Spring, 2015 CIT 590

Programming Languages and Techniques

Homework 2

Goal

In this assignment we will use several number theoretic definitions and learn to write functions to check whether a number from 1 to 10000 satisfies those properties.

The main goal of this assignment is to write modular code - code with a lot of small functions and a high amount of code reuse.

The definitions we will use are

- Prime number - A number with exactly 2 factors. The way to check whether one number divides another is to use the % operation. Try out a few examples on the prompt. 4 % 2 should return 0. 17 % 5 should give you 2 (do not worry about negative numbers, numbers like 0.5 etc)

- Composite number - A number with more than 2 factors. 1 is neither prime nor composite.

- Perfect number - A number is said to be perfect if it is equal to the sum of all its factors (for obvious reasons the list of factors being considered does not include the number itself). 6 = 3 + 2 + 1, hence 6 is perfect. 28 (1, 2, 4, 7, 14)is another example.

- Abundant number - A number is considered to be abundant if the sum of its divisors is greater than the number itself. For instance 12 is abundant since $1+2+3+4+6 = 16 > 12$

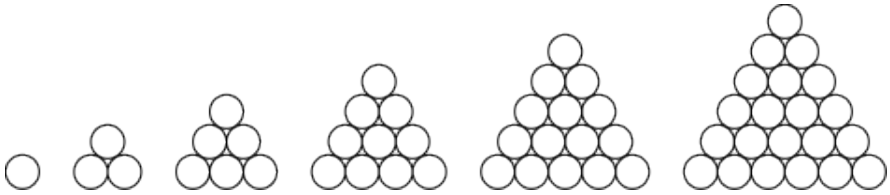
- Triangular number -

  The triangular number T_n is a figurate number that can be represented in the form of a triangular grid of points where the first row contains a single element and each subsequent row contains one more element than the previous one

  (see http://mathworld.wolfram.com/TriangularNumber.html for more interesting mathematical information.)

  For the purposes of this assignment, we just use the fact that the nth triangular number can be found by using this formula

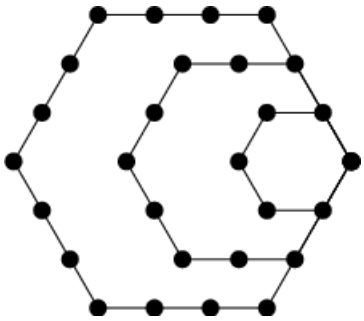
$$\frac{n(n+1)}{2}$$

Here is a useful visualization 

taken from the following website

http://thinkmath.edc.org/resource/triangular-numbers

- Pentagonal number - The nth pentagonal number is one-third of the $3n-1^{th}$ triangular number. (Hint: Think about how you can reuse code for checking for a pentagonal number)
- Hexagonal number - A number that can be expressed in the form $2n^2 - n$. The nth hexagonal number will be the number of points in a hexagon with n regularly spaced points on a side as shown in the figure below.



The actual program

Write a Python program with at least the following functions

- isPrime(x) - function that returns whether or not the given number x is prime. This function returns boolean.
- isComposite(x) - function that returns whether or not the given number x is composite. This function returns boolean.
- isPerfect(x) - code that returns whether or not the given number x is perfect. This function returns boolean.

- isAbundant(x) - code that returns whether or not the given number x is abundant. This function returns boolean.
- isTriangular(x) - code that returns whether or not the given number x is triangular. This function returns boolean.
- isPentagonal(x) - code that returns whether or not the given number x is pentagonal. This function returns a boolean.
- isHexagonal(x) - code that returns whether or not the given number x is Hexagonal. This function returns boolean.

and one special function

main():

a function that repeatedly asks a user for a number between 1 and 10000 and prints out whether or not the number is prime, perfect, abundant If the user inputs a number that is outside the range of acceptable input, your program should handle that by printing a useful error message. Else it should print something like this

```
6 is not prime, is perfect, is not abundant
```

The program should be quit if the user enters -1.

At the end of your Python program (after all your function definitions), insert the following lines:

```
if __name__ == "__main__":
   main()
```

Here's what this does. If you are in the IDLE window containing your program, and you click F5 (or choose Run - Run module) from the menu, IDLE will automatically run your main function. If you are in the IDLE window containing your test cases (supplied) and click F5 or use the menu equivalent, it will run the tests and tell you the results.

Helper functions

The big goal in this assignment is to introduce you to the concept of code reuse. So look at all the functions that we are asking you to write. Are there common aspects to those functions? If so, separate the common pieces of code into helper functions.

For example, a number is triangular if $n^2 + n - 2 \times number = 0$ has integer solutions. Similarly, a number is hexagonal if $2n^2 - n - number = 0$ has integer solutions. So, it makes sense to write a helper function that takes in the coefficients of a quadratic equation and checks for integer solutions. So write a helper function that given a, b and c of a quadratic equation $ax^2 + bx + c$ determines whether the equation has integer solutions. Then use that helper function for both triangular and hexagonal number determination.

Feel free to use an online resource if you have forgotten the formula to solve quadratic equations. Or, if you show up to the recitation, the TAs will gladly remind you of it.

To check whether a number is an integer or not use the following

```
def isInt(x):
    ''' checks if x is an integer or not. returns boolean '''
    return x == int(x)
```

What to submit

Submit one single file called **numberTheory.py**

Make sure every function has a docString comment (triple quoted brief explanation of what the function does).

Also, please comment your code. Since you are writing some of your own functions, it might not be completely obvious to us what your code is trying to do. Therefore, your comments become important!