

A Caffeinated Crash Course in Python

Python is not....

- Java
- C
- Perl

The Python Interpreter

- Type “python” at the command prompt
- In windows, find the python icon on the start menu

```
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Dir and Help

```
Blicket:~ havasi$ python
Python 2.3.5 (#1, Aug 19 2006, 21:31:42)
[GCC 4.0.1 (Apple Computer, Inc. build 5363)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> help(math)

>>> dir(math)
['__doc__', '__file__', '__name__', 'acos', 'asin', 'atan', 'atan2', 'ceil', 'cos', 'cosh', 'degrees', 'e', 'exp', 'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp', 'log', 'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh']
>>> █
```

help()

dir()

Syntax Errors

- Python Errors show the line number of the error
- Check the line above if your error makes no sense

```
>>> 1 +  
Traceback (most recent call last):  
  File "<stdin>", line 1  
    1 +  
      ^  
SyntaxError: invalid syntax  
>>>
```

White Space

String Basics

- Not a mutable data type

```
>>> 'Hello World'  
'Hello World'  
>>> "Hello World"  
'Hello World'  
>>>
```

- String can be delimited with either the “
or ‘

More Strings

- Concatenation uses the +

```
>>> 'Hello' + 'World'
'HelloWorld'
>>>
```

- You can do math with strings!

```
>>> 'Hi' + 'Hi' + 'Hi'
'HiHiHi'
>>> 'Hi' * 3
'HiHiHi'
>>>
```


Output

```
Blicket:~ havasi$ python
Python 2.3.5 (#1, Aug 19 2006, 21:31:42)
[GCC 4.0.1 (Apple Computer, Inc. build 5363)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = "I'm sorry Dave"
>>> b = "I can't do that"
>>> print a
I'm sorry Dave
>>> print b
I can't do that
>>> print a, b
I'm sorry Dave I can't do that
>>> █
```

Indexing

- To index into a string, specify the position inside square brackets

```
>>> msg = 'Hello World'
>>> msg[0]
'H'
>>> msg[3]
'l'
>>> msg[5]
'.'
>>>
```

- You can index into a string from the “end” of the string.

```
>>> msg[-3]
'r'
>>> msg[-6]
'.'
>>>
```

Slicing

- A Substring of a string is a slice

```
>>> msg[1:4]
'ell'
>>>
```

- Your head or tail can be a negative index

```
>>> msg[0:-6]
'Hello'
>>>
```

More Slicing

- You don't need to specify the beginning and end of the string

```
>>> msg[:3]
'Hel'
>>> msg[6:]
'World'
>>>
```

- Find the length of a string with len()

```
>>> len(msg)
11
>>> msg[0:11]
'Hello World'
>>>
```

Example

-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	
H	e	l	l	o		W	o	r	l	d	
0	1	2	3	4	5	6	7	8	9	10	11

Lists

- Lists in python are made of any data type delimited by commas and surrounded by brackets.

```
>>> cgi = ['colorless', 'green', 'ideas']  
>>> cgi  
['colorless', 'green', 'ideas']  
>>>
```

- Lists are mutable

More on Lists

- You can index into lists

```
>>> len(cgi)
3
>>> cgi[0]
'colorless'
>>> cgi[-1]
'ideas'
```

- You can slice lists

```
>>> cgi[1:3]
['green', 'ideas']
>>> cgi[-2:]
['green', 'ideas']
>>>
```

Modifying Lists

- You can add lists

```
>>> chomsky = ['curious', 'green', 'ideas']
>>> chomsky = chomsky + ['sleep', 'furiously']
>>> chomsky
['curious', 'green', 'ideas', 'sleep', 'furiously']
>>> █
```

- And append to them

```
>>> chomsky.append('said')
>>> chomsky.append('Chomsky')
>>> chomsky
['sleep', 'ideas', 'green', 'furiously', 'colorless', 'said', 'Chomsky']
>>> chomsky.index('green')
2
>>>
```


List Methods

```
>>> chomsky = ['curious', 'green', 'ideas', 'sleep', 'furiously']
>>> alpha = sorted(chomsky)
>>> alpha
['curious', 'furiously', 'green', 'ideas', 'sleep']
>>> chomsky.sort()
>>> chomsky
['curious', 'furiously', 'green', 'ideas', 'sleep']
>>> chomsky.reverse()
>>> chomsky
['sleep', 'ideas', 'green', 'furiously', 'curious']
>>> █
```

- `sort` - sorts the list in place, returns nothing
- `sorted` - does not modify the list, returns new sorted list
- `reverse` - reverses the list in place, returns nothing

String Formatting

```
>>> for word in chomsky:  
...     print "%s (%d)," % (word, len(word)),  
colorless (9), green (5), ideas (5), sleep (5), furiously (9),
```

- The % operator substitutes values into a string
- %s and %d are placeholders for the values (%d makes sure it's a number)
- “%s has %d letters” % (“colorless”, len(“colorless”)) becomes the string “colorless has 9 letters”

Converting from Strings to Lists

- Join a list to make a string

```
>>> sent = ' '.join(chomsky)
>>> sent
'colorless green ideas sleep furiously'
>>>
```

- Split a string to make a list

```
>>> sent.split(' ')
['colorless', 'green', 'ideas', 'sleep', 'furiously']
>>>
```

For and If

- If statements

```
>>> word = "cat"
>>> if len(word) < 5:
...     print 'word length is less than 5'
...
word length is less than 5
>>>
```

- For Statements

```
>>> for num in [1, 2, 3]:
...     print 'The number is', num
...
The number is 1
The number is 2
The number is 3
```

List Comprehensions

- Applies a function to every element of a list

```
>>> x = [6, 7, 8, 0, 2, 1]
>>> [item for item in x]
[6, 7, 8, 0, 2, 1]
>>> [item*2 for item in x]
[12, 14, 16, 0, 4, 2]
>>> [item for item in x if item > 5]
[6, 7, 8]
>>> █
```

Dictionaries

- Hash - maps things to things!

Phone List

Alex	x154
Dana	x642
Kim	x911
Les	x120
Sandy	x124

Domain Name Resolution

aclweb.org	128.231.23.4
amazon.com	12.118.92.43
google.com	28.31.23.124
pythonb.org	18.21.3.144
sourceforge.net	51.98.23.53

Word Frequency Table

computational	25
language	196
linguistics	17
natural	56
processing	57

Even More Dictionaries

```
>>> pos = {'furiously': 'adv', 'ideas': 'n', 'colorless': 'adj'}  
>>>
```

```
>>> pos.keys()  
['colorless', 'furiously', 'ideas']  
>>> pos.values()  
['adj', 'adv', 'n']  
>>> pos.items()  
[('colorless', 'adj'), ('furiously', 'adv'), ('ideas', 'n')]  
>>> for (key, val) in pos.items():  
...     print "%s ==> %s" % (key, val)  
...  
colorless ==> adj  
furiously ==> adv  
ideas ==> n  
>>>
```

Example: Letter Frequencies

```
>>> phrase = 'colorless green ideas sleep furiously'
>>> count = {}
>>> for letter in phrase:
...     if letter not in count:
...         count[letter] = 0
...         count[letter] += 1
>>> count
{'a': 1, ' ': 4, 'c': 1, 'e': 6, 'd': 1, 'g': 1, 'f': 1, 'i': 2,
 'l': 4, 'o': 3, 'n': 1, 'p': 1, 's': 5, 'r': 3, 'u': 2, 'y': 1}
>>>
```


Classes

```
from math import sqrt

class Point(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.label = None

    def distanceOrigin(self):
        return sqrt(self.x^2 + self.y^2)

    def setLabel(self, labelString):
        self.label = labelString

    def __repr__(self):
        if self.label is not None: return self.label
        return '(' + str(self.x) + ', ' + str(self.y) + ')'
```

Importing and the Python path

- Import using the import command
- You can import everything from a module using the syntax “from <module> import *”

```
>>> import math
>>> math.sqrt(4)
2.0
>>> from math import sqrt
>>> sqrt(4)
2.0
>>> █
```

Files

```
Filename = "/home/havasi/input.txt"  
input = open(Filename, 'r')  
output = open(Filename + '.out', 'w')  
for line in input.readlines():  
    input.write('Cows! \n')  
input.close()  
output.close()
```

Resources

- Python.org
- NLTK Python Tutorial
 - <http://nltk.org/doc/en/programming.html>
- IDLE (Windows Development Env.)
 - <http://www.python.org/idle/>