### Converting Data Types

#### Convert anything to a string:

```python
eString = str(<any variable>)
eString = str(100)
```

#### Convert String to Integer:

```python
newInt = int(<string>[,base])
```

Example:

```
>>> ten = int("1010",2)
>>> ten = int("0010")
>>> ten = int("000A",16)
```

#### Convert Float to Integer by dropping decimal:

```python
newInt = int(<float>)
```

Instantiating:

```
>>> print int(3.1415)
3
>>> print int(3.6)
3
```  

#### Convert String Character to ASCII decimal:

```python
newInt = ord(<string length 1>)
```

Instantiating:

```
>>> print ord("A")
65
```  

#### Convert ASCII decimal to String of length 1:

```python
newStr = chr(<integer 1 to 255>)
```

Instantiating:

```
>>> print chr(65)
A
```  

### String Operations

#### Useful String functions:

- **Make lowercase:**
  ```python
  "A".lower()="a"
  ```

- **Make UPPERCASE:**
  ```python
  "a".upper()="A"
  ```

- **Make Title Format:**
  ```python
  "hi world".title()="Hi World"
  ```

- **Replace a substring:**
  ```python
  "123".replace(2,'z')="1z3"
  ```

#### Count occurrences of substring:

```python
"1123".count(1)=2
```

#### Get offset of substring in string:

```python
"123".index("2")=1
```

#### Detect substring in string:

```python
"is" in "fish" ===> True
```

#### Example with the ROT13 codec:

```python
>>> 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):

```python
newList = aString.split(<separator>)
```

Instantiating:

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

#### Convert a list (or other iterable object) to a string:

```python
"astring".join(List)
```

Instantiating:

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

### 3 Methods of Python Execution

- **Command line Execution with -c:**
  ```bash
  # python –c "["script string"]
  python –c "print 'Hello World!'"
  ```

- **Python Interpreter Script Execution:**
  ```bash
  # cat helloworld.py
  print "Hello World"
  # python helloworld.py
  Hello World
  ```

- **Python Interactive Shell:**
  ```bash
  # python
  >>> print "Hello World"
  Hello World
  ```

### Python Command Line Options

- **Execute a string containing a script:**
  ```bash
  # python –m <module> [module args]
  ```

- **Find module in path and execute as a script:**
  Example:
  ```bash
  python –m "SimpleHTTPServer"
  ```

- **Drop to interactive shell after script execution:**
  ```bash
  # python –i <python script>
  ```
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 dict.keys():`
  - For all items in dict: `for key,value in dict.items():`
  - while <logic test> do:

**Loop Control statements (for and while):**
- Exit loop immediately: `break`
- Skip rest of loop and do loop again: `continue`

---

**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:

```python
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

**Misc**

**Slicing and Indexing Strings, Lists, etc**

<table>
<thead>
<tr>
<th>Slicing strings and lists:</th>
<th>x=start:stop:step</th>
<th>x=[4,8,9,3,0]</th>
<th>x=&quot;48930&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>x[0]</td>
<td>4</td>
<td>'4'</td>
<td></td>
</tr>
<tr>
<td>x[2]</td>
<td>9</td>
<td>'9'</td>
<td></td>
</tr>
<tr>
<td>x[3]</td>
<td>[4,8,9]</td>
<td>'489'</td>
<td></td>
</tr>
<tr>
<td>x[3:]</td>
<td>[3,0]</td>
<td>'30'</td>
<td></td>
</tr>
<tr>
<td>x[::-2]</td>
<td>[4,8,9]</td>
<td>'489'</td>
<td></td>
</tr>
<tr>
<td>x[:2]</td>
<td>[4,9]</td>
<td>'490'</td>
<td></td>
</tr>
<tr>
<td>x[::-1]</td>
<td>[0,3,9,8,4]</td>
<td>'03984'</td>
<td></td>
</tr>
<tr>
<td>len(x)</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>sorted(x)</td>
<td>[0,3,4,8,9]</td>
<td>['0', '3', '4', '8', '9']</td>
<td></td>
</tr>
</tbody>
</table>

---

**SEC573 PyWars Essentials**

**Create pyWars Object**
>>> import pyWars
>>> game= pyWars.exercise()

**Change Scoring Server IP**
>>> game.serverip="127.0.0.1"

**Register a Team**
>>> game.register("team","password")

**Query a question:**
>>> game.question(<question #>)

**Query the data:**
>>> game.data(<question #>)

**Submit an answer:**
>>> game.answer(<question #>, solverfunc(game.data(<question#>)))

---

**Math Operator**

<table>
<thead>
<tr>
<th>Math Operator</th>
<th>Example</th>
<th>X=7, Y=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>X + Y</td>
<td>12</td>
</tr>
<tr>
<td>Subtraction</td>
<td>X - Y</td>
<td>2</td>
</tr>
<tr>
<td>Multiplication</td>
<td>X * Y</td>
<td>35</td>
</tr>
<tr>
<td>Division</td>
<td>X / Y</td>
<td>1</td>
</tr>
<tr>
<td>Exponent</td>
<td>X ** Y</td>
<td>16807</td>
</tr>
<tr>
<td>Modulo</td>
<td>X % Y</td>
<td>2</td>
</tr>
</tbody>
</table>

**Logic Operator**

<table>
<thead>
<tr>
<th>Logic Operator</th>
<th>Example</th>
<th>X=7, Y=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equality</td>
<td>X == Y</td>
<td>False</td>
</tr>
<tr>
<td>Greater Than</td>
<td>X &gt; Y</td>
<td>False</td>
</tr>
<tr>
<td>Less Than</td>
<td>X &lt; Y</td>
<td>True</td>
</tr>
<tr>
<td>Less or Equal</td>
<td>X &lt;= Y</td>
<td>True</td>
</tr>
<tr>
<td>Not Equal</td>
<td>X !=&quot;Y&quot; or X&lt;&quot;Y&quot;</td>
<td>True</td>
</tr>
</tbody>
</table>

Other Logical Operators: AND, OR and NOT