Assignment 3 – Due Thursday March 12, 2021 at 11:55 pm

Each program must start with a multi-line comment as shown below. For Assignment 3 question 1 replace x with 3 and y with 1. Replace each occurrence of DentStew and Stew Dent with your name. Replace yyyy/mm/dd with the date you completed the program.

```
# DentStewAxQy.py
#
# Course:      COMP 1012
# Instructor:  Ramin Soltanzadeh
# Assignement: x Question y
# Author:      Stew Dent
# Version:     yyyy/mm/dd
#
# Purpose:     The purpose of the program.
```

Begin each program with the following statement.

```
from time import ctime
```

The name of each program should be of the form:
    LastnameFirstnameAnQm.py

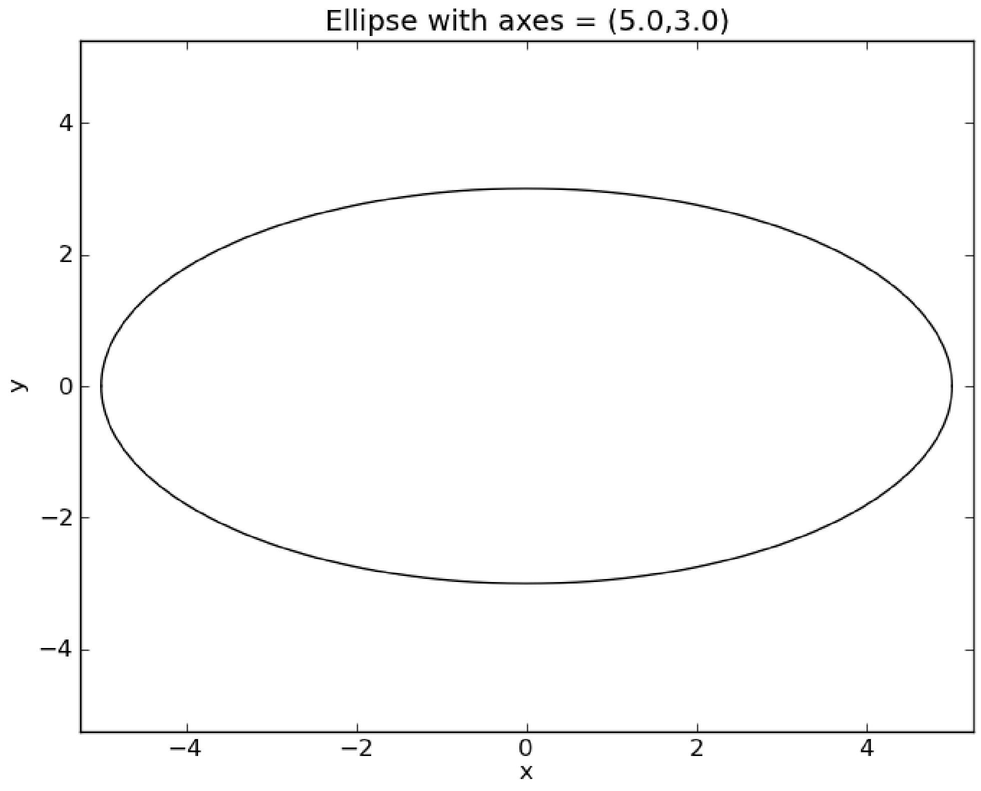
If your name is Stew Dent and the program is for assignment 3 question 1 then the name of the program would be:
    DentStewA3Q1.py

Submit your solution to this assignment by email. Attach the .py file for each program. There should be only one email to which you attach each of the programs. Do **NOT** use zip, rar or any other packaging of your files!

You must use only the material covered in the course notes for weeks 1 to 9 to solve the problems in this assignment.

In questions 1, 2 and 3 of this assignment you will solve the same problem, calculating the approximate circumference of an ellipse. To do this sum the lengths of the hypotenuses of very small triangles to compute the approximate circumference of an ellipse. In question 1 you must use lists and in question 2 you must use arrays but not vector arithmetic and in question 3 you must use arrays and vector arithmetic. You will use the *timeit* module to determine how much time each program uses to calculate the circumference of the ellipse for a large number of intervals and compare the times.

An ellipse has a major axis and a minor axis, where the length of the major axis is greater than the length of the minor axis. The length of the semi-major axis is one half the length of the major axis and the length of the semi-minor axis is one half the length of the minor axis. An example of an ellipse is shown below where the length of the semi-major axis is 5.0 and the length of the semi-minor axis is 3.0.



τ

For each question in this assignment you will use the function named *displayTerminationMessage* from assignment 2.

Question 1

You must use lists in this question. Do not use arrays in this question!

The purpose of this question is to write a complete Python program that computes the approximate circumference of an ellipse. The ellipse is defined by the lengths of the semi-major and semi-minor axes.

Write a function that begins with the following header:

```
def getPositiveNumber(prompt):
```

Call this function to get all input from the user. Valid input is either an *int* greater than or equal to zero or a *float* greater than or equal to zero. The function displays the *prompt* to tell the user what to enter. If the user does not enter valid input display the appropriate error message as described below and display the *prompt* again asking for more input from the user. Repeat this until the user enters valid input. When the user enters valid input return that input.

Error situations:

- If the user presses enter/return without entering anything display the message 'Missing input!'
- If the input causes an exception when passed to *eval* display the value of the input and the message 'is not valid!'
- If the input is neither an *int* nor a *float* display the value of the input and the message 'is not a number!'
- If the input is less than zero display the value of the input and the message 'is less than zero!'

Write a function that begins with the following header:

```
def computeXcoordinates(a, intervals):
```

Given the value of **a**, the length of the semi-major axis, and the number of **intervals** create a list of equally spaced X coordinates from x = 0 to x = **a**. The first element in the list is 0 and the last element in the list is **a**. The number of X coordinates in the list is intervals + 1. Return the list of x coordinates.

Note: You **must** use list comprehension to create the list of X coordinates.

Write a function that begins with the following header:

```
def computeYcoordinates (a, b, xValues):
```

Given the values of **a**, the length of the semi-major axis, **b**, the length of the semi-minor axis and a list of X coordinates named **xValues** this function creates a list of Y coordinates using equation 1. There will be one Y coordinate for each X coordinate. Return the list of Y coordinates.

$$y = b \cdot \sqrt{1 - (\frac{x}{a})^2} equation 1$$

Note: You **must** use list comprehension to create the list of Y coordinates.

Write a function that begins with the following header:

```
def calcCircumference(a, b, intervals):
```

Call *computeXcoordinates* to get the list of X coordinates and then call *computeYcoordinates* to get the list of corresponding Y coordinates. Use these two lists to compute the approximate circumference of an ellipse using the method described in class. Display the approximate circumference to 14 decimal places using exponential format. See the sample run of the program for the exact format of the output.

This function does not return a value.

The main program does the following:

- Using string replication display a line of dashes.
- Display a message indicating that zero may be entered after any prompt to terminate the program.
- In a loop:
  - call *getPositiveNumber* to input the value of ***a***, the length of the semi-major axis, as a float, if the value returned by *getPositiveNumber* is equal to 0 exit from the loop
  - call *getPositiveNumber* to input the value of ***b***, the length of the semi-minor axis, as a float, if the value returned by *getPositiveNumber* is equal to 0 exit from the loop
  - call *getPositiveNumber* to input the integer number of ***intervals***, if the value returned by *getPositiveNumber* is equal to 0 exit from the loop
  - use the *Timer* and *timeit* functions from the *timeit* module to calculate the time it takes to compute the circumference of the ellipse.
  - display the time it took to compute circumference of the ellipse to 3 decimal places, see the sample output for the exact format of the output
- Call *displayTerminationMessage* to display the termination message.

To use the *Timer* and *timeit* functions from the *timeit* module to calculate the time it takes to compute the circumference of the ellipse cut and paste the following statements into your program:

```
t = timeit.Timer('calcCircumference(a, b, intervals)',
    'from __main__ import calcCircumference, a, b, intervals')
computeTime = t.timeit(1)
print("Compute time using lists is %.3f seconds." % computeTime)
```

There are two underscores before and after the word *main* in the statements given above.

To use the *Timer* and *timeit* functions import the *timeit* module.

Note: **There is NO function named *main* in this program!**

A sample run of the program is shown below.

```
-----------------------------------------------------------------------------

To terminate the program enter 0 after any prompt.

Enter the length of the semi-major axis in cm (> 0):
Missing input!

Enter the length of the semi-major axis in cm (> 0): sdsfg
sdsfg is not valid!

Enter the length of the semi-major axis in cm (> 0): True
True is not a number!

Enter the length of the semi-major axis in cm (> 0): -1
-1 is less than zero!

Enter the length of the semi-major axis in cm (> 0): 10.25

Enter the length of the semi-minor axis in cm (> 0): 7.75

Enter the number of intervals (> 0): 10000000

The approximate circumference of the ellipse is 5.68217058429445e+01 cm.
Compute time using lists is 7.271 seconds.

Enter the length of the semi-major axis in cm (> 0): 0

Programmed by Stew Dent.
Date: Tue Feb  16 13:11:22 2021
End of processing.
```

Question 2

Do NOT use vector arithmetic in your solution to question 2.

Modify your solution to question 1 to use arrays instead of lists. Add the appropriate import statement.

Modify the function named *computeXcoordinates* so that it creates and returns an array of X coordinates.

Modify the function named *computeYcoordinates* so that it converts the list of Y coordinates into an array given an array of X coordinates.

Modify a print statement in the main program so that instead of displaying "The compute time using lists is" it displays "The compute time using arrays is".

Do **not** make any other changes to your program as they are not allowed.

Test your program using the same data used for question 1.

Question 3

You must use arrays and vector arithmetic in your solution to question 3.

Modify your solution to question 2 to use vector arithmetic on the arrays. Add or modify import statements as necessary.

Modify the function named *computeYcoordinates* so that it uses vector arithmetic to create an array of Y coordinates given an array of X coordinates. Return the array of Y coordinates. There must not be any loops in this function.

Modify the function named *calcCircumference* so that it uses vector arithmetic. There must not be any loops in this function.

Modify a print statement in the main program so that instead of displaying "The compute time using arrays is" it displays "The compute time using vector arithmetic is".

Do **not** make any other changes to your program as they are not allowed.

Test your program using the same data used for questions 1 and 2.

Question 4

Run the three programs that are your solutions to questions 1, 2 and 3 with **a**=10.25, **b**=7.75 and for 10 million intervals. Type your answers to the following questions in a .py file and hand in that file. The marker will not run the file.

a) How long did it take for your solution to question 1 to compute the circumference of the ellipse?
b) How long did it take for your solution to question 2 to compute the circumference of the ellipse?
c) How long did it take for your solution to question 3 to compute the circumference of the ellipse?
d) Which program ran the slowest?
e) Which program ran the fastest?
f) What do you conclude from this? Do not simply state what you observed. State when it is appropriate to use lists and when it is appropriate to use arrays in the solution of a problem.

Question 5

The purpose of this question is to write a complete Python program that computes the frequencies of palindromes in some English language text. The text is in a file named *relativity.txt*, which you must download from Moodle.

Insert the following function near the beginning of your program.

```
def getWords(filename):
    """ filename -> list of words from the file
    Given the name of a file return a list of words from
    the file.
    """
    # filename - the name of the file containing the words
    # infile - the file descriptor
    # data - the content of the file as one string
    infile = open(filename, 'r')
    data = infile.read()
    infile.close()
    return data.split()
```

Later in the course we will learn what each statement does. This function reads the text from the file and splits the file up into a list of words and then returns the list of words.

Write a function that begins with the following header:

```
def isPalindrome(word):
```

This function is given a string of characters known as *word*. The characters might not form an actual word, but that doesn't matter. The word is a palindrome if there are at least three characters in the word and the characters read the same from right to left and from left to right. For example, the word level is a palindrome. Return True if the word is a palindrome, otherwise return False.

Note: You are **not** allowed to use the *reverse* function or slices in this function.

Write a function that begins with the following header:

```
def findPalindromes(text):
```

This function is given a list of strings named *text*. Most of the strings are words but some of them are not, but that doesn't matter. Create two lists named *palindromes* and *frequencies*. The list *palindromes* holds the unique palindromes from the list named *text*. The list *frequencies* holds the frequencies of the unique palindromes. For each word in *text* call *isPalindrome* to determine if the word is a palindrome. If the word is a palindrome and is not in the list of palindromes add the word to the list of palindromes and add 1 to the list of frequencies. If the word is a palindrome and is in the list of palindromes add 1 to the corresponding frequency in the list of frequencies. Return the list of unique palindromes followed by the list of the frequencies of the palindromes.

Hint: The *index* function can be used to find the position of the palindrome in the list of palindromes.

Write a function that begins with the following header:

```
def displayPalindromes(palindromes, frequencies):
```

This function is given the list of palindromes and the list of frequencies. Create a new list that holds the palindromes sorted in ascending order. Use the *sorted* function to do this. Display the heading as shown in the sample run of the program. Display the palindromes in ascending order along with the associated frequencies one pair of values per line as shown in the sample run of the program. The palindrome is left justified in 20 character positions. The frequency is in 9 character positions. This function does not return anything.

Write a function that begins with the following header:

```
def main():
```

This function does the following:

- prints a line of dashes using string replication
- calls *input* to get the name of the file containing the text
- loops as long as the name of the file is an empty string, in the loop:
  - displays the error message as shown in the sample run of the program
  - calls *input* to get the name of the file containing the text
- calls *getWords* to get the list of words that make up the text
- calls *findPalindromes* to get the lists of palindromes and frequencies
- calls *displayPalindromes* to display the lists of palindromes and frequencies
- calls *displayTerminationMessage* to display the termination message

The main program, not to be confused with the function named *main*, contains all the import statements, the functions and a call to the function named *main*.

Acknowledgment:

The data in the file relativity.txt is from the Project Gutenberg EBook of Relativity: The Special and General Theory, by Albert Einstein

A sample run of the program is shown below.

```
----------------------------------------------------------------

Enter the name of the file containing the text:
The name of the file must not be an empty string!

Enter the name of the file containing the text: relativity.txt

Palindrome          Frequency
***                   3
did                  5
refer                 3
tenet                 1

Programmed by Stew Dent.
Date: Tue Feb  16 14:01:26 2021
End of processing.
```

Hand in your complete program.