SY201 – Cyber Operations Fundamentals; Programming Assignment **01**

Calculations and Conversions

Name: _____ Alpha: _____

Begin this assignment by making sure your repo is up to date. Change to **~/repo201** and type:

git pull

All the files required for this assignment, including an electronic version of this handout, are available in:

~repo201/programming/pa01

Welcome to your first programming assignment in Python!

For this assignment, you're going write a Python program to perform conversions and calculations based on Newton's second law of motion, which describes the relationship between an object's mass and the amount of force needed to accelerate it. The law is commonly stated as:

$F = ma$

(Force = mass * acceleration)

F is expressed as *Newtons; m* is represented in kg; a is represented in m/s^2

The more mass an object has, the more force you need to accelerate it. Also: the greater the force, the greater the object's acceleration.

1. **Your Program Should Do This**:

    a. Prompt the user for a given mass in **pounds (*lbs*)**.
    b. Prompt the user for a given acceleration in **feet / second^2 (*ft/s^2*)**.
    c. Convert and display the mass in **kilograms (*kg*)**.
    d. Convert and display the acceleration in **meters per second^2 (*m/s^2*)**.
    e. Calculate and display the force on the object in ***Newtons***.

2. **Design Requirements:**

    Think carefully about how you want to approach this assignment before you start. It's rarely a good idea to approach a computing problem by opening up your editor and just start typing. Think first about the kinds of variables you will use, their names, how they will interact. Take out a piece of paper and pencil and sketch out your ideas. Draw simple flow

diagrams that capture the major components of your program. These diagrams help you decompose larger problems into smaller chunks, and are often translated directly into blocks of code.

Figure 1 is an example of a flow diagram template for this assignment. Different programmers will take different approaches to a particular problem, so there may be many ways to complete the diagram below.

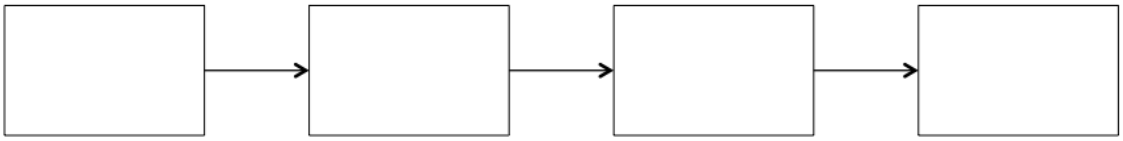

Figure 1. Flow Diagram Template

3. **Error Handling**

Your program doesn't need to handle input errors at this point. Nevertheless, once your program is working you should attempt to provide it invalid (bogus) input to see what happens. In future assignments, we're going to examine some of the ways a program can crash or be compromised simply based on what values you enter into it.

4. **Program Flow**

a. Provide meaningful, descriptive prompts to your user when you ask for input values. This prompt: `"mass (lb): "`
is not as descriptive as this prompt: `"Object's mass in lbs: "`

b. Don't worry if your output isn't very "pretty" (for example, you might get something like: 873436.4903128779 Newtons). We'll learn about rounding floating-point numbers and formatting output later.

5. **Assumptions You May Make:**

When you prompt for input, you'll always get valid floating-point numbers. It will not always be the case that you get valid input in our later programs, so start thinking now about things like, "*Instead of a valid mass, what if I type GONAVY! instead?*" Go ahead and try that ☺

6. **Hints:**

   a. Don't hesitate to refer back to the *Liang* text often. It's a great resource.

   b. Give your variables meaningful, descriptive names. While it's true that you could represent the mass of the object in pounds using a variable named `x`, it will be much easier to read and debug your code if you use a variable name like `massInLbs` instead.

   c. The directory for this project also includes an executable version that you can run to get a sense for how your program should operate. The program is called `newton`.

      Let's assume you're working in your `~/shares/sy201` directory. You can run `newton` by first copying it to your working directory and then adjusting its permissions with the following commands:

      i. `cp ~/repo201/programming/pa01/newton .`
      (*the trailing space and period in the command above are important*)
      ii. `chmod 755 newton`

      You only have to perform steps (i) and (ii) once. After that, you can run the program any time you want by typing: `./newton`

   d. Don't worry if you get slightly different answers in your program than what you see when you run `newton`. Depending on how many decimal places you use in your conversion factors when you convert feet to meters and pounds to kilograms, the answers may differ slightly. If your answers differ by a wide margin then you may want to check your code.

   e. It takes time and practice to become a proficient coder; just like learning to play a musical instrument or learning a new language. Don't get discouraged if it seems confusing at first. By the end of the course, you'll look back and be amazed how far you've come.

7. **Deliverables And Due Dates:**

   a. Using pen or pencil, complete the flow diagram on page 5. Complete this *before* you start coding. Be sure to indicate your name and alpha.

b. Your completed source code (the one ending on .py).

c. Submit parts (a) and (b) by 2359 on Tuesday, 28 August, in accordance with your instructor's directions.

SY201 – Cyber Operations Fundamentals; Programming Assignment `01`

Calculations and Conversions

Name: _____ Alpha: _____

Fill-in the diagram below, where each block represents a single "chunk" of the overall problem. This does not mean that each block represents one line of code, rather, use plain language to describe what needs to be done for each part of the problem. If necessary, feel free to make any other notes or sketches on this page that help you explain your approach.