Midterm Test

Practices

Sample Programs

- Write a program to find all of the divisors of a positive integer **n**.
- Must divide the integer **n** by the the integers from 1 up to and including int(sqrt(**n**)).
- If **n** % **i** == 0, where 1 ≤ **i** ≤ int(sqrt(**n**)), then **i** is a divisor of **n** and (**n** / **i**) is also a divisor of **n**.
- Store the divisors in a list.
- After all of the divisors have been found sort the list into ascending order.

- A perfect number is a number whose divisors (excluding the number itself) add up to the number.
- The divisors of 6 are 1, 2, 3 and 6.
- Exclude 6 and add up the other divisors.
- 1 + 2 + 3 = 6
- 6 is a perfect number.
- What other numbers are perfect numbers?
- We will also determine if the number entered into our program is a perfect number.
- Example: divisors.py

```
from time import ctime
from math import sqrt

def getPosInt(prompt):
    # prompt - the prompt to display to the user for input
    # number - the number entered by the user
    while True:
        number = input(prompt).strip()
        if number != '':
            try:
                number = eval(number, {}, {})
            except:
                print('Invalid input!')
            else:
                if type(number) is int:
                    if number > 0:
                        break
                    else:
                        print(number, 'is not a positive integer!')
                else:
                    print(number, 'is not an integer!')
        else:
            break
    return number
```

```
def findDivisors(number):  # find divisors of number
    divisors = []
    for divisor in range(1, int(sqrt(number))+1):
        if number % divisor == 0:
            divisors.append(divisor)
            divisors.append(number//divisor)
        # if
    # for
    divisors.sort()  # sort into ascending order
    return divisors
# findDivisors


def displayDivisors(divisors):# display the divisors
    for divisor in divisors:
        print(divisor)
    # for
# displayDivisors
```

```
def perfectNumber(divisors): # is this a perfect number?
    if sum(divisors[:-1]) == divisors[-1]:
        print(divisors[-1], 'is a perfect number!')
# perfectNumber

def main(): # get input from user and process it
    print('\n----------------------------------------')
    number = getPosInt('Enter a positive number (Enter to quit): ')
    while type(number) is int:
        print('The divisors for %d are:' % number)
        divisors = findDivisors(number)
        displayDivisors(divisors)
        perfectNumber(divisors)
        number = getPosInt(
                        'Enter a positive number (Enter to quit): ')
    print("""
Programmed by Stew Dent.
Date: %s.
End of processing.""" % ctime())

main()
```

```
--------------------------------------
Enter a positive number: 1001
The divisors for 1001 are:
1
7
11
13
77
91
143
1001

Programmed by Stew Dent.
Date: Thu May  31 15:22:17 2018.
End of processing.
```

- Write a program to find the perfect numbers between 2 and some upper bound (1 is not considered to be a perfect number).
- The program should ask for the upper bound and display each perfect number along with its divisors.
- How many perfect numbers are there between 2 and 1000.
- What are those perfect numbers?
- Example: perfectNumber.py

```
from time import ctime
from math import sqrt

def getPosInt(prompt):
    while True:
        number = input(prompt).strip()
        if number != '':
            try:
                number = eval(number, {}, {})
                 except:
                print('Invalid input!')
                 else:
                if type(number) is int:
                    if number > 0:
                        break
                    else:
                        print(number, 'is not a positive integer!')
                    else:
                    print(number, 'is not an integer!')
         else:
            print('Missing input!')
return number
```

```
def findDivisors(number):
    divisors = []
    for divisor in range(1, int(sqrt(number))+1):
        if number % divisor == 0:
            divisors.append(divisor)
            divisors.append(number//divisor)
        # if
    # for
    divisors.sort()
    return divisors

def displayDivisors(divisors):
    for divisor in divisors:
        print(divisor)
```

```
def perfectNumber(divisors):
    return sum(divisors[:-1]) == divisors[-1]

def main():
    print('\n-----------------------------------------------')
    UPPER_BOUND = getPosInt('Enter the upper bound: ')

    for number in range(2,UPPER_BOUND + 1):
        divisors = findDivisors(number)
        if perfectNumber(divisors):
            print('The divisors for %d are:' % number)
            displayDivisors(divisors)
            print(number, 'is a perfect number!')

    print("""
Programmed by Stew Dent.
Date: %s.
End of processing.""" % ctime())

main()
```

```
Enter the upper bound: 1000
The divisors for 6 are:
1
2
3
6
6 is a perfect number!

The divisors for 28 are:
1
2
4
7
14
28
28 is a perfect number!

The divisors for 496 are:
1
2
4
8
16
31
62
124
248
496
496 is a perfect number!
```

- Write a program to calculate the area of a circle whose radius is **r**.
- This can be done by computing the area under the curve of a circle for x = 0 to x = **r** (which is ¼ of a circle) and multiplying by 4.
- To approximate the area under a curve add together the area of a large number of very narrow trapezoids that fit in the circle for x = 0 to x = **r**.
- The result is an approximation of the area of the circle.
- Example: areaCircleLists1.py

```
from time import ctime
from math import sqrt, pi

def getPosNum(prompt):
    while True:
        number = input(prompt).strip()
        if number != '':
            try:
                number = eval(number, {}, {})
            except:
                print('Invalid input!')
            else:
                if type(number) is int or type(number) is float:
                    if number >= 0:
                        break
                    else:
                        print(number, 'is not a positive number!')
                else:
                    print(number, 'is not an number!')
        else:
            print('Missing input!')
    return number
```

```
def fxCircle(radius, xCoords):
    rSquared = radius * radius
    return [sqrt(rSquared-min(x**2, rSquared))
            for x in xCoords]

def areaCircle(fx, radius, intervals):
    xCoords = [i * radius/intervals for i in
               range(intervals+1)]
    yCoords = fx(radius, xCoords)
    area = 0.
    for i in range(intervals):
        area += yCoords[i] + yCoords[i+1]
    #area = 4. * area / 2 * deltaX
    area *= 2 * radius/intervals
    return area
```

```
def main():
    print('\n' + '-' * 80)
    print('\nEnter 0 at any prompt to terminate the program.')
    while True:
        radius = float(getPosNum(
                  'Enter the radius in cm (>0): '))
        if radius == 0.0:
            break
        intervals = int(getPosNum(
                 'Enter the number of intervals (>0): '))
        if intervals == 0:
            break
        area = areaCircle(fxCircle, radius, intervals)
        actual = pi * radius * radius
        error = abs(actual - area)
        print("""
Approximate area of circle is %.14e cm^2
     Actual area of circle is %.14e cm^2
     The error in the area is %e cm^2""" % (area, actual, error))
    displayTerminationMessage()

main()
```

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

Enter 0 at any prompt to terminate the program.

Enter the radius in cm (>0): 1

Enter the number of intervals (>0): 1000

Approximate area of circle is 3.14155546691103e+00 cm^2
     Actual area of circle is 3.14159265358979e+00 cm^2
     The error in the area is 3.718668e-05 cm^2

Enter the radius in cm (>0): 1

Enter the number of intervals (>0): 100000

Approximate area of circle is 3.14159261640195e+00 cm^2
     Actual area of circle is 3.14159265358979e+00 cm^2
     The error in the area is 3.718784e-08 cm^2
```

```
Enter the radius in cm (>0): 1

Enter the number of intervals (>0): 10000000

Approximate area of circle is 3.14159265355226e+00 cm^2
     Actual area of circle is 3.14159265358979e+00 cm^2
     The error in the area is 3.753220e-11 cm^2

Enter the radius in cm (>0): 0

Programmed by Stew Dent.
Date: Thu May 31 16:43:15 2018
End of processing.
```

- Use Simpson's rule to find the area under a curve.

$$\int_a^b f(x)dx \simeq \frac{b-a}{3n}\left(f(a) + f(b) + 4\sum_{i=1}^{n/2} f(a+(2i-1)h) + 2\sum_{i=1}^{n/2-1} f(a+2ih) \right)$$

- f(x) is the function for which the area under the curve is to be found, y=f(x)
- a is the lower bound for x, b is the upper bound for x, h is (b-a)/n where n is the number of intervals.
- (b-a)/3n = h/3

- The number of intervals must a positive even integer.
- To find the area of a circle whose radius is r evaluate:

$$f(x) = \sqrt{(r^2 - x^2)}$$

- The values of x must in the range x=0 to x=r
- Therefore a=0 and b=r.
- The step size is h = r / n, where n is the number of intervals.
- The approximate area of the circle is 4 times the area under the curve for x=a to x=b .

```
from time import ctime
from math import sqrt, pi

def  getPosNumber(prompt):
    while True:
        number = input(prompt).strip()
        if number != '':
            try:
                number = eval(number, {}, {})
            except:
                print('Invalid input!')
            else:
                if type(number) is int or type(number) is float:
                    if number >= 0:
                        break
                    else:
                        print(number, 'is not a positive number')
                else:
                    print(number, 'is not an integer!')
        else:
            print('Missing input!')
    return number
```

```
def f(rSquared, x): # for a circle
    return sqrt(rSquared-min(x**2, rSquared))

def simpsonCircle(a, b, intervals):
    # a - the first x value, must be a real number
    # b - the last x value, must be a real number
    # intervals - the number of intervals, must be an even number
    # rSquared - the square of the radius of the circle
    # h - the width of equally spaced intervals
    # firstSum - the first sum in Simpson's rule
    # secondSum - the second sum in Simpson's rule
    # i - a general purpose index
    rSquared = b*b
    h = (b - a) / intervals
    firstSum = 0.
    for i in range(1, intervals//2 + 1):
        firstSum += f(rSquared, a + (2*i - 1)*h)
    secondSum = 0.
    for i in range(1, intervals//2):
        secondSum += f(rSquared, a + 2*i*h)
    area = h/3. * (f(rSquared, a) + 4 * firstSum + 2 * secondSum +
           f(rSquared, b))
    return area * 4  # area was for 1/4 of a circle
```

```
def main():
    print('\n' + '-' * 80)
    print('\nEnter 0 at any prompt to terminate the program.')
    while True:
        intervals = int(getPosNumber('Enter the number of intervals (even > 0):
'))
        while (intervals % 2 != 0):
            print('The number of intervals, %d, must be even!' % intervals)
            intervals = int(getPosNumber(
                            'Enter the number of intervals (even > 0): '))
        if intervals == 0:
            break
        radius = float(getPosNumber('Enter the radius of the circle in cm: '))
        if radius == 0.0:
            break
        area = simpsonCircle(0.0, radius, intervals)
        actual = pi * radius * radius
        error = abs(actual - area)
        print('\nArea is from x=%.2f to x=%.2f' % (0.0, radius))
        print('Approximate area of circle is %.14e cm^2' % (area))
        print('     Actual area of circle is %.14e cm^2' % actual)
        print('Error in the approximation is %e cm^2' % error)
    displayTerminationMessage()

main()
```

```
--------------------------------------------------------------------------------
Enter 0 at any prompt to terminate the program.

Enter the number of intervals (even > 0): 1000

Enter the radius of the circle in cm: 2.25

Area is from x=0.00 to x=2.25
Approximate area of circle is 1.59042392842083e+01 cm^2
     Actual area of circle is 1.59043128087983e+01 cm^2
Error in the approximation is 7.352459e-05 cm^2

Enter the number of intervals (even > 0): 10000000

Enter the radius of the circle in cm: 2.25
Area is from x=0.00 to x=2.25
Approximate area of circle is 1.59043128087246e+01 cm^2
     Actual area of circle is 1.59043128087983e+01 cm^2
Error in the approximation is 7.375434e-11 cm^2
Enter the number of intervals (even > 0): 0
Programmed by Stew Dent.
Date: Thu May 31 17:22:50 2018.
End of processing.
```