Getting to know Python

Python is a computer programming language developed in 1991 by Guido van Rossum at the Center for Mathematics and Computer Science (CWI) in the Netherlands. It is a language meant to be both high-level and multi-purpose. High-level means that the individual statements should use ordinary words and mathematical systems that are easy to read and understand. Multi-purpose here means that the language can be used to help solve any type of problem given to the computer.

Python is widely recognized as one of the easiest programming languages to learn. There is very little syntax (grammar and typographical rules) or vocabulary to worry about. It is a popular language today, used extensively by such firms as Google and in computer graphics studios. Python is particularly efficient in the way it can process large amounts of numerical or text data. To simplify coding, Python includes many built-in functions, such as sorting numbers, creating images, and looking up Web pages.

Here are some practical things to keep in mind when working with Python. For more example code, please refer to the Python programs in lab, for example the program list.py.

1. Python loves variables. And Python understands that a variable can be used for different purposes (i.e. data types). For example, we can have:
	1. Numbers (both integer and real) radius = 2.5
	2. Text (a.k.a. string) name = “Martha”
	3. True/False (a.k.a. boolean values) leap = True
	4. List of values L = [ 5, 4, 7, 3 ]
2. Note that with lists, we sometimes want to access a specific element, such as the first one. We use the brackets [ ] to specify which value we want. For example, if L is the name of a list, then L[0] means just the first value in a list. Note that these positions start at zero. So the 10th value (if there is a 10th value) is L[9], and so on.

Note that the individual values in the list do not have to be numbers. They can be strings or even lists themselves!

1. The division operator is /. But please use caution when you divide. There are actually two kinds of division: integer division and real-number division. If you divide two integers, the result is also an integer! If you divide real-numbers (or one integer and one real number), the result is a real number.

What this means is that if you need the exact quotient, you want to use real-number division. But if you don’t care about throwing away a remainder, then integer division is fine. Here are contrasting examples:

1. / 3.0 computes exactly one-third: 0.33333333…

1/3 literally divides 3 into 1 and discards any remainder, so the result is 0!

1. When doing integer division, if you are just interested in getting the remainder, use the % operator. For example, let’s see if there is going to be a Summer or Winter Olympics in a given year:

if year % 4 == 0:

 print “Summer Olympics in the year “ + str(year)

elif year % 4 == 2:

 print “Winter Olympics in the year” + str(year)

1. The symbol for raising a number to a power is \*\*. Thus, 2\*\*5 is 32.
2. Traversing a list means visiting all the values in the list one by one. Here is a very elegant way to do it in Python:

L = [ 5, 4, 7, 3, 2, 6, 1]

for number in L:

 print number

On the other hand, there is another, more old-fashioned, way to traverse a list that makes explicit use of the list’s positions or “apartment numbers”:

 for i in range (0, 7):
 print L[i]

1. The difference between = and ==.

The = sign is used to assign a value to a variable. So, x = 5 means we want x to have the value 5. On the other hand, == means we are asking if two values are equal. So, x == 5, which might appear in an if-statement, asks whether x equals 5 or not. Asking this question does not change the value of x.

Here is an example that uses both = and ==:

 if hours == 40:

 overtime = False

1. Lines that begin with # are called comments. The purpose of a comment is to tell the human reader what is going on. It is an excellent idea to start your Python program with a comment. Here, you should indicate the name of the source file, your name as the author, and a brief statement about the purpose of the program. Beyond this, it is also a very good idea throughout the program to insert comments near where there are statements whose purpose is not obvious. For example, you may want to have a comment immediately before a loop that explains what that loop is accomplishing.
2. Anytime you introduce a loop or if-statement, you must put a colon at the end of the line. This includes statements that start with these words: for, if, elif, and else.

Here are some simple example problems, followed by their solutions in Python.

1. Ask the user for the length and width of a rectangle. Output the area and perimeter.
2. Ask the user what 7 + 5 is. Tell the user if the answer is correct or not.
3. Count how many numbers in a list equal 5.

Solutions:

# rectangle.py
# Find the area and perimeter of the user’s rectangle

length = input("What is the length? ")
width = input("What is the width? ")

area = length \* width
perimeter = 2 \* (length + width)

print "The area is " + str(area)
print "the perimeter is " + str(perimeter)

-----------------------------------------------------------
# quiz.py
# See if the user knows what 7+5 is.

answer = input("What is 7 plus 5?")

if answer == 12:
 print "Good job!"
else:
 print "Sorry, that’s not correct."

-----------------------------------------------------------

# count5.py
# Count the 5’s in a list

L = [ 5, 4, 2, 5, 1, 2, 5 ]

numFives = 0
for number in L:
 if number == 5:
 numFives = numFives + 1

print "I found " + str(numFives) + " fives in the list."

-----------------------------------------------------------