Python

Cam Allen

cam@cs.duke.edu

Based on slides by Zhenyu Zhou, Richard Guo

DUKE
COMPUTER
SCIENCE

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

```
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

random_ints.py UNREGISTERED

random_ints.py

```python
import random

for i in range(5):
    print(random.randint(10,99))

```

python — bash

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

Indentation

indentation.py UNREGISTERED

indentation.py

```python
x = 0

while x < 10:
    if x % 2 == 0:
        print x

    x += 1

print 'done.'
```

python — bash

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

Indentation Errors

indentation.py UNREGISTERED

indentation.py

```
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

dynamic-typing.py UNREGISTERED

dynamic-typing.py

```
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 — Python

```
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

camallen — Python

```
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 — Python

```
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	`s = "Don't touch that dial!"`
List	`L = [1, 2, 3, 4, 5]`
Tuple	`t = ('Check', 1, 2)`
(more)	`...`

Sequence Types

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

Dictionaries

camallen — Python

```
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

while.py UNREGISTERED

if.py

```python
age = 22

if age < 13:
    print 'kid'
elif age < 18:
    print 'teen'
else:
    print 'adult'
```

for.py

```python
for i in range(5):
    pass

for i in [0, 1, 2, 3, 4]:
    if i > 5:
        break
else:
    print 'Did not break'
```

while.py

```python
x = 1024

while x > 1:
    x = x / 2
    if (x % 10) != 2:
        continue
    print x

```

try.py

```python
try:
    1/0
except:
    print 'Exception!'
else:
    print 'No exception!'
finally:
    print 'Done.'
```

Functions

functions1.py UNREGISTERED

functions1.py

```python
def fib(n):
    """Print a Fibonacci series up to n."""
    a, b = 1, 1
    while a < n:
        print a,
        a, b = b, a+b
    return None

fib(500)
```

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

functions2.py UNREGISTERED

functions1.py functions2.py

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

python — bash

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

Classes

classes.py UNREGISTERED

```python
import math

class Vector2:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def len(self):
        return math.sqrt(self.x**2 +
                         self.y**2)

    _DoNotTouch = 10


v = Vector2(3, 4)
print "({},{}):".format(v.x, v.y), \
      "len = {}".format(v.len())
```

python — bash

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

Inheritance

inheritance.py UNREGISTERED

inheritance.py

```python
class shape:
    def __init__(self, b, h):
        self.base = b
        self.height = h
    def __str__(self):
        return str( (self.base, self.height) )

class rectangle(shape):
    def area(self):
        return self.base * self.height

class triangle(shape):
    def area(self):
        return self.base * self.height / 2


rect = rectangle(4.0, 3.0)
tri  = triangle(4.0, 3.0)
print " rect: {}   area: {}".format(rect, rect.area())
print " tri:  {}   area: {}".format(tri, tri.area())
```

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

random_ints.py

```python
import random

for i in range(5):
    print(random.randint(10,99))

```

random_floats.py

```python
import random

for i in range(5):
    print(random.randint(10,99)*1.0)

```

modules.py

```python
import random_ints
import random_floats
print ' Done.'

```

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