String Operations

Useful String functions:

Make lowercase:	"A".lower()="a"
Make UPPERCASE :	"a".upper()="A"
Make Title Format:	"hi world".title()="Hi World"
Replace a substring:	"123".replace('2','z')= "1z3"
Count occurrences of substring:	"1123".count("1")=2
Get offset of substring in string:	"123".index("2")=1
Detect substring in string:	"is" in "fish" == True
Encode "a string":	"a string".encode(*codec name*)
Decode a string:	"a string".decode(*codec name*)

Example with the ROT13 codec:

```
>>> print "RAPBQR-ZR".decode("rot13")
ENCODE-ME
```

Some String encoders/decoder codec names:
Base64, bz2 , hex, rot13, uu, zip, string_escape

Convert a string to a list (default separator=space):

```
newlist = astr.split(<separator>)
>>> print "A B C".split()
['A', 'B', 'C']
>>> print "A B      C".split()
['A', 'B', 'C']
>>> print "A,B,  ,C".split(",")
['A', 'B', '  ', 'C']
>>> print "WANNA BANANA?".split("AN")
['W', 'NA B', '', 'A?']
```

Convert a list (or other iterable object) to a string:
Join a list together putting string between elements.
"astring".join([list])

```
>>> print "".join(['A','B','C'])
ABC
>>> print ",".join(['A','B','C'])
A,B,C
```

Converting Data Types

Convert anything to a string:

```
newstring = str(<any variable>)
newstring = str(100)
```

Convert String to Integer:

```
newint = int(<string>[,base])
All of the following assign the
variable ten the integer 10
>>> ten = int("1010",2)
>>> ten = int("0010")
>>> ten = int("000A",16)
```

Convert Float to Integer by dropping decimal:

```
newint = int(<float>)
>>> print int(3.1415)
3
>>> int(3.6)
3
```

Convert String or number to Float:

```
afloat = float(<var>)
>>> print float("1.5")
1.5
>>> print float(1)
1.0
```

Convert String Character to ASCII decimal:

```
newint = ord(<string length 1>)
>>> print ord("A")
65
```

Convert ASCII decimal to String of length 1:

```
newstr = chr(<integer 1 to 255>)
>>> print chr(65)
A
```

SANS INSTITUTE

Python 2.7 Essentials

~~POCKET REFERENCE GUIDE~~
SANS Institute

www.sans.org/sec573
http://isc.sans.edu
Mark Baggett Twitter: @markbaggett

3 Methods of Python Execution

Command line Execution with -c:

```
# python –c ["script string"]
python –c "print 'Hello World!'"
```

Python Interpreter Script Execution:

```
# cat helloworld.py
print "Hello World"
# python helloworld.py
Hello World
```

Python Interactive Shell:

```
# python
>>> print "Hello World"
Hello World
```

Python Command Line Options

```
# python –c "script as string"
```
Execute a string containing a script
```
# python –m <module> [module args]
```
Find module in path and execute as a script
Example: python –m "SimpleHTTPServer"
```
# python –i <python script>
```
Drop to interactive shell after script execution

Loops Lists & Dictionaries

List essentials:

Create an empty list:	newlist=[]
Assign value at index:	alist[index]= value
Access value at index	alist[index]
Add item to list:	alist.append(new item)
Insert into list:	alist.insert(at position, new item)
Count # of an item in list:	alist.count(item)
Delete 1 matching item:	alist.remove(del item)
Remove item at index	del alist[index]

Dictionary essentials:

Create an empty dict:	dic={}
Initialize a non-empty dictionary:	dic = { "key1":"value1","key2":"value2"}
Assign a value:	dic["key"]="value"
Determine if key exists:	dic.has_key("key")
Access value at key:	dic["key"], dic.get("key")
List of all keys:	dic.keys()
List of all values:	dic.values()
List of (key,value) tuples:	dic.items()

Looping examples:

For loop 0 thru 9:	for x in range(10):
For loop 5 thru 10:	for x in range(5,11):
For each char in a string:	for char in astring:
For items in list :	for x in alist:
For loop retrieving indexes and values in a list :	for index,value in enumerate(alist):
For keys in a dict :	for x in adict.keys():
For all items in dict:	for key,value in adict.items():
while <logic test> do:	

Loop Control statements (for and while):

Exit loop immediately	break
Skip rest of loop and do loop again	continue

Misc

Adding Comments to code:

```
#Comments begin the line with a pound sign
```

Defining Functions:

Here is a function called "add". It accepts 2 arguments num1 and num2 and returns their sum. Calling "`print add(5,5)`" will print "10" to the screen:

```
def add(num1, num2):
    #code blocks must be indented
    #each space has meaning in python
    myresult = num1 + num2
    return myresult
```

if then else statements:

```
if <logic test 1>:
    #code block here will execute
    #when logic test 1 is True
elif <logic test 2>:
    #code block executes logic test 1 is
    #False and logic test 2 is True
else:
    #code block for else has no test and
    #executes when if an all elif are False
```

Slicing and Indexing Strings, Lists, etc

Slicing strings and lists:

x[start:stop:step]	x=[4,8,9,3,0]	x="48930"
x[0]	4	'4'
x[2]	9	'9'
x[:3]	[4,8,9]	'489'
x[3:]	[3,0]	'30'
x[:-2]	[4,8,9]	'489'
x[::2]	[4,9,0]	'490'
x[::-1]	[0,3,9,8,4]	'03984'
len(x)	5	5
sorted(x)	[0,3,4,8,9]	['0', '3', '4', '8', '9']

SEC573 PyWars Essentials

Create pyWars Object

```
>>> import pyWars
>>> game= pyWars.exercise()
```

Account Mangement

```
>>> game.new_acct("username","password")
>>> game.login("username","password")
>>> game.logout()
```

Query a question:

```
>>> game.question(<question #>)
```

Query the data:

```
>>> game.data(<question #>)
```

Submit an answer:

```
>>> game.answer(<question #>,
   solverfunc(game.data(<question#>)))
```

Logic and Math Operators

Math Operator	**Example**	**X=7, Y=5**
Addition	X + Y	12
Subtraction	X - Y	2
Multiplication	X * Y	35
Division	X / Y	1
Exponent	X ** Y	16807
Modulo	X % Y	2
Logic Operator		
Equality	X == Y	False
Greater Than	X > Y	False
Less Than	X < Y	True
Less or Equal	X <= Y	True
Not Equal	X !=Y or X<>Y	True
Other Logical Operators: AND, OR and NOT		