COSC 130 – HW 03 Instructions

General Instructions

Create a new notebook named **`HW_03_YourLastName.ipynb`** and complete problems 1 – 8 described below.

Any set of instructions you see in this document with an orange bar to the left will indicate a place where you should create a markdown cell. For each new problem, create a markdown cell that indicates the title of that problem as a level 2 header.

Any set of instructions you see with a blue bar to the left will provide instructions for creating a single code cell.

Read the instructions for each problem carefully. Each problem is worth 6 points. An additional 2 points are allocated for formatting and following general instructions.

The assignment should be completed using only base Python commands. Do not use any external packages.

Assignment Header

Create a markdown cell with a level 1 header that reads: "COSC 130 - Homework 03". Add your name below that as a level 3 header.

Problem 1: Calculating Exam Scores

In this problem, you will use a loop to calculate scores for two students who have taking a multiple-choice exam with 25 questions. Assume that the correct answers on the exam are as follows:

Correct Answers: `D, B, C, A, C, D, A, C, C, B, D, A, B, D, C, D, C, D, C, A, B, D, C, B, A`

The answers submitted by the two students are provided below.

Student 1 Answers: `A, B, C, A, B, D, A, A, C, B, D, A, D, C, C, B, C, D, B, A, D, D, C, C, A`
Student 2 Answers: `D, A, C, A, B, D, A, C, C, B, D, A, B, D, A, D, C, D, C, A, B, C, C, B, A`

Perform the following steps in a single code cell:

1. Create lists of named **`correct`**, **`answers1`**, and **`answers2`** to store the correct answers for the exam, as well as the answers submitted by the two students. The answers should be stored as strings of the form `'A'`, `'B'`, `'C'`, and `'D'`.
2. Create variables named **`count1`** and **`count2`** to store the number of correct answers for each student.
3. Use a single loop to simultaneously loop over all three lists. As you do so, count the number of correct answers for student 1 and student 2.
4. When the loop is finished executing, divide each count of correct answers by the total number of questions and multiply by 100 to obtain the grades for both students. Store the results **as integers** in variables named **`grade1`** and **`grade2`**.
5. Print the results with descriptive messages as shown below, with the **`xxxx`** symbols replaced with the appropriate values. Match the format and spacing exactly.

```
Student 1 Grade: xxxx%
Student 2 Grade: xxxx%
```

No lists should be created for this problem other than the three described in Step 1. Only one loop should be used in this problem.

Problem 2: Calculating Price of a Bulk Order

WidgCo is a company that sells widgets. The company uses the bulk pricing policy described below.

- If 100 or fewer widgets are ordered, then each widget costs $350.
- If 200 or fewer widgets are ordered, the first 100 widgets each cost $350, and each additional widget costs $300.
- If more than 200 widgets are ordered, then the first 100 each cost $350, the next 100 each cost $300, and each widget beyond the first 200 will cost $250.

In this problem, you will determine the price for orders containing the following numbers of widgets:

```
84, 100, 126, 150, 186, 200, 216, 248
```

Perform the following steps in a single code cell:

1. Create a list named **`quantities`** to store the order quantities mentioned above.
2. Loop over this list. Each time the loop executes, perform the following steps:
    - Calculate the price of the order.
    - Print a message in the format shown below, with the xxxx symbols replaced by the appropriate values. Match the format exactly, including the period at the end of the sentence. Do not include extra spaces.

      ```
      The cost for an order of xxxx widgets is $xxxx.
      ```

Only one list should be created for this problem, and only one loop should be used. Your loop should contain only one print statement.

Problem 3: Creating an Amortization Schedule

Suppose that $1000 is borrowed at the beginning of the year. The loan is charged interest at an annual interest rate of 5%. The borrower will repay the loan by making payments of $150 at the end of the year until the loan is repaid. Note that when the final payment is made, the balance will be less than $150, and so a partial payment will be made at that time.

Perform the following steps in a single code cell.

1. Create variables named **`balance`**, **`i`** and **`pmt`** to store the initial loan amount, annual interest rate, and annual payment amount mentioned in the paragraph above. Also create a variable **n** to track the number of payments.
2. Use a **while** loop to perform the tasks described below. The loop should continue to execute for as long as the balance of the loan is greater than 0.
    - Increase the number of payments that have been made by 1.
    - Calculate the new balance. This will be done by multiplying the current balance by **`1+i`**, and then subtracting the payment amount. Store the result back into **`balance`**, rounded to 2 decimal places.
    - If the new balance is less than 0, then calculate the the final payment amount. This will be equal to the new balance plus the payment amount. Then set the balance to 0.
    - Print the message below, with the xxxx symbols replaced with the appropriate values. Match the format exactly.

      ```
      The balance at the end of year xxxx is $xxxx.
      ```

The first two lines of output should appear as shown below:

```
The balance at the end of year 1 is $900.00.
The balance at the end of year 2 is $795.00.
```

No lists should be created for this problem, and only one loop should be used.

Use a new code cell to print the final payment amount with a message as shown below:

```
The amount of the final payment will be $xxxx.
```

Problem 4: Hailstone Sequences

In mathematics, a hailstone sequence is a sequence creates as follows:

- Select a starting value for the sequence.
- If a particular term in the sequence is equal to n, then the next term will be equal to $n/2$ if n is even, and will be equal to $3n + 1$ if n is odd.
- The sequence continues until reaching a term that is equal to 1.

For example, the hailstone sequence beginning with 12 is: `12, 6, 3, 10, 5, 16, 8, 4, 2, 1`

It is widely believed that any starting value will result in a sequence that eventually reaches 1, but this has not been proven to be true.

Use **while** loops to calculate the hailstone sequences beginning with 15, 17, and 22, storing the sequences (in order) in lists named **`hs15`**, **`hs17`**, and **`hs20`**. This will require three separate (but nearly identical) loops. Print the contents of the three lists.

Problem 5: Predator/Prey Model

Assume that a certain ecosystem contains both rabbits and wolves. The size of the rabbit population and the size of the wolf population both fluctuate, and have an effect on each other. Suppose that the current sizes of the populations can be used to determine the sizes of the populations one year from now according to the following population model:

- Let r_c be the current size of the rabbit population.
- Let w_c be the current size of the wolf population.
- Let r_n be the size of the rabbit population one year from now.
- Let w_n be the size of the wolf population one year from now
- The population sizes one year from now are determine by the following equations:

$$r_n = 1.5 \cdot r_c - 0.002 \cdot r_c \cdot w_c$$
$$w_n = 0.8 \cdot w_c + 0.0003 \cdot r_c \cdot w_c$$

Suppose that the current rabbit population is 1000 and the current wolf population is 200. Under these circumstances, the populations will fluctuate for several years, but the rabbit population will eventually die out completely.

In this problem, you will be asked to calculate the size of each population at the end of each year until the the rabbit population dies out.

Perform the following steps in a single code cell:

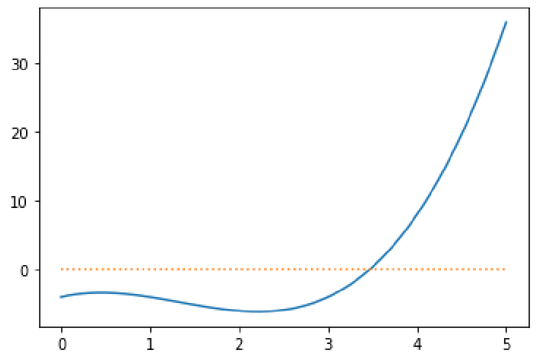
1. Create variables **`r_pop`** and **`w_pop`** to store the current rabbit and wolf populations.
2. Print the first two three rows of the table shown below.
3. Use a loop to determine the new populations at the end of each new year. Each time the loop executes, perform the steps below. The loop should continue to run for as long as both populations are greater than zero.
   - Store the current population values in temporary variables named **`r_pop_prev`** and **`w_pop_prev`**.
   - Use the formulas provided to calculate the new population sizes. Round the results to the nearest integer, and store these values in **`r_pop`** and **`w_pop`** (as integers).
   - Print a new line of the table below with updated values.

```
Year      R_pop    W_pop
-----------------------
0         1000     200
1         1100     220
2         1166     249
...       ...      ...
```

No lists should be created for this problem, and only one loop should be used.

Problem 6: Finding a Root of a Polynomial Function

Consider the function $f(x) = x^3 - 4x^2 + 3x - 4$. A plot of this function for values of x between 0 and 5 is provided below.



The plot above shows that the function $f(x)$ is equal to zero somewhere between $x = 3$ and $x = 4$. In other words, the plot shows that the equation $x^3 - 4x^2 + 3x - 4 = 0$ has a solution somewhere between $x = 3$ and $x = 4$. Your goal in this problem will be to approximate the value of this solution.

Perform the following steps in a single code cell:

1. Create variables **`x1`** and **`x2`**, setting them equal to 3 and 4, respectively. We will use a loop to update the values of these variables to close in on to the solution.
2. Create a variable named **`val1`**, setting it equal to the value you would obtain by plugging **`x1`** into the function f. Create a variable named **`val2`**, setting it equal to the value you would obtain by plugging **`x2`** into the function f. Notice that we can see from the plot that **`val1`** should be negative and **`val2`** should be positive.
3. Use a loop to iteratively update the values of the four variables above as described below. This loop should continue to execute until the absolute value of **`val1`** and **`val2`** are both less than 0.00001. Each time the loop executes, perform the following steps:
   - Create a variable named **`new_x`** that is equal to the average of **`x1`** and **`x2`**.
   - Create a variable named **`new_val`** that is equal to **`f(new_x)`**.
   - If **`new_val`** is negative, then set **`x1`** to **`new_x`** and set **`val1`** to **`new_val`**. Otherwise, set **`x2`** to **`new_x`** and set **`val2`** to **`new_val`**.
4. When the loop is finished executing, **`x1`** and **`x2`** should be very near each other, and near to the solution. Take the average of the two values to be the approximation for the solution. Print your result with a message as shown below. Round the apprxoimate solution to four decimal places.

```
The approximate solution is xxxx.
```

No lists should be created for this problem, and only one loop should be used.

Problem 7: Grade Database

In this problem, you will be displaying (fictional) grade information that is assumed to be taken from a university's grade database. The information is stored in two dictionaries and three lists, as shown below.

Copy the code below into a code cell, and then execute that cell.

```
students = {
    146832:{'first':'Brendan', 'last':'Small'},
    147354:{'first':'Melissa', 'last':'Robbins'},
    149126:{'first':'Jason', 'last':'Penopolis'},
    149735:{'first':'Fenton', 'last':'Mulley'}
}

courses = {
    'ENGL 101':'Composition I',
    'ENGL 104':'Composition II',
    'MATH 117':'College Algebra',
    'MATH 151':'Calculus I',
    'CHEM 103':'General Chemistry I',
    'ECON 201':'Macroeconomics'
}

sid = [149126, 146832, 147354, 149735, 149126, 146832,
       146832, 149735, 149126, 147354, 147354, 149735]
cid = ['ENGL 101', 'MATH 117', 'ENGL 104', 'CHEM 103', 'MATH 117', 'ECON 201',
       'ENGL 101', 'ENGL 101', 'CHEM 103', 'ENGL 104', 'MATH 151', 'MATH 117']
grade = ['D', 'C', 'B', 'A', 'B', 'C',
         'A', 'F', 'B', 'A', 'A', 'C']
```

The **`students`** dictionary is a lookup table that allows you to obtain the name of a student based on their Student ID number. The **`courses`** dictionary is a lookup table that allows you to obtain the name of a course based on the Course ID (which consists of a department name and course number). The lists **`sid`**, **`cid`**, and **`grade`** are parallel lists that represent columns from a grade database. The **`sid`** list contains student ID numbers associated with 12 different records form the database, the **`cid`** list contains course ID information for the 12 records, and the **`grade`** list contains the grade obtained by the corresponding student in the corresponding course.

Use a loop to print a table displaying the twelve records. The table should be formatted as follows:

- The first row of output should contain column headers.
- The second row should be a dividing line consisting of 69 dashes.
- The table should have columns for student ID, first name, last name, course id, course name, and grade.
- The number of characters reserved for each column should be 9, 10, 12, 11, 22, and 5 (in that order).
- All columns should be left-aligned except for the grade column, which should be right-aligned.

The first few rows rows of the output are shown below. Try to match the format of this output exactly.

```
SID      First     Last        CID        Course Name           Grade
---------------------------------------------------------------------
149126   Jason     Penopolis   ENGL 101   Composition I             D
146832   Brendan   Small       MATH 117   College Algebra           C
```

Problem 8: Warehouse Inventory

Suppose that a company sells five products with product codes **p101**, **p107**, **p122**, **p125**, and **p126**. The company has three warehouses, which are located in St. Louis, Chicago, and Kansas City. The retail value for each of the five products and the inventory for each of the warehouses are stored in dictionaries, as shown below.

Copy the code below into a code cell, and then execute that cell.

```
prices = {'p101':37.52, 'p117':56.98, 'p122':43.72, 'p125':48.33, 'p126':52.45}

inventory = {
    'STL':{'p101':520, 'p117':315, 'p125':258, 'p126':345},
    'CHI':{'p117':864, 'p125':612},
    'KC':{'p101':264,  'p122':772, 'p125':467}
}
```

Note that none of the warehouses stock all five of the products. You will now be asked to calculate the total retail value for the company's stock of each of the five products.

Use a code cell to create a dictionary named **`total_inv`**. This dictionary should ultimately contain five key/value pairs. The key for each pair should be a product code, and the value should be the total retail value for the company's stock of the corresponding product. This task can be accomplished as follows:

1. Create **`total_inv`** as an empty dictionary.
2. Loop over the key/value pairs stored in the **`prices`** dictionary. In each iteration of the loop, do the following:
    a. Create a variable named **`total`** to store the total retail value of the current product.
    b. Loop over the values stored in the **`inventory`** dictionary. Note that each of these values will be a dictionary representing the inventory for a single warehouse. In each iteration, check to see if the current product being considered (from the outer loop) is stored at the warehouse. If it is, calculate the retail value of the inventory for this product at this warehouse and add it to **`total`**.
    c. Add the product code and the total to **`total_inv`** as a new key/value pair.

Loop over the key values pairs in **`total_inv`**, printing these pairs in the format shown below.

```
p101: $xxxx
p117: $xxxx
p122: $xxxx
p125: $xxxx
p126: $xxxx
```

Submission Instructions

When you are done, click **Kernel > Restart and Run All**. If any cell produces an error, then manually run every cell after that one, in order. Save your notebook, and then export the notebook as an HTML file. Upload the HTML file to Canvas and upload the IPYNB file to CoCalc, placing it in the **Homework/HW 03** folder.