1.How many seconds are in an hour? Use the interactive interpreter as a calculator and multiply the number of seconds in a minute (60) by the number of minutes in an hour (also 60).

sol. 60

**Ans.** There are 3600 seconds in an hour.

I can use the Python interpreter as a calculator to verify this by typing:

**60 \* 60**

The output should be:

**3600**

2. Assign the result from the previous task (seconds in an hour) to a variable called seconds\_per\_hour.

**Ans.** **seconds\_per\_hour = 60 \* 60**

This assigns the value of 3600 (the number of seconds in an hour) to the variable seconds\_per\_hour.

3. How many seconds do you think there are in a day? Make use of the variables seconds per hour and minutes per hour.

**Ans.** There are 86400 seconds in a day.

Seconds\_per\_hour = 60\*60

Minutes\_per\_hour = 60

There are 24 hours a day.

So, 24\* Seconds\_per\_hour = 86400 seconds

4. Calculate seconds per day again, but this time save the result in a variable called seconds\_per\_day

**Ans.** Seconds\_per\_hour = 60\*60

seconds\_per\_day = 24\*Seconds\_per\_hour

seconds\_per\_day = 86400

5. Divide seconds\_per\_day by seconds\_per\_hour. Use floating-point (/) division.

**Ans.** seconds\_per\_day/Seconds\_per\_hour

The output will be:

**24.0**

6. Divide seconds\_per\_day by seconds\_per\_hour, using integer (//) division. Did this number agree with the floating-point value from the previous question, aside from the final .0?

**Ans.** seconds\_per\_day//Seconds\_per\_hour

The output will be:

**24**

Yes, this number agree with the floating-point value from the previous question, aside from the final .0

7. Write a generator, genPrimes, that returns the sequence of prime numbers on successive calls to its next() method: 2, 3, 5, 7, 11, ...

**Ans.**

**def genPrimes():**

**yield 2**

**current\_num = 3**

**primes = [2]**

**while True:**

**is\_prime = True**

**for prime in primes:**

**if current\_num % prime == 0:**

**is\_prime = False**

**break**

**if is\_prime:**

**yield current\_num**

**primes.append(current\_num)**

**current\_num += 2**

The output will be:

**prime\_generator = genPrimes()**

**for i in range(10):**

**print(next(prime\_generator))**

2

3

5

7

11

13

17

19

23

29