# CMPT 120—LAB WEEK 3 (Jan. 20)

*In Lecture 4, you were introduced to IDLE and the Python shell; then in Lecture 5, we started exploring elementary data types in Python (integers, floats, strings) and we saw two controls structures (if statements, and iterating over a range of integers). We tried out many expressions in class to give you the beginning of an intuition on how Python works.*

*For those of you that did not have a computer handy during class for Lecture 5, I strongly recommend that you go over all expressions of the lecture on your own.*

*This lab will continue the exploration.*

**GOALS OF THIS LAB:**

- Get comfortable with the Python environment (IDLE and Python shell).
- Get comfortable with elementary Python expressions.
- Understand how to write a small program from instructions

General instructions:

- It is recommended you do these exercises in teams of 2 people each. Make sure to take a partner that has roughly the same experience and understand of the topic as you. (If you team up with someone who comprehends the material much better than you, then chances are you will not learn anything.)

- Save your work for future reference and to show the TAs and instructor, so as to show you have actively participated in the lab.

- **Save your files in your home folder U:** (files saved on the Desktop may get lost). Python files should have the **.py** extension written explicitly—the *extension* is the end of filename.

Submission of this lab:

- Take a whole sheet of paper (no splitting a sheet in half), write your name, SFU ID (not student number).

- Whenever you see an assignment question, noted **Q1**, **Q2**, **Q3**, etc., write down the answer on your sheet.

- Hand it in to a TA before leaving. *These questions will not be graded, and serve only to check that you have understood the concepts that are introduced—and help you, if you do not. Take this seriously!*

## —PART 1: EXPLORATION—

*Exercises in this first part encourage you to explore the behavior of Python.*

**EXERCISE 1: first steps in IDLE**
*The purpose of this exercise is for you to make sure you discover the IDLE environment which you will be coding in for most of the term.*

First, if you are not familiar with the IDLE environment, follow the tutorial (available in many different languages) "one day of IDLE toying", which is available here:

https://hkn.eecs.berkeley.edu/~dyoo/python/idle_intro/

The tutorial is slightly outdated (it is seven years old, and uses Python 2.4.3, whereas we are using Python 2.7.6 or a similarly recent 2.7.x version), but should be helpful if you are completely lost. Do not follow the links provided in the tutorial, they are beyond the scope of this lab.

*Note: there also some instructions on how to use IDLE at the end of Lecture 4.*

Then, you will obtain a Python program containing the following (if you have chosen not to follow the tutorial, simply open IDLE, create a new file, and paste the following code in it):

```
print "Hello world!"
print "Here are the ten numbers from 0 to 9"
for i in range(10):
  print i,
print "I'm done!"
```

**Listing 1: IDLE example**

*Note: the function `range(b)` is the equivalent to `range(0, b)` using the function with saw during the lecture that takes two parameters; in other words `range(b)` gives all numbers from 0 to b-1.*

When the program's window is selected, you can run the program by going to the "Run" menu and selecting "Run module", and the output will occur in the shell.

Play around with the code. In particular note what happens when you:

- Change the indentation of the sentences
- Remove the colon ":"
- Alter some of the names, keywords
- Remove some quotes and/or make any other modification

## EXERCISE 2: iterations

*The purpose of this exercise is to understand better how the number iteration works, and what mistakes should not be made.*

**For this exercise, if you are typing the code directly in the Python shell, make sure you RESTART THE PYTHON INTERPRETER. You can do this in IDLE by selecting the prompt/shell's window, and then going in the "Shell" menu, and selecting "Restart Shell."**

As we've seen in Lecture 5, we can use the `for` loop to repeat instructions several times, and to count integers within a range.

### The loop's iteration variable

The variable name that comes right after the `for` keyword (for instance, in Listing 2, this is i) is called the *iteration variable*. This variable is modified at the beginning of every iteration (repetition) of the loop and takes the value of the next element to be processed. We are going to investigate a few of the particularities of this variable.

1. What happens when you enter this code:

```
for i in range(1, 10):
  print 123
```

**Listing 2: a simple iteration**

2. What about this one:

```
for i in range(1, 10):
  print i
```

**Listing 3: another simple iteration**

3. Does the name of the loop's variable matter?

```
for crazyVariableName in range(1, 10):
  print crazyVariableName
```

**Listing 4: changing the loop's variable name**

4. What about this one:

```
for crazyVariableName in range(1, 10):
  print i
```

**Listing 4: undeclared variables**

5. What about this one (*be careful to respect the indentation*):

```
i = 101010101
print i
for i in range(1, 10):
  print i
print i
```

**Listing 5: using an existing variable as iteration variable**

<u>Loop indentation</u>

We mentioned the notion of blocks. Find out for yourself what this means, by running each of the following examples.

```
sumvar = 0
for i in range(1, 10):
  sumvar = sumvar + i
print sumvar
```

**Listing 6**

```
sumvar = 0
for i in range(1, 10):
  sumvar = sumvar + i
  print sumvar
```

**Listing 7**

```
sumvar = 0
for i in range(1, 10):
sumvar = sumvar + i
print sumvar
```

**Listing 9**

Submission questions

**Q1.** Once a loop is done, what value does its iteration variable hold?
**Q2.** Describe what is the behavior for programs in Listing 6 and Listing 7. If they are different, explain why.

## EXERCISE 3: the if structure

*The purpose of this exercise is to take a look at the if structure a little bit more in detail than in Lecture 5.*

Remember that `True` is a condition that is always verified and `False` is a condition that is never verified.

1. Run the program of Listing 10, and check that the correct messages are being printed.

```
if True:
  print "This will always get printed."
else:
  print "This will never get printed."

if False:
  print "This again will never be printed."
else:
  print "But this will."
```

**Listing 10: using True and False**

2. What happens when you replace `True` by `1` and `False` by `0`? What happens when you replace `True` by `"True"` and `False` by `"False"`?

3. Which print statements in the following example will be printed?

```
if True:
  print "Print #1"
elif True:
  print "Print #2"
elif True:
  print "Print #3"
else:
  print "Print #4"
print "Print #5"
```

**Listing 11: priority and control flow in an if structure**

**Q3.** Can you explain the behavior of the program in Listing 11? Which of the following statements would you say is true: a) Python runs every part of an if statement that is verified; or b) Python runs the first part of an if statement that it finds to be verified, and then skip all other parts of the if statement?

## EXERCISE 4: correct a program

*The purpose of this exercise is to check if you have integrated the main syntax points of programming in Python.*

**For this exercise in particular, it is strongly recommended you do not work directly in the Python shell, but instead type the code in a window, and use the "Run module" option.**

The program in Listing 12 is supposed to print all integers from 1 to 10, and then print the sum of these numbers. The output it is supposed to give is shown in Listing 13. But there are a number of bugs inside the program, all of which have been discussed either in Lecture 5, or you should have understood from the previous exercises.

Correct the mistakes (there are five mistakes to be corrected).

**Q4.** Try to write the name or description of all the errors you get (this requires trying the program after correcting each error to see what Python says).

```
print "This program will print all numbers from 1 to 10,"
print "and then their sum."
print ""
print "Here are the numbers

sumvar = 0
for i in range(1, 10)
  print k
sumvar = sumvar + i

print "And the sum of 1+2+...+10 is"
print "sumvar"
```

**Listing 12: program with five mistakes**

The output it is supposed to give (once you have made it work!).

```
This program will print all numbers from 1 to 10,
and then their sum.

Here are the numbers
1
2
3
4
5
6
7
8
9
10
And the sum of 1+2+...+10 is
55
```

**Listing 13: output of the program (once corrected)**

## —PART 2: DESIGN AND PROGRAMMING—

*Exercises in this second part encourage you to write your own code from scratch.*

Notes and recommendations for all exercises of this type (in this lab, and in your future work):

- **To test and debug,** include enough printing messages so that you may see intermediate and final results and also from where the printing is done.

- Alternatively if your program is running fine (that is, there is no syntax error), but rather your problem is that your problem is not computing the result you intend it to, you can also use the Python Tutor website to see step by step what is happening.

- Always **consider different cases and special limit cases**: when your parameters have values that require special case.

- Make sure to initialize all your variables before using them.

- Don't forget you can put comments in your code, by putting a sharp # at the beginning of the line.

- In a program, do not write sentences that are so long that you cannot see them in the IDLE editor original screen.

**EXERCISE 5: write your own program, "Sums and differences of integers"**
*The purpose of this exercise is to have you practice your coding skills, and practice using and understand the for loop, and indentation.*

Write a program that will add up all integers from 1 to 10, then subtract all integers from 11 to 20, then add all integers from 21 to 100, then subtract all integers from 101 to 110. And prints out the total.

- Define a result variable, which you initialize (assign an initial value) to 0.
- Write four loops one after the other.
    - The loop from 1 to 10 adds the integers to your result variable.
    - The loop from 11 to 20 subtracts the integers from your result variable.
    - And so on!
- Print out the result variable.

**EXERCISE 6: write your own program, "Difference of even and odd integers"**
*The purpose of this exercise is to have you practice your coding skills, and practice using the for loop and the if statement.*

1. Write a program that will initialize a result variable, and then add all odd integers between 1 and 100 and subtract all even integers between 1 and 100. In other words, computer 1-2+3-4+5-6+…-100.
   - Define a result variable and initialize it to 0.
   - Write a loop ranging over the integers from 1 to 100.
   - Inside the loop put an if structure that sees if the current value of the iteration variable (go to exercise 2 if you don't remember what an iteration variable is) is even or odd.
   - To do this, use the modulo operator, written with a percent sign %, that gives the remainder of the integer division, i.e., 10 % 5 = 0 because 5 divides 10 exactly, but 11 % 5 = 1, because you can subtract 5 twice from 11, and then you have 1 which is too small to subtract 5 again.

2. Once your program works, copy it, and modify it so that you are now summing all numbers such that their remainder by 5 is either 2 or 4 (that is, if `myvar` contains that number, `myvar % 5 == 2` is true or `myvar % 5 == 4` is true).

**Q5.** What is the result of this second program?

**EXERCISE 7: write your own program, "Convert years and months"**
*The purpose of this exercise is to check if you have integrated the main syntax points of programming in Python.*

Write a Python program that initializes two variables (choose adequate names for these variables), one consisting of a (positive integer) number of years, and the other a (positive integer) number of months not necessarily less than 12. The program should then print the total number of months, and then the number of years and months, where the months is less than 12.

For instance, if the number of years is 18 and the number of months is 27, your program could output what is in Listing 14.

```
18 years and 27 months are equivalent to: 243 months
243 total months are equivalent to: 20 years and 3 months
```

**Listing 14: using True and False**

A limit case in this example (see general remarks at the beginning of this second part) would be to see what happens to your program if you have 0 years and/or 0 months.

*Note: to print a mix of integers and strings on the same line, you can use the print statement with several values separated by commas. Each value will be separated by a space.*

```
myvar = 10
myothervar = 20

print "First variable is", myvar, "other:", myothervar
```

**Listing 15: printing several values in the same statement**

*This will produce the following output.*

```
>>> First variable is 10 other: 20
```

**Listing 16**