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).

**Answer: 60 \* 60**

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

**Answer: seconds\_per\_hour = 60 \*60**

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

**Answer:**

**Seconds\_per\_hour = 60**

**Minutes\_per\_hour = 60**

**Hours\_in\_day = 24**

**Seconds\_per\_hour \* Minutes\_per\_hour \* Hours\_in\_day**

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

**Answer:**

**seconds\_per\_day =** **Seconds\_per\_hour \* Minutes\_per\_hour \* Hours\_in\_day**

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

**Answer: seconds\_per\_day/ seconds\_per\_hour 🡪 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?

**Answer: seconds\_per\_day// seconds\_per\_hour 🡪 24**

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

**Answer:**

**def genPrime():**

**primes = [ 2, 3, 5, 7, 11 ]**

**def isPrime(n):**

**if n in primes:**

**return True**

**for elem in primes:**

**if n % elem == 0:**

**return False**

**primes.append(n)**

**return True**

**num = 1**

**while True:**

**num += 1**

**if isPrime(num):**

**next = num**

**yield next**

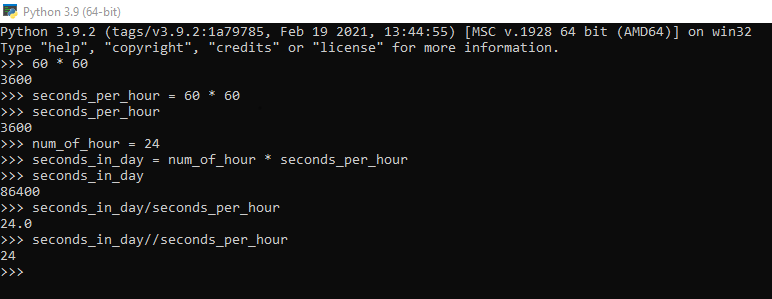
**num = next**

**prime= genPrime()**

**for i in range(100):**

**print(prime.\_\_next\_\_())**

**Screenshot for above answers:**

****