PYCADEMY

Mater says....

- Python enabled us to create EVE Online, a massive multiplayer game, in record time. The EVE Online server cluster runs over 25,000 simultaneous players in a shared space simulation, most of which created in Python. (Hilmar Veigar)

Mater says….

- Python … is compact you can hold its entire feature set in your head (Eric S. Raymond)

- Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we are looking for more people with skills in this language. (Peter Norvig)

Mater says….

- NASA is using Python to implement a CAD/CAE/PDM repository and model management, integration, and transformation system which will be the core infrastructure for its next generation collaborative engineering environment…(Steve Waterbury)

Mater says….

- Python enabled us to create EVE Online, a massive multiplayer game, in record time. The EVE Online server cluster runs over 25,000 simultaneous players in a shared space simulation, most of which created in Python. (Hilmar Veigar)

What is Python

- Developed and supported by a large team of volunteers – Python Software Foundation

- Major implementations: Cpython, Jython, Iron Python, PyPy
  - Cpython – implemented in C, the primary implementation
  - Jython – implementation for JVM
  - Pypy – implementation in Python
  - IronPython – implemented in C#, allows python to use the .NET libraries

Batteries Included

- The python standard library is very extensive

  Regular expressions, codecs, data time, collections,
  theads and mutexs, OS and shell level functions,
  Support for SQLite and Berkley databases,
  Zlib, gzio, bz2, tarfile, csv, xml, md5, sha, logging, email,
  Json, httlip, imaplib, nntplib, smtplib,
  And much, much more ….

Python Libraries

- Biopython – Bioinformatics
- SciPy – Linear algebra, signal processing
- NumPy – Fast compact multidimensional arrays
- PyGame – Game Development
- Visul Python – real time 3D output
- Django – High level python web framework

And much more

HELLO WORLD

```
print "hello world"
```

Python Libraries

- Biopython – Bioinformatics
- SciPy – Linear algebra, signal processing
- NumPy – Fast compact multidimensional arrays
- PyGame – Game Development
- Visul Python – real time 3D output
- Django – High level python web framework

And much more

»» 2 + 2

Variables

```
a = 4           # Integer
b = 5.6         # Float
c = "hello"     # String
a = "4"     # rebound to String
```

Math

Math

+, **-**, *****, **/**, ****** (power), **%** (modulo)

Careful with integer division

```
>>> 3/4
0
>>> 3/4.
0.75
```

(In Python 3 // is integer division operator)

What happens when you raise 10 to the 100th?

Long

```
>>> 10**100
10000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000L
```

Long (2)

```
>>> import sys
>>> sys.maxint
9223372036854775807
>>> sys.maxint + 1
9223372036854775808L
```

Strings

STRINGS

```
name = 'matt'
with_quote = "I ain't gonna"
longer = """This string has
multiple lines
in it"""
```

String methods

- `s.endswith(sub)`

  Returns True if endswith `sub`

- `s.find(sub)`

  Returns index of `sub` or `-1`

- `s.format(*args)`

  Places args in string

String methods (2)

- `s.index(sub)`

  Returns index of `sub` or exception

- `s.join(list)`

  Returns `list` items separated by string

- `s.strip()`

  Removes whitespace from start/end

Comments

COMMENTS

Comments follow a `#`

COMMENTS

No multi-line comments

More Types

None

Pythonic way of saying `NULL`. Evaluates to False.

```
c = None
```

BOOLEANS

```
a = True
b = False
```

SEQUENCES

- *lists*
- *tuples*
- *sets*

LISTS

Hold sequences.

How would we find out the attributes & methods of a list?

LISTS

```
>>> dir([])
['__add__', '__class__', '__contains__',...
'__iter__',... '__len__',... , 'append', 'count',
'extend', 'index', 'insert', 'pop', 'remove',
'reverse', 'sort']
```

LISTS

```
>>> a = []
>>> a.append(4)
>>> a.append('hello')
>>> a.append(1)
>>> a.sort() # in place
>>> print a
[1, 4, 'hello']
```

LISTS

How would we find out documentation for a method?

LISTS

`help` function:

```
>>> help([].append)
Help on built-in function append:

append(...)
    L.append(object) -- append object to end
```

List methods

- **`l.append(x)`**

  Insert `x` at end of list

- **`l.extend(l2)`**

  Add `l2` items to list

- **`l.sort()`**

  In place sort

List methods (2)

- **`l.reverse()`**

  Reverse list in place
- **`l.remove(item)`**

  Remove first `item` found
- **`l.pop()`**

  Remove/return item at end of list

Dictionaries

DICTIONARIES

Also called *hashmap* or *associative array* elsewhere

```
>>> age = {}
>>> age['george'] = 10
>>> age['fred'] = 12
>>> age['henry'] = 10
>>> print age['george']
10
```

DICTIONARIES

```
>>> age.keys()
#['george', 'fred', 'henry']

>>> age.values()
#['10', '12', '10']

>>> age.items()
#[['george', 10], ['fred', 12], ['henry', 10]]
```

DICTIONARIES (2)

Find out if `'matt'` in `age`

```
>>> 'matt' in age
False
```

DELETING KEYS

Removing `'charles'` from `age`

```
>>> del age['charles']
```

Functions

FUNCTIONS

```python
def add_2(num):
    """ return 2
    more than num
    """

    return num + 2

five = add_2(3)
```

WHITESPACE

Instead of `{` use a `:` and indent consistently (4 spaces)

DEFAULT (NAMED) PARAMETERS

```
def add_n(num, n=3):
    """default to
    adding 3"""
    return num + n

five = add_n(2)
ten = add_n(15, -5)
```

Conditionals

CONDITIONALS

```
if grade > 90:
    print "A"
elif grade > 80:
    print "B"
elif grade > 70:
    print "C"
else:
    print "D"
```

Remember the colon/whitespace!

Booleans

```
a = True
b = False
```

Comparison Operators

Supports (`>`, `>=`, `<`, `<=`, `==`, `!=`)

```
>>> 5 > 9
False
>>> 'matt' != 'fred'
True
>>> isinstance('matt',
basestring)
True
```

Boolean Operators

`and`, `or`, `not` (for logical), `&`, `|`, and `^` (for bitwise)

```
>>> x = 5
>>> x < -4 or x > 4
True
```

Boolean note

Parens are only required for precedence

```
if (x > 10):
    print "Big"
```

same as

```
if x > 10:
    print "Big"
```

Chained comparisons

```
if 3 < x < 5:
    print "Four!"
```

Same as

```
if x > 3 and x < 5:
    print "Four!"
```

Iteration

ITERATION

```
for number in [1,2,3,4,5,6]:
    print number

for number in range(1, 7):
    print number
```

`range` NOTE

Python tends to follow *half-open interval* (`[start,end)`) with `range` and slices.

- end - start = length
- easy to concat ranges w/o overlap

ITERATION (5)

Can `continue` to skip over items

```
for item in sequence:
    if item < 0:
        continue
    # process all positive items
```

pass

`pass` is a null operation

```
for i in range(10):
    # do nothing 10 times
    pass
```

Slicing

Slicing

Sequences (lists, tuples, strings, etc) can be *sliced* to pull out a single item

```
my_pets = ["dog", "cat", "bird"]
favorite = my_pets[0]
bird = my_pets[-1]
```

Negative Indexing

Proper way to think of [negative indexing] is to reinterpret `a[-X]` as `a[len(a)-X]`

@gvanrossum

Slicing (2)

Slices can take an end index, to pull out a list of items

```
my_pets = ["dog", "cat", "bird"]
 # a list
cat_and_dog = my_pets[0:2]
cat_and_dog2 = my_pets[:2]
cat_and_bird = my_pets[1:3]
cat_and_bird2 = my_pets[1:]
```

Slicing (3)

Slices can take a stride

```
my_pets = ["dog", "cat", "bird"]
 # a list
dog_and_bird = [0:3:2]
zero_three_etc = range(0,10)
[::3]
```

Slicing (4)

Just to beat it in

```
veg = "tomatoe"
correct = veg[:-1]
tmte = veg[::2]
eotamot = veg[::-1]
```

File IO

File Input

Open a file to read from it (old style):

```
fin = open("foo.txt")
for line in fin:
    # manipulate line

fin.close()
```

File Output

Open a file using `'w'` to `write` to a file:

```
fout = open("bar.txt", "w")
fout.write("hello world")
fout.close()
```

Always remember to close your files!

CLOSING WITH `with`

implicit `close` (new 2.5+ style)

```
with open('bar.txt') as fin:
    for line in fin:
        # process line
```