**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 \* 60

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

**Sol :** seconds\_per\_hour = 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.**

**Sol :** minutes\_per\_hour = seconds\_per\_hour \* 24

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

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

seconds\_per\_day

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

**Sol :** seconds\_per\_day / seconds\_per\_hour

**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 :** Yes, Agree

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

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

|  |  |
| --- | --- |
| def genPrimes(): | |
|  | primes = [] |
|  | n = 2 |
|  | last = n |
|  |  |
|  | while True: |
|  | for i in primes: |
|  | if n % i == 0: |
|  | n += 1 |
|  | break |
|  |  |
|  | else: |
|  | primes.append(n) |
|  | last = n |
|  | n += 1 |
|  | yield last |

**OR**

**def** genPrimes():

primes = [] *# primes generated so far*

last = 1 *# last number tried*

**while** True:

last += 1

**for** p in primes:

**if** last % p == 0:

**break**

**else**:

primes.append(last)

**yield** last

p = genPrimes()