = 5.0
camallen@apogee:python$
```

Inheritance

```
inheritance.py  UNREGISTERED

1 class shape:
2     def __init__(self, b, h):
3         self.base = b
4         self.height = h
5     def __str__(self):
6         return str( (self.base, self.height) )
7
8 class rectangle(shape):
9     def area(self):
10        return self.base * self.height
11
12 class triangle(shape):
13     def area(self):
14        return self.base * self.height / 2
15
16
17 rect = rectangle(4.0, 3.0)
18 tri = triangle(4.0, 3.0)
19 print " rect: {}    area: {}".format(rect, rect.area())
20 print " tri:  {}    area: {}".format(tri, tri.area())
21
```

```
python — bash
camallen@apogee:python$ python inher
itance.py
rect: (4.0, 3.0)    area: 12.0
tri:  (4.0, 3.0)    area: 6.0
camallen@apogee:python$
```

Importing Modules

```
random_ints.py  UNREGISTERED
1 import random
2
3 for i in range(5):
4     print(random.randint(10,99))
5

random_floats.py
1 import random
2
3 for i in range(5):
4     print(random.randint(10,99)*1.0)
5

modules.py
1 import random_ints
2 import random_floats
3 print ' Done. '
4
```

```
python — bash
camallen@apogee:python$ python modules.py
88
31
29
47
23
89.0
14.0
58.0
92.0
67.0
Done.
camallen@apogee:python$
```

Summary

- Why we're using Python
- How to use Python
- Language basics
- Building blocks

References

- Content is based on slides by Zhenyu Zhou, Richard Guo
- python.org - Official Python website
- [Berkeley Python/UNIX tutorial](#) - Available on course webpage
- learnpython.org - Basic tutorials, examples
- [A Byte of Python](#) - Beginner's tutorial
- [Oliver Fromme](#) - Python Information and Examples
- tiobe.com - Language popularity index