Homework 5
Iterators and Generators
Out 10/26 – Due 11/10

Exercise 1. Reverse iterator (30 pts)
Write an iterator class ReverseIter, that takes a list and iterates it from the reverse direction. Save your code as hw5_firstname_lastname_ex_1.py. An example could be like this (hint – use a list as the input argument of your constructor):

```
>>> it = ReverseIter([1, 2, 3, 4])
>>> next(it)
4
>>> next(it)
3
>>> next(it)
2
>>> next(it)
1
>>> next(it)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

Exercise 2. Pythogorian triplets (40 pts)
Use a ***generator comprehension expression*** to find first 10 (or any n) pythogorian triplets. A triplet (x, y, z) is called a pythogorian triplet if x*x + y*y == z*z, where x/y/z are all integers.

You may want to use the integers() and take(n, seq) functions explained in class. Implement the generator with name **pyt**, and then you could do something like this:

```
print(take(10, pyt))
```

And the output would be (notice all x/y/z are in a tuple):

```
[(3, 4, 5), (6, 8, 10), (5, 12, 13), (9, 12, 15), (8, 15, 17), (12, 16, 20), (15, 20, 25), (7, 24, 25), (10, 24, 26), (20, 21, 29)]
```

Save your code as hw5_firstname_lastname_ex_2.py.

Exercise 3. The Generator Version of range() (30 pts)
The range() function creates a sequence. For very large sequences, this consumes a lot of memory. You can write a version of range which does not create the entire sequence, but instead yields the individual values. Using a generator will have the same effect as iterating through a sequence, but won't consume as much memory.

Define a generator, genrange(), which generates the same sequence of values as range(), without creating a list object.

The original range() function is used as follows:

- range(stop)

- range(start, stop)
- range(start, stop, step)

For simplicity, the genrange() can be used as follows:

- genrange(stop)
- genrange(stop, start)
- genrange(stop, start, step)

where start and step are optional arguments.

Save your code as hw5_firstname_lastname_ex_3.py.

Submit your code files in **a zipped archive named hw5_firstname_lastname.zip**. Comment everything so we know you wrote the code! On top of your files write this multiline comment with your information:

```
"""
Homework 5, Exercise 1 (or 2…)
Name
Date
Description of your program.
"""
```