A Quick Python Cheat Sheet

February 19, 2013

1 Getting Help

```
# Get help for a module, function, class, etc.
help('math')
help('str')

# Get a brief listing of names in namespace.
import math
dir(math)
dir(str)
```

2 Strings

```
# Creation.
s = "Hello world!"
s = 'Hello world!'
s = 'Hello' + ' ' + 'world!'

# Access.
print s[0] # returns 'H'
print s[1] # returns 'e'

# Operations.
s.split() # returns ['Hello', 'world!']
s = ' '.join(['Hello\\n', 'world\\n']) # s becomes 'Hello\\nworld\\n'
s.splitlines() # returns ['Hello', 'world']
s.strip('.,') # removes periods and commas from string
```

3 Lists

```
# Creating.
l = []
l = [1, 'two', 3]
l = list([1, 'two', 3]) # copy a list

# Modifying lists.
```

```
l.append(4)
l.extend([5, 6, 7])
l.reverse()
l.sort()

# Querying lists.
4 in l
l.count(1) # how many 1's in the list?

# Iteration.
for item in l:
  ...
```

4 Dictionaries

```
# Creating dictionaries.
d = {}
d = {1: 'one', 2: 'two'}
d = dict([(1, 'one'), (2, 'two'), (3, 'three')])
d = dict(zip([1, 2, 3], ['one', 'two', 'three']))

# Modifying dictionaries.
d[5] = 'five'
d.pop(1) # removes key/value, returns value

# Querying dictionaries.
1 in d
d.keys() # returns list of keys
d.values() # returns list of values
d.items() # returns list of key/value tuples.

# Iteration.
for key in d.iterkeys(): # iterate over keys
  ...
for value in d.itervalues(): # iterate over values
  ...
for key, value in d.iteritems(): # iterate over keys, values
  ...
```

5 Files

```
# Open a text file for reading.
f = open('input.txt')

# Open a text file for writing.
f = open('output.txt', 'w')

# Close a file.
f.close()
```

```
# Safely open a text file for reading.
with open('input.txt') as f:
  # Do something here...
  # File is automatically closed when we
  # exit from the 'with' block.

# Iterate over lines in a file.
with open('input.txt') as f:
  for line in f:
    # Do something here...
```

6 Functions

```
# Declare a function.
def func():
  # Do something here...

# Accept arguments.
def magnitude(x, y, z):
  mag = sqrt(x*x + y*y + z*z)

# Return values.
def magnitude(x, y, z):
  mag = sqrt(x*x + y*y + z*z)
  return mag

# Calling functions.
value = magnitude(3, 4, 5)
```

7 List Comprehensions Basics

```
a_list = [1, 2, 3, 4]

# Square each element of a list.
a_list = [l*l for l in a_list]

# Filter a list.
a_list = [l for l in a_list if l < 3]

# Filter and square each element in a list.
a_list = [l*l for l in a_list if l < 3]
```

8 Modules

```
# Import a module and access contents.
import math
dir(math)
```

```
math.cos(0.3)
math.sin(0.3)

# Import all module contents directly.
from math import *
cos(0.3)
sin(0.3)

# Import one symbol from module.
from math import cos
cos(0.3)
sin(0.4) # error!
```

9 Object Oriented Basics

```
# Define an empty class.
class MyClass(object):
  pass

# Define a constructor.
class MyClass(object):

  def __init__(self):
    self.some_int = 10
    self.my_name = 'MyClass'

# Define methods.
class MyClass(object):

  def __init__(self):
    self.some_int = 10

  def get_some_int_squared(self):
    return self.some_int*self.some_int
```

10 More Object Oriented Programming

```
# Inheritance.
class MyClass(object):
  def __init__(self):
    pass

class MySubClass(MyClass):
  def __init__(self):
    super(MySubClass, self).__init__(self)

# Operator overloading.
class MyClass:
```

```
  # Equal (==).
  def __eq__(self, value):
     pass

  # Not equal (!=).
  def __ne__(self, value):
     pass

  # Less than (<).
  def __lt__(self, value):
     pass

  # Less than or equal (<=).
  def __le__(self, value):
     pass

  # Greater than (>).
  def __gt__(self, value):
     pass

  # Greater than or equal (>=).
  def __ge__(self, value):
     pass

# Basic introspection.
class MyClass(object):
  def __init__(self):
    self.some_int = 10

my_class = MyClass()
attrs = vars(my_class) # get the attributes dictionary

getattr(my_class, "some_int")
getattr(my_class, "not_there") # error!
setattr(my_class, "another_int")
```