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

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

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

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

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?

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:  
Here are the answers to your questions:**

**1. To calculate the number of seconds in an hour:**

**```python**

**60 \* 60**

**```**

**This equals 3600 seconds.**

**2. Assign the result to a variable called `seconds\_per\_hour`:**

**```python**

**seconds\_per\_hour = 60 \* 60**

**```**

**3. To calculate the number of seconds in a day using the variables `seconds\_per\_hour` and `minutes\_per\_hour` (assuming 24 hours in a day):**

**```python**

**seconds\_per\_hour \* 24**

**```**

**This equals 86400 seconds.**

**4. Calculate `seconds\_per\_day` and save the result in a variable:**

**```python**

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

**```**

**5. Divide `seconds\_per\_day` by `seconds\_per\_hour` using floating-point division:**

**```python**

**seconds\_per\_day / seconds\_per\_hour**

**```**

**This equals 24.0, as expected.**

**6. Divide `seconds\_per\_day` by `seconds\_per\_hour` using integer division:**

**```python**

**seconds\_per\_day // seconds\_per\_hour**

**```**

**This also equals 24. The result is the same as the floating-point division, except that it's represented as an integer.**

**7. Below is a generator function `genPrimes` that returns the sequence of prime numbers on successive calls to its `next()` method:**

**```python**

**def genPrimes():**

**primes = []**

**n = 2**

**while True:**

**if all(n % p != 0 for p in primes):**

**primes.append(n)**

**yield n**

**n += 1**

**# Usage example:**

**prime\_generator = genPrimes()**

**print(next(prime\_generator)) # Output: 2**

**print(next(prime\_generator)) # Output: 3**

**print(next(prime\_generator)) # Output: 5**

**print(next(prime\_generator)) # Output: 7**

**print(next(prime\_generator)) # Output: 11**

**```**

**This generator function generates prime numbers indefinitely. Each time `next()` is called on the generator object, it returns the next prime number in the sequence.**