Homework 4

OOP, Iterators and Generators

Given February 12th – Due February 24th

Exercise 1. Rectangle, Circle and Square

Write three Python classes named Rectangle constructed by a length and width, a Circle constructed by a radius and a Square constructed by a side length. Both classes should have the methods that compute:

- The area
- The diagonal
- The perimeter

Use as much abstraction as you can. At the end of the file, use those classes to calculate the perimeter of a circle with radius the half of the diagonal of a rectangle with length 20 and width 10.

Exercise 2. Reverse iterator

Write an iterator class *ReverseIter*, that takes a list and iterates it from the reverse direction. An example could be like this:

```
>>> 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 3. Pythogorian triplets

Use a generator comprehension expression to find first 10 (or any n) pythogorian triplets. A triplet (x, y, z) is called pythogorian triplet if x*x + y*y == z*z.

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

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

And the output would be:

```
[(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)]
```

Exercise 4. The Generator Version of range()

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 range() functions 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.

Submit your code files as *hm4_name_surname_ex_num.py*, where *num* is the exercise number 1, 2, etc. Comment everything so we know you wrote the code! On top of your files write this multiline comment with your information:

"""

Homework 4, Exercise 1 (or 2…)

Name

Date

Description of your program.

"""