Lab 1 – A Gentle Introduction to Python

1. Here is our program from class for converting Celsius to Fahrenheit:

```
def c2f():
    celsius = eval(input("Please enter a Celsius temperature: "))
    fahrenheit = 9/5 * celsius + 32
    print(celsius, "degrees Celsius is", fahrenheit, "degrees Fahrenheit")
```

   Write a similar program called **f2c()** that converts Fahrenheit to Celsius using the formula $C = \frac{5(F-32)}{9}$

2. Write a program called **miles2inches()** to compute the number of inches in a given number of miles entered by the user. There are 12 inches in a foot and 5280 feet in a mile. Your output should look like the example below:

```
>>> miles2inches()
Enter the number of miles: 6
There are 380160 inches in 6 miles
```

3. Write a program called **liters2gallons()** to convert a number of liters (entered by the user) to gallons. There are 3.79 liters in a gallon. Your output should look like the example below:

```
>>> liters2gallons()
Enter the number of liters: 379
379 liters equals 100.0 gallons
```

4. Write a program called **pizza()** that calculates the cost per square inch of a circular pizza. Your program should ask the user for the pizza's diameter in inches and the price of the pizza. Hint: the formula for the area A of a circle is $A = \pi r^2$, where r is the radius of the circle (half the diameter) and $\pi = 3.14159$. For example:

```
>>> pizza()
What is the diameter of the pizza in inches? 9
How much does the pizza cost? 100
Your pizza costs $ 1.5719020002413655 per square inch!
```

5. The built-in Python function **`round`** can be used to round a floating-point value to a desired number of decimal places. For example, if the value of the variable `x` is currently 0.6666666666, you can update `x` to be 0.67 by executing the following assignment statement:

```
 x = round(x, 2)
```

   Use this idea to improve the output generated by your pizza() program, and test it on the example shown above.

6. Write a program called **hms()** that asks the user to enter an amount of time represented separately as hours, minutes, and seconds, and then converts this to an equivalent value expressed as a decimal number of hours. Hint: there are 60 seconds in a minute, and 60 × 60 seconds in an hour. Some examples:

```
>>> hms()                      >>> hms()                      >>> hms()
Hours: 3                       Hours: 5                       Hours: 0
Minutes: 30                    Minutes: 16                    Minutes: 75
Seconds: 0                     Seconds: 30                    Seconds: 180
That equals 3.5 hours          That equals 5.275 hours        That equals 1.3 hours
```

7. Write a program called **height()** that converts people's heights into different units of distance. Your program should ask the user for their height in feet and inches. It should then report the user's total height in inches, feet, and centimeters (assuming 1 foot equals 30.48 centimeters), as shown below:

```
>>> height()
Number of feet: 5
Number of inches: 11
You are 71 inches tall
You are 5.916666666666667 feet tall
You are 180.34 cm tall

>>> height()
Number of feet: 7
Number of inches: 6
You are 90 inches tall
You are 7.5 feet tall
You are 228.6 cm tall
```

8. Write a program called **redundant()** that asks the user to enter a number, stores the number in a variable called `n`, and then prints the word "redundant" over and over, `n` times, using a for-loop of the form "`for c in range(n):`" For example:

```
>>> redundant()
Enter a number: 3
redundant
redundant
redundant
```

9. Next, make a copy of your redundant() program from the previous exercise and change its name to **count()**. Then change the statement `print("redundant")` to `print(n)`. Run the program a few times and see what happens when you enter different input values, such as 3, 5, and 10 (you can try other input values if you like). Do you understand what's going on?

10. Next, modify your **count()** program again, this time by changing `print(n)` to `print(c)`. Now what happens when you run the program and enter different inputs such as 3, 5, and 10? What is the first value that gets printed out each time? What happens to the variable `c` on each cycle of the loop?

11. Write a program called **squares()** that asks the user for a number *n* and then, for each number starting from 0 and going up to but not including *n*, prints that number and the square of that number as shown below. Hint: the structure of this program will be very similar to your count() program from the previous exercise.

```
>>> squares()
Enter a number: 4
0 squared is 0
1 squared is 1
2 squared is 4
3 squared is 9
```

12. Now write a version of the previous program called **squares2()** that prints squares up to *and including* the value the user enters. Hint: modify the expression inside the `range( )`. For example:

```
>>> squares2()
Enter a number: 4
0 squared is 0
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
```

13. Write a program called **sequence()** that asks the user for a starting number and an ending number, and prints out the sequence of numbers from start to end, inclusive. Hint: your loop will need exactly *end* – *start* + 1 loop cycles. For example, the following loop runs for 10 – 5 + 1 = 6 cycles:

```
>>> sequence()
Enter start value: 5
Enter end value: 10
5
6
7
8
9
10
```

14. Write a program called **addSequence()** that asks the user for a starting number and an ending number, and computes the *sum* of the sequence. For example, adding up the numbers 5 through 10 (inclusive) is shown below. Hint: use an extra variable called `total` to keep track of the running total.

```
>>> addSequence()
Enter start value: 5
Enter end value: 10
The sum of the numbers from 5 to 10 is 45
```

15. Write a program called **product()** that is just like addSequence(), except that it computes the *product* instead of the sum of the numbers. For example:

```
>>> product()
Enter start value: 3
Enter end value: 6
The product of the numbers from 3 to 6 is 360
```

16. Write a program called **beer()** that asks the user for a number *n* of bottles of beer, and prints out the lyrics to the classic song "*n* Bottles of Beer on the Wall". You can print a blank line between each verse using an empty `print()` statement, but don't worry about making the output completely grammatical. For example, we could generate the lyrics to "3 Bottles of Beer on the Wall" like this:

```
>>> beer()
How many bottles of beer? 3
3 bottles of beer on the wall
3 bottles of beer
Take one down, pass it around
2 bottles of beer on the wall

2 bottles of beer on the wall
2 bottles of beer
Take one down, pass it around
1 bottles of beer on the wall

1 bottles of beer on the wall
1 bottles of beer
Take one down, pass it around
0 bottles of beer on the wall
```

17. Now add one final verse to your bottles-of-beer program, which should get printed at the very end, after the for-loop finishes. For example, the ending of "3 Bottles of Beer" should look like this:

```
...
1 bottles of beer on the wall
1 bottles of beer
Take one down, pass it around
0 bottles of beer on the wall

No more bottles of beer on the wall
No more bottles of beer
Go to the store, buy some more
3 more bottles of beer on the wall
```