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

The correct solution is to multiply the number of seconds in a minute (60) by the number of minutes in an hour (60):

60 seconds/minute \* 60 minutes/hour = 3600 seconds/hour

Therefore, there are 3600 seconds in an hour.

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

To assign the result from the previous task (seconds in an hour) to a variable called seconds\_per\_hour, you can use the following code:

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.

To calculate the number of seconds in a day, you can use the variables seconds\_per\_hour and the number of hours in a day (24). Here's the calculation:

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

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

number of seconds in a day and save the result in a variable called seconds\_per\_day, you can use the previously calculated value of seconds\_per\_hour:

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

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

To divide seconds\_per\_day by seconds\_per\_hour using floating-point division, you can simply use the / operator. Here's the code:

result = 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?

To divide seconds\_per\_day by seconds\_per\_hour using integer division, you can use the // operator. Here's the code:

result = 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, ...

To generate prime numbers using a generator, you can implement the genPrimes generator as follows:

def genPrimes():

primes = [] # List to store prime numbers

# Helper function to check if a number is prime

def isPrime(num):

for prime in primes:

if num % prime == 0:

return False

return True

# Yield prime numbers on successive calls

num = 2 # Start with the first prime number

while True:

if isPrime(num):

primes.append(num)

yield num

num += 1