**Lab 01 – Lab procedure, Python basics, and IDLE**

**COMP130 - Introduction to Computing**

**Dickinson College**

**Instructor: John MacCormick**

# Introduction

This lab will introduce you to some of the tools and practices that we will be using throughout this course. In particular, this lab focuses on:

The use of *IDLE* for writing and executing Python programs.

The practice of *Pair Programming*.

Basic Python language expressions and statements.

# IDLE

We will be using a program called IDLE for writing and executing Python programs. If using a college computer, Python and IDLE are already installed. If using your own computer, first install Python from python.org. IDLE is automatically installed as part of that process, so you can open it on your own computer.

Work with your partner on a single computer. Within IDLE, create a new file called lab01.py:

* If you don’t already have a folder for your work in this course, create a COMP130 folder. Inside that, create a lab01 folder.
* In File menu, choose New File
* In File menu, choose Save As
* Navigate to your lab01 folder and save as lab01.py

Put a comment at the top of the file, explaining what the file is and who the authors are. See section 2.7 of the textbook for a description of comments. Here is an example:

# Lab01 submission for comp130-01

# Authors: Hannah Jang and Angelina Danilova

# Date: 9/15/2035

# Pair Programming

We will be using pair programming for the labs in this course. The instructor will provide an overview of pair programming at the start of today’s lab. Based on this overview, answer the questions below.

To record your answers, create a new document in any word processing program. Create only one document for your team. You will eventually convert this document into PDF and submit it as part of the lab submission. We call this your *responses document*. Most labs will require you to submit one or more Python program files (e.g., lab01.py) and a responses document in PDF format (e.g., lab01-responses.pdf).

Put a title, authors, and date at the top of your responses document. Then record your answers to the following questions.

**Q1:** What are the two roles in pair programming? Give brief descriptions (one sentence each).

**Q2:** Describe a benefit that you anticipate from working in pairs on the labs (one sentence).

**Q3:** Briefly discuss why it is important to switch roles while pair programming in the classroom (one sentence).

# Python Code in the IDLE shell

Currently you have two windows open in IDLE: the lab01.py file and the IDLE shell. The IDLE shell can be used interactively: you can type some Python, press enter, and immediately see the result. For example, at the “>>>” prompt, type 1+1 and press enter. You should see the answer.

If you haven’t already done so, work through sections 1.3, 1.4, and 1.5 of the textbook, trying each example in the interactive IDLE shell. Then give a one-sentence answer to the following question in your responses document.

**Q4:** Explain the different output to the Python inputs type(42.0) and type('42.0')

# Python Code in a script

It’s not feasible to execute complex computer programs in the interactive IDLE shell. Instead, we store all the Python commands in one or more Python files, such as lab01.py. These files are sometimes called *scripts*. Review section 2.4 of the textbook to understand the difference between the interactive shell and scripts, if desired.

**Q5:** Now add two print() commands to your lab01.py script. Each of the two commands should print out some information about you. Label your code with a comment indicating the question number. Here is an example:

# Q5

print('My name is Hannah and my favorite band is The Fratellis')

print('My name is Angelina and my favorite sports team is North Carolina Courage')

Run your script using the Run Module command in the Run menu of IDLE. Seek assistance if it is not producing the desired output in the shell window.

Now, before moving on to the next question, *comment out* your code for the previous question. To do this, select the above lines of code and then choose Comment Out Region from the Format menu. It should now look like this:

### Q5

##print('My name is Hannah and my favorite band is The Fratellis')

##print('My name is Angelina and my favorite sports team is North Carolina Courage')

Save your lab01.py script. Now you are ready to move on to the next question. Your answer to the previous question has been commented out, so it will not be executed when you run the program. If you ever need to go back and execute this code, use the Uncomment Region command from the Format menu. The instructor will uncomment your code if needed when grading, so it is fine to submit it commented out, as above. To assist with grading, it will be greatly appreciated if you format your solutions with separators that make it easy to find the various questions. For example, the following formatting for your entire lab01.py file would be preferred:

# Lab01 submission for comp130-01

# Authors: Hannah Jang and Angelina Danilova

# Date: 9/15/2035

####################################

### Q5

##print('My name is Hannah and my favorite band is The Fratellis')

##print('My name is Angelina and my favorite sports team is North Carolina Courage')

####################################

####################################

# Q6 -- start writing your Q6 code here

####################################

Edit your lab01.py file until it looks similar to the above. Then you are ready to begin question 6.

# Arithmetic Operators

**Q6:** Give a line of Python code that computes and prints the sum 3+4+5+6. Hint: print(18) is not an adequate answer. Your code must demonstrate how the computation is done in Python. Here, the expected answer is print(3+4+5+6). The same principle applies to other homework and labs this semester.

**Q7:** What are the seven arithmetic operators in Python and what operation does each perform? Give your answer in your responses document. Hint: There are five operators in the textbook section 1.4 and two operators in section 5.1 – ignore the bitwise operator ^ in section 1.4.

**Q8:** Give a line of Python code that computes and prints the total cost of 3 items that each cost $2.57 and 2 items that each cost $1.98. Give your answer in your lab01.py script. Test that it is working correctly, then comment it out. (If there is a very small numerical error in the answer, you can ignore it. When using floating-point arithmetic on a computer, pining numerical round-off errors are not uncommon.)

The style of work should be clear by now. Non-code answers go in your responses document, which will be converted to PDF and submitted when the lab is complete. Coding answers go in your lab01.py script. Always run your code to check that it works, then comment it out so that you can move on. The lab01.py script, with most or all code commented out, will also be submitted when the lab is complete.

**Q9:** Repeat the previous question if a 10% discount is applied to the lower priced item.

**Q10:** Give a line of Python code that computes and prints the number of whole meters (no portions left over) in 27 feet.

**Q11:** A relief agency had a supply of 2457 meals are to be distributed equally to 183 people. Write Python code that computes and prints the number of meals received by each person and the number of meals that are left over.

# Order of Operations

Python's arithmetic operators follow the same order of operations rules (i.e., have the same *order of precedence*) as you learned in your grade school mathematics courses. Also, as in mathematics, parentheses can be used in Python to dictate the order of operations and/or to clarify the meaning of an expression. The following questions are intended to give you additional practice writing Python expressions, thinking about the *human readability* of code.

Consider the following expression that "[Tried to Stump the Internet](https://www.nytimes.com/2019/08/02/science/math-equation-pedmas-bemdas-bedmas.html?smid=nytcore-ios-share)":



**Q12:** What value does Python compute for this expression? (You will need to insert a \* to represent the implicit multiplication after the first 2.) Based on that result, when evaluating operators with equal precedence (e.g., \* and / or + and -) does Python evaluate them left to right? Or right to left?

**Q13:** Rewrite the expression so that the multiplication occurs before the division.

In constructing a new weather app, a software team is adding an option to allow temperatures to be displayed in Fahrenheit or Celsius. To do so, they need expressions for converting from one to the other. After some research, they find that a temperature in Fahrenheit is converted to Celsius by subtracting 32 and then dividing by 1.8.

**Q14:** Give code that converts 84 degrees Fahrenheit to Celsius and prints the answer.

**Q15:** Give code that converts 28 degrees Celsius to Fahrenheit and prints the answer.

Consider the following scenario. If you buy more than 10 items at $2.87 then a 10% discount is applied to half of the items, and you can buy 10 more at a 50% discount. A direct translation of this scenario for someone buying 15 items plus the additional 10 leads to the following messy expression:

15 \* 2.87 - 15//2 \* 2.87 \* 0.1 + 10 \* 2.87 \* 0.5

**Q16:** Rewrite this as an equivalent expression by factoring out the price of the item. Print the answer twice, computed once using the messy formula and once using the simpler formula. Verify that the answer is the same in both cases.

**Q17:** Recall that the volume of a cylinder with a height and radius is:

The SpaceY corporation is making plans to build a rocket with a radius of 3 meters and must be able to carry (i.e., have a volume of) 9000 cubic meters of combined fuel and cargo. SpaceY wants to know how tall the rocket will need to be. A new programmer does some quick algebra to solve for the height:

Thus, the approximate height of the rocket is given by .

The new programmer's attempt to translate this to Python resulted in the following incorrect expression:

9000 / 3.14\*3\*\*2

Rewrite this expression in two different ways that are correct. Print your answer both times to show that it calculates the same value.

**Q18:** Think about the above expression. Describe two distinct reasons why it is useful for programmers to use parentheses in expressions. (One sentence for each reason is sufficient.)

The author of our text offers wise words at the end of section 2.5: "I don’t work very hard to remember the precedence of operators. If I can’t tell by looking at the expression, I use parentheses to make it obvious."

Submit your responses document as PDF and your lab01.py code file. Submit both documents to the appropriate dropbox on Moodle. For each team of two people, only one person should submit. You must work as a team and submit only one version of your solutions. Except in highly unusual circumstances, both members of the team will receive the same grade.

Note: This lab is considered a practice and will not be graded. However, you are encouraged to complete it and submit it, so that you can practice all aspects of the lab process.

Acknowledgment: This lab was originally authored by Grant Braught, with edits by John MacCormick.