A Few Words On **Testing**…
Test #1
Test #1
Test #2
Test #1
Test #2
Test #3
...
Test #n
Testing Recommendations

• When dealing with **lists** and **indexing** in the lists, verify that indexes are not larger (in magnitude) than the length of the list.

• When **reading integers** with possible **leading 0s**, check what built-in functions will give good results.

• When dealing with **user defined lists**, make sure your program works with **empty lists**.

• **Do not trust yourself!** (Count the number of months in your list!)
Splitting strings

Files

Reading Files

Writing Files

Functions

Function Parameters

Functions Returning Values
line = "The quick red fox jumped over the dog"
line = "The quick red fox jumped over the dog"

line.split( '' )
line = "The quick red fox jumped over the dog"

line.split(' ')
line = "The quick red fox jumped over the dog"

line.split( ' ' )

["The", "quick", "red", "fox", "jumped", "over", "the", "dog" ]
line = "The quick red fox jumped over the dog"

line.split( 'o' )

["The quick red f",
 "x jumped",
 "ver the d",
 "g" ]
line = "hello Roth"

line.split('h')

["", "ello Rot", "]
line = """The quick red fox
jumped over
the lazy brown sleeping dog"""

line.split( ??? )

[ "The quick red fox",
"jumped over",
"the lazy brown sleeping dog" ]
```python
line = """The quick red fox
jumped over
the lazy brown sleeping dog"""

line.split( "\n" )

[ "The quick red fox",
"jumped over",
"the lazy brown sleeping dog" ]
```
line = """The quick red fox
jumped over
the lazy brown sleeping dog""""

line = "The quick red fox\njumped over\nthe lazy...dog"
poem = """Chocolate
Chocolate is the first luxury.
It has so many things wrapped up in it:
Deliciousness in the moment,
childhood memories,
and that grin-inducing
feeling of getting a reward for being good.
--Mariska Hargitay"
"
# display each line centered in 60 spaces.
# first line all uppercase.
# last line right justified in 60 spaces.
Write a program that takes a collection of lines of text, where all the words are separated by commas, and prints only the 2nd word of each line.

Same problem, but this time the program is given a huge long string with lines separated by \n characters.
Splitting strings

**Files**

Reading Files

Writing Files

Functions

Function Parameters

Functions Returning Values
Files
Files
Splitting strings

Files

**Reading Files**

Writing Files

Functions

Function Parameters

Functions Returning Values
Opening a File for **Reading**
Opening a File for Reading
Opening a File for **Reading**
Reading a file:

- Open
- Read
- Close
# read the file into a single string
fileVar = open( "poems.txt", "r" )
text = fileVar.read()
fileVar.close()
# read the file into a single string
fileVar = open( "poems.txt", "r" )
text = fileVar.read()
fileVar.close()

# read the file into a list of strings
fileVar = open( "poems.txt", "r" )
lines = fileVar.readlines()
fileVar.close()
```python
>>> demo = ""

Demo Time

""

>>> 

>>> 
```
Splitting strings

Files

Reading Files

Writing Files

Functions

Function Parameters

Functions Returning Values
Writing a file:

- Open
- Write
- Close
# write a string to a file

text = """The quick red fox
jumped over the lazy brown dog"""

fileVar = open( "poems2.txt", "w" )
fileVar.write( text )
fileVar.close()
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[ GCC 4.0.1 (Apple Inc. build 5493) ] on darwin
Type "copyright", "credits" or "license()" for more information.

```python
>>> demo = "Demo Time"
```

```python
"Demo Time"
```
We stopped here last time...
CSC111: Amount of Work

Level of Difficulty

Semester
CSC111: Amount of Work

Level of Difficulty vs Semester
CSC111: Amount of Work

Level of Difficulty vs. Semester

Frustration
Splitting strings

Files

Reading Files

Writing Files

Functions

Function Parameters

Functions Returning Values
Functions
Functions

are your minions!
They have names...
(def main)
They work only if you call them…

main()
They can work on different things...

daveWorkOn( "Banana" )
daveWorkOn( "Lots of Bananas" )
Real Example

http://cs.smith.edu/dftwiki/index.php/MakeStaticPackage.py_Source_Code
main()

functions
dave() carl() stuart()
Two different types of functions:
def printBar():
    print( 60 * '-' )

def writeDataFile():
    file = open( "test.txt", "w" )
    file.write( "Carl\nJorge\nDave\n" )
    file.close()
Some functions always do the same thing

def printBar():
    print( 60 * '−' )

def writeDataFile():
    file = open( "test.txt", "w" )
    file.write( "Carl\nJorge\nDave\n" )
    file.close()
Some functions **adjust their behavior** depending on what we give them to work with.
work for Dave to perform:  
**Eat** ( *fruit* )  
- open mouth  
- put *fruit* in mouth  
- chew  
- swallow
work for Dave to perform:

**Eat** (fruit)
- open mouth
- put *fruit* in mouth
- chew
- swallow

Dave, eat( banana )
work for Dave to perform:

**Eat( fruit )**
- open mouth
- put *fruit* in mouth
- chew
- swallow

Dave, eat( banana )

Dave, eat( orange )
work for Dave to perform:

**Eat**\( (\text{fruit} ) \)
- open mouth
- put *fruit* in mouth
- chew
- swallow

Dave, eat( banana )

Dave, eat( orange )

Dave, eat( apple )
def daveEats( fruit):
    print( "Dave opens his mouth" )
    print( "and eats the", fruit )

def main():
    daveEats("banana")

main()
def daveEats( fruit ):
    print( "Dave opens his mouth" )
    print( "and eats the", fruit )

def main():
    daveEats( "banana" )
    daveEats( "apple" )
main()

Dave opens his mouth and eats the banana
Dave opens his mouth and eats the apple
def daveEats( fruit ):
    print( "Dave opens his mouth" )
    print( "and eats the", fruit )

def main():
    daveEats( "banana" )
    daveEats( "apple" )

main()
def daveEats( fruit ):
    print( "Dave opens his mouth" )
    print( "and eats the", fruit )

def main():
    daveEats( "banana" )
    daveEats( "apple" )

main()
def daveEats(fruit):
    print("Dave opens his mouth")
    print("and eats the", fruit)

def main():
    daveEats("banana")
    daveEats("apple")

main()
Another Example

def printBar( char, length ):
    print( char * length )

def main():
    printBar( "#", 10 )

main()
Another Example

```python
def printBar( char, length ):
    print( char * length )

def main():
    printBar( "#", 10 )

main()
```

#######
Another Example

```python
def printBar( char, length ):
    print( char * length )

def main():
    printBar( "#", 10 )
    printBar( "a", 5 )

main()
```

#~~~~~~~~~~~
Another Example

def printBar( char, length ):
    print( char * length )

def main():
    printBar( "#", 10 )
    printBar( "a", 5 )

main()

##########
aaaaaa
def printBar( char, length ):
    print( char * length )

def main():
    printBar( "#", 10 )
    printBar( "a", 5 )
    printBar( 3, 5 )

main()
Write a program that sings happy birthday to **Dave**.

Write a program that sings happy birthday to **Dave** and to **Stuart**.

Let’s sing **Happy Birthday to some minions**
Let’s sing Happy Birthday to some minions.

Modify the program so that it sings happy birthday to Dave, Stuart, Jerry, Jorge, Tim, Mark, Phil, Kevin, and Jon.

Modify the program so that it sings happy birthday, and formats the song with bars made with the person’s name, e.g. *Dave*Dave*Dave*Da… above and below the message.
We stopped here last time...
Splitting strings

Files

Reading Files

Writing Files

Functions

Function Parameters

Functions Returning Values
Class Example

Worker 0:
- clap!

Worker 1:
- get a number
- return that number multiplied by 2 and add 1

Worker 2:
- get a number
- return that number modulo 10
Class Example

**Worker 3:**
- get a number
- compute a new number equal to that number multiplied by 2 and incremented by 1
- return the new number modulo 10
# Functions.py
# Example of functions returning values.

def worker1( num1 ):
    num2 = num1 * 2 + 1
    return num2

def worker2( num1 ):
    num2 = num1 % 10
    return num2

def worker3( num1 ):
    num2 = worker1( num1 )
    num3 = worker2( num2 )
    return num3

def main():
    n = 12
    print( "n = ", n, " worker1(n) = ", worker1( n ) )
    print( "n = ", n, " worker2(n) = ", worker2( n ) )
    print( "n = ", n, " worker3(n) = ", worker3( n ) )

main()
Python Version

```python
# Functions.py
# Example of functions returning values.

def worker1( num1):
    num2 = num1 * 2 + 1
    return num2

def worker2( num1):
    num2 = num1 % 10
    return num2

def worker3( num1):
    num2 = worker1( num1 )
    num3 = worker2( num2 )
    return num3

def main():
    n = 12
    print( "n = ", n, " worker1(n) = ", worker1( n ) )
    print( "n = ", n, " worker2(n) = ", worker2( n ) )
    print( "n = ", n, " worker3(n) = ", worker3( n ) )

main()
```

```python
>>> n = 12  worker1(n) = 25
n = 12  worker2(n) = 2
n = 12  worker3(n) = 5
```
Exercise 1:
• Write a function that gets a temperature in Fahrenheit and returns the equivalent Celsius.
• Test the function from the Python shell.
Exercise 1:
• Write a function that gets a temperature in Fahrenheit and returns the equivalent Celsius.
• Test the function from the Python shell.
Function Boot Camp

Exercise 2:
• Write a function that is given the name of a file and returns the contents of the file.
Exercise 2:
• Write a function that is given the name of a file and returns the contents of the file.
Exercise 3:
• Write a function that is given the name of a file and returns the contents stripped of blank lines at the beginning and end of the file.
Exercise 3:
• Write a function that is given the name of a file and returns the contents stripped of blank lines at the beginning and end of the file.

```python
def getTextFromFileStrip( fileName ):
    file = open( fileName, "r" )
    text = file.read()
    file.close()
    return text.strip()

def main():
    data = getTextFromFileStrip( "poem.txt" )
    print( data )

    n = 12
    print("n = ", n, " worker1(n) = ", worker1(n))
```
Exercise 3:
• Write a function that is given a list in the form mmddyyyy, and that returns it as dd Mmm yyyy.
Functions Boot Camp

Exercise 3:
• Write a function that is given a list in the form mmddyyyy, and that returns it as dd Mmm yyyy.

```python
def formatDate(date):
    months = "XXXJanFebMarAprMayJunJulAugSepOctNovDec"
    # date is in mmddyyyy format
    index = int(date[0:2]) * 3
    month = months[index:index+3]
    day = date[2:4]
    year = date[4:]
    return day + " " + month + " " + year

def main():
    date = "02272015"
    newDate = formatDate(date)
    print(date, "-->", newDate)

main()
```
Side-Step & Review
How to Write Programs in Top-Down Fashion

1. Start with the problem

2. **Break** problem down into a **few** simple parts

3. Call a **function** for each **part**

4. Take each part as a separate, **new, smaller problem**

5. **Go back** to Step 2 until the smaller problem can be solved by **just a few** Python **statements**.
Something Fun: Generate Random Poems for Everybody in Class