String Operations

Types of String functions:

Default Python3 Unicode (UTF8 default) strings:

```
newstr = "Create Unicode String"
newstr = b"Python2 Str".decode()
```

bytes (Python2 like strings):

```
newstr = b"Create byte str"
newstr = "UTF8 Str".encode()
```

bytearrays strings (mutable Python2 like strings):

```
newstr = bytearray(data,encoding)
```

String prefixes :

Bytes - b before quotes create a string of bytes:

```
newstr = b"Python2 like string"
```

Raw - r before quotes auto-escape \ characters:

```
newstr = r"\x\x\x"
```

Format - f before quotes is 3.6+ format str:

```
newstr = f"Python {variable}"
```

Useful string, bytes, bytearray methods & functions : (strings shown)

Make lowercase: "Ab".lower()="ab"
Make UPPERCASE : "Ab".upper()="AB"
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
Convert to a list :(default separator is whitespace):
newlist="astr".split(separator [,max])

```
>>> "A,B,C".split(",")
['A', 'B', 'C']
>>> "A,B,C".split(",",1)
['A', 'B,C']
```

Convert list to a string: "astring".join([list])

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

Converting Data Types

Various functions and methods exist to convert from one type of data to another type. Here are some commonly used conversions.

Convert	Syntax	Example	Result
Number to string	**str(number)** *int, float or long*	str(100) str(3.14)	'100' '3.14'
Encoded bytes to string	**str(txt,encoding)** *txt from files, web,sockets, etc*	str(data,"utf8")	string with data
String of numbers to int	**int("string",base)** *default base is 10*	int("42") int("101",2) int("ff", 16)	42 5 255
integer to hex string	**hex(integer)**	hex(255) hex(10)	'0xff' '0xa'
integer to binary string	**bin(integer)**	bin(5) bin(3)	'0b101' '0b11'
float to integer	**int(float)** *drops decimal*	int(3.14159) int(3.9)	3 3
int or str to float	**float(int or str)**	float("3.4") float(3)	3.4 3.0
String len 1 to ASCII	**ord(str len 1)**	ord("A") ord("1")	65 49
Integer to ASCII	**chr(integer)**	chr(65) chr(49)	'A' '1'
bytes to string	**<bytes>.decode()**	b'ABC'.decode()	'ABC'
string to bytes	**<str>.encode ()**	'abc'.encode()	b'abc'

Python 3 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 = { "key":"value","key2":"value2"}
Assign a value:	dic["key"]="value"
Determine if key exists:	"key" in dic
Access value at key:	dic["key"], dic.get("key")
Iterable View of all keys:	dic.keys()
Iterable View of all values:	dic.values()
Iterable View 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 each key 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 if logic test 1 is
    #False and logic test 2 is True
else:
    #else has no test and executes when if
    #and 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	1.4
Floor	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		