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

There are 3600 seconds in an hour. (60 seconds/minute \* 60 minutes/hour = 3600 seconds/hour)

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

`seconds\_per\_hour = 3600`

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

There are 86400 seconds in a day. (60 seconds/minute \* 60 minutes/hour \* 24 hours/day = 86400 seconds/day)

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

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

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

`seconds\_per\_day / seconds\_per\_hour` results in `24.0`, since floating-point division is used.

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?

`seconds\_per\_day // seconds\_per\_hour` also results in `24`. This is expected since integer division truncates any decimal part of the result, so the final `.0` is removed.

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

Here's an implementation of the genPrimes generator:

```

def genPrimes():

# initialize the list of primes with the first prime number, 2

primes = [2]

# start the candidate at 3

candidate = 3

while True:

# check if the candidate is prime

for prime in primes:

if candidate % prime == 0:

# if the candidate is divisible by any prime, it's not prime

break

else:

# if the candidate wasn't divisible by any primes, it's prime

primes.append(candidate)

yield candidate

# move on to the next candidate

candidate += 1

```