CSE 231 **Spring 2014**

Computer Project #4

Assignment Overview

This assignment focuses on the design, implementation and testing of a Python program which uses nested control structures to solve the problem described below.

It is worth 30 points (3% of course grade) and must be completed no later than 11:59 PM on Monday, February 3.

Assignment Specifications

1. The program will repeatedly prompt the user to select from the following menu of options:

    A. Display the value of the factorial of N.
    B. Display the value of the sine of X.
    C. Display the value of the cosine of X.
    M. Display the menu of options.
    Q. Exit from the program.

The program will accept both upper case and lower case letters for the menu options.

The program will display the menu of options once when execution begins, whenever the user selects Option M, and whenever the user selects an invalid menu option.

2. Option A will permit the user to enter an integer value N. If N is a positive integer, it will calculate and display the value of the factorial of N.

3. Option B will permit the user to enter a real value X (measured in radians). It will calculate and display the value of the sine of X.

4. Option C will permit the user to enter a real value X (measured in radians). It will calculate and display the value of the cosine of X.

5. The program will use a power series approximation for Options B and C.

Assignment Deliverable

The deliverable for this assignment is the following file:

proj04.py – the source code for your Python program

Be sure to use the specified file name (“proj04.py”) and to submit it for grading via the **handin** system before the project deadline.

Assignment Notes

1. All program output will be formatted for readability.

2. The program will detect, report and recover from erroneous user inputs.

3. You may assume that the user enters a string representing a valid integer value when prompted for an integer. However, you should not assume that the integer value will be positive.

4. You may assume that the user enters a string representing a valid floating point value when prompted for a real.

5. Be sure to prompt the user for the inputs in the correct order. And, your program cannot prompt the user for any supplementary inputs. To grade your program, your TA will enter the same series of inputs for each student's program.

6. The program will calculate the values for Options A-C using repetitive execution (loops). It will not use existing functions (such as math.sin() or math.cos()) to perform the calculations.

7. The power series approximation for the sine of X can be expressed as:

$$\text{sine}(X) = X - (X^3/3!) + (X^5/5!) - (X^7/7!) + (X^9/9!) \ldots.$$

Note that an individual term in that power series can be expressed as:

$$(-1)^k * X^{2k+1} / (2k+1)! \quad \text{where } k = 0, 1, 2, 3, \ldots.$$

Similarly, the power series approximation of the cosine of X can be expressed as:

$$\text{cosine}(X) = 1 - (X^2/2!) + (X^4/4!) - (X^6/6!) + (X^8/8!) \ldots.$$

Note that an individual term in that power series can be expressed as:

$$(-1)^k * X^{2k} / (2k)! \quad \text{where } k = 0, 1, 2, 3, \ldots.$$

8. When computing the sine of X or the cosine of X, the program will expand the power series until the absolute value of the next term in the series is less than 1.0e-8 (the specified epsilon). That term will not be included in the approximation.

9. For large values of X, the number of terms required to get a good approximation of the sine of X or the cosine of X becomes quite large. Since sine and cosine are periodic functions, you can map large values of X closer to zero by getting the remainder of X divided by 2 * pi. Use the % operator for that purpose.

10. Your program may not use any library modules and it may not contain any "import" statements (such as "import math").

Suggested Procedure

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step is best done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python statements, you must work on your own.

- Write a simple version of the program. Run the program and track down any errors.

- Use the **handin** system to turn in the first version of your program.

- Cycle through the steps to incrementally develop your program:
  - Edit your program to add new capabilities.
  - Run the program and fix any errors.
  - Use the **handin** system to submit your current version of the program.

- Use the **handin** system to submit your final version of the program.

- You would be wise to back up your files on your H: drive, also.

- Be sure to log out when you leave the room, if you’re working in a public lab.