#### 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).

I/P:

60**\***60

O/P:

3600

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

I/P:

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

print(seconds\_per\_hour)

O/P:

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.

I/P:

minutes\_per\_hour **=** 60

print(seconds\_per\_hour**\***24)

O/P:

86400

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

I/P:

seconds\_per\_day **=** 24**\***60**\***60

print(seconds\_per\_day)

O/P:

86400

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

I/P:

print(seconds\_per\_day**/**seconds\_per\_hour)

O/P:

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?

I/P:

print(seconds\_per\_day**//**seconds\_per\_hour, end**=**'')

print(' -> yes this values agree with the floating point value from the previous question')

O/P:

24 -> yes this values agree with the floating point value from the previous question

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

I/P:

**def** genPrimes():

n **=** 0

**while** **True**:

**if** n **==** 2 **or** n **==** 3 :

**yield** n

**elif** ((n**-**1)**%6** == 0 or (n+1)%6 == 0) and n !=1:

**yield** n

n **=** n**+**1

output **=** genPrimes()

**for** ele **in** range(5):

print(next(output))

O/P:

2

3

5

7

11