**Class 2: Introduction to Python**

#### **Review from class 1**

Data types:

* Numbers
  + 1, 4, 23
  + 2 + 89, 4 \*\* 8
* Strings
  + “bob”, “banana”, “cake”
  + “hello “ + “world”
  + “hello”[3], “hello”[2:3]
  + “hello”.capitalize(), “hEllo”.swapcase()
  + len(“hello”), print(“hello”)
  + A complete list of built in string functions is at http://docs.python.org/2/library/stdtypes.html#string-methods
* Lists
  + [2, 5, 100], [True, False, True], [“bob”, “banana”, “elephant”], [[1, 3], [4, 5]]
  + list = [2, 5, 100]
  + list [0], list[1]
  + len(list), print(list)
  + 5 in list
  + A complete list of built in functions is at http://docs.python.org/2/library/stdtypes.html#sequence-types-str-unicode-list-tuple-bytearray-buffer-xrange

Programming concepts

* Variables
  + word = “banana”
  + print(word + “ split”)
* Types of functions
  + Operators: Symbols that can be used to perform operations
    - 3 + 5, “banana” \* 5, “banana”[3],
    - List of python operators is at http://docs.python.org/2/library/operator.html#mapping-operators-to-functions
  + Methods: functions on a particular object
    - “banana”.capitalize()
  + Functions: functions with no particular object these will generally be applicable to more than one data type
    - len(“banana”), print(“banana”)

#### **Functions**

Sometimes it’s useful to re-use code in multiple places. This can be done by writing and calling a function.

*Remember, you can create a file that contains the program by selecting File->New Window in IDLE. Save the file and select Run->Python Shell to run it.*

*Hint: pay attention to indentation!*

#### Example 1

#### Write a function that will sing birthday song:

def sing\_happy\_birthday(name):

happy = "Happy Birthday to "

print happy + "you"

print happy + "you"

print happy + name

print happy + "you"

sing\_happy\_birthday("Molly")

#### Exercises

1. Write a function called \_sum that takes in two numbers and returns their sum. [BONUS] What happens when you give this function two strings or two lists?
2. Write a function called first that returns the first letter of a string. What happens when you pass the function the empty string (“”)?
3. Write a function that takes a word and prints out the word five times.
4. Write a function that takes a string, *word* and a number *repeats* and prints out *word* *repeats* times with a space between each time word comes out. (Hint you probably will print out one more space than you will need.)
5. Write some code that calls that function, and asks for user input to select the word.
6. Write a function that takes a temperature and converts it from Fahrenheit to Celsius. (*hint:* C = 5/9 (*F*-32))

#### **Loops**

Sometimes you might want to do something more than once in your program. What do you think this code does?

*Example 1:*

for i in [1,2,3,4,5]:

print i

#### Exercises:

1. Use a for loop to get the computer to print:

1

little monkeys

2

little monkeys

3

little monkeys

(use separate print statements for the number and the text)

1. Create a new copy of the program (File->New Window) and modify it to use words (one, two, three) instead of numbers.
2. Find out how the range function works. Use it to create a list.
3. Use a loop to write a function that takes a list of numbers and prints out the sum of all those numbers. [BONUS] Write a function that computes the average of all the numbers. What happens when you give these functions a list of strings?
4. [Challenge] Take in a list of words and print out a numbered list. (Hint: You will have to use len() and range())
5. [Challenge] Write a function to print out a right justified Mario staircase of a height specified in the function argument. A staircase of height 4 is below:

# #

# # #

# # # #

# # # # #

|  |
| --- |
| Python Imaging Library (PIL) example http://ywla.bigomega.org/ |

#### **Comparison Operators and Booleans** For reasons we shall see shortly, sometimes we need to know if a statement is true or false.

#### Try in IDLE:

#### >>> 6 > 4

#### >>> 6 == 6

#### >>> 6 != 6

#### >>> True and True

#### >>> True or False

#### >>> not(True)

## *Exercises*

#### Try a testing out the operators above with strings instead of numbers. (Remember, a string is created using "quotes")

#### Try true/false comparisons with these operators: <=, <>

#### Try to string multiple comparisons together. eg: not(True & False), 4 > 3 and 3 > 3

1. See what happens when you use = instead of ==. What is = good for?

#### **Conditionals**

#### You may want your program to behave differently depending on the input, for example maybe it is not your birthday today.

#### Example 3

answer = raw\_input("Is it your birthday? ")

if answer == "yes":

name = raw\_input("What is your name? ")

happy = "Happy Birthday to "

print happy + "you"

print happy + "you"

print happy + name

print happy + "you"

else:

age\_string = raw\_input("How old are you? ")

print "I am also " + age\_string + " years old"

*Exercises*

1. What does elif do? Try replacing else with elif in your program
2. Write a program that asks if the weather is sunny, cloudy or rainy and tells the user what to wear accordingly. Be sure to include a response if they give an unexpected answer!

|  |
| --- |
| Python Imaging Library (PIL) example  http://ywla.bigomega.org/ |

**Dictionaries**

Dictionaries are data structures that allow you access data based on “keys” instead of position. For example, a phone book associated phone numbers with names.

>>> phoneBook = {'Alistair':1234567, 'Beatrice':1234568, 'Cordelia':1234569}

>>>phoneBook['Beatrice']

1234568

>>>phoneBook['Beatrice'] = 9876543 #update an entry

>>>phoneBook

{'Beatrice': 9876543, 'Cordelia': 1234569, 'Alistair': 1234567}

>>>phoneBook['Daphne'] = 1234570

>>>phoneBook

{'Daphne': 1234570, 'Beatrice': 9876543, 'Cordelia': 1234569, 'Alistair': 1234567}

Just like lists and strings, dictionaries have methods.

>>> phoneBook.keys()

['Daphne', 'Beatrice', 'Cordelia', 'Alistair']

>>>phoneBook.pop('Beatrice')

1234568

>>> phoneBook

{'Daphne': 1234570, 'Cordelia': 1234569, 'Alistair': 1234567}

*Exercises*

1. Use a dictionary data structure to create a very small regular dictionary that associates three words with their definitions.
2. Use the “has\_key” method to check if a word is in your dictionary.

**Loops and Dictionaries**

In the loop examples above, we used a loop to iterate over a list. Loops can also iterate over a dictionaries.

>>>phoneBook = {'Alistair':1234567, 'Beatrice':1234568, 'Cordelia':1234569}

>>> for k in phoneBook:

print(phoneBook[k])

1234568

1234569

1234567

#### Exercises:

1. The phone company has incremented all phone numbers by 1! Use a loop to update your phone book.

#### **Data Types Review: Casting from one type to another**

Here are the data types we’ve dealt with so far:

String: s = "I am a string"

Number: n = 9, n = 2.4

Boolean: b = True, b = False

List: list = [1,2,3,4]

*Exercise*

1. What data type will the last line in each example return? When possible give the result.
2. "test" + "ing" + " 1" + " 2" + " 3"
3. 7 == 8
4. "8" == 8
5. int("3") < 8
6. str(3) == "3"
7. str(3)+ str(5)
8. "test"[2]
9. Grocery list: Here we’ll write a program to keep a grocery list and allow the user to add and remove items. In a new file, follow these steps to make the program:
   1. Create a default list of groceries.
   2. Write a function to to print out a list with one item per line.
   3. Use a while loop (*example 4*) to display the list, and ask the user if they want to. If the user types quit, break from the loop..
   4. Modify the program to give the user the options to add or remove an item. If so, ask what they want to add or remove and modify the list accordingly.
   5. Modify the program so it deals with unexpected input.

**Introduction to the Python Standard Library**

Python has built in functions like int() and str() that we learned above. Python is also distributed with a standard library of additional functions. These functions are organized into “modules” and you have to tell Python when you want to use a module. We’re going to use a standard library function that randomly selects items from a list.

First, try

>>>random.choice([1,2,3,4,5])

Now, import the library and try again.

>>>import random

>>>random.choice([1,2,3,4,5])

>>>random.choice([1,2,3,4,5]) #run this line a couple times to see what happens

The complete list of functions in the standard library is available here:<http://docs.python.org/library/>

Exercises:

1. **Grocery - Feeling Lucky:** Update your Grocery List program to return an random item when you type "feeling lucky"
2. **Excuses Challenge**: For those of you who are finished, write a program that has two listss of strings, one called tasks that has elements like "take out the garbage", or "do my homework", and another one called excuses that has elements like "I was abducted by aliens" or "the dog ate it." Now have your program output a random task and excuse sentence like:  
   "I didn’t take out the garbage, because I was abducted by aliens."

#### **Reading and Writing from Files**

We can read and write to files from python. This is the code for opening a new file:

>>>fid = open("//directory//filename.txt","w")

Where directory is the path to some directory on your computer. The second argument, "w", tells Python to open this file and let us write in it.

Try:

>>>print fid

Write some content to the file

>>>fid.write("blah blah blah there is content in this file.\n”)

>>>fid.write(“This is the second line of the file.\n") #\n creates a new line.

Always close the file when you have done working with it:

### fid.close()

Now find the file on your computer and open it in a text editor or word processor. Is the text you just wrote there? Close the file before continuing.

### Open a file for reading:

>>>fid = open("//directory//filename.txt","r") # the ‘r’ is for **r**ead

>>>fid.read() # returns the content of the file

### *Exercises:*

1. Try the closing the file, reopening it, and reading the content with readline() or readlines()
2. Try closing the file and opening by replacing “w” with “a”. What happens when you write a line to the file?
3. Grocery - Write:Modify your grocery list program from last week so that it writes the grocery list to file each time you update the list (replacing the whole file will do).

**Grocery - Read:** Write a separate program (in a new .py file) that reads your grocery list and prints out each item.

### Homework:

1. If you didn’t complete it in class, do the ‘excuses’ exercise (number 18).
2. Given two words test to see if one is an anagram of the other.
3. Write a program that asks a user for a word and outputs a list with the letters from the word in a random order.  
   *Hint: use the list() function to convert a string to a list (myList = list(myString)) and ''.join(myList) to convert a list back to a string.*
4. Rewrite the previous program to be a game where the user has to guess the word that the computer came up with.

**Cliff Notes version:**

* Comparison Operators such as ==, >= to compare two values and return a boolean.
* The if statement can be used to make a program behave differently depending on the input.
* Functions make it possible to reuse code without needing to copy and paste.
* Last week we learned about data structures that hold one piece of information such as strings and integers. This week we learned about lists that can hold multiple pieces of information. For example, list = [1,2,”three”,[“another”,”list”]]
* A for loop lets you repeat something over and over in a program.
* Some data types can be caste to other data types. For example, a string can become an integer, int(100) = “100”
* Python has some built in functions but the standard library has even more. To use these functions, import the library.
* There is a standard library function that can pick a random item from a list.

>>>import random

>>>random.choice([1,2,3,4])