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

Ans : To calculate the number of seconds in an hour, you can multiply the number of seconds in a minute (60) by the number of minutes in an hour (60). Using the interactive interpreter as a calculator, you can simply enter the following:

60 \* 60

The result will be:

3600

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.

Ans : In Python, you can assign the result of a calculation to a variable using the assignment operator (=). To assign the result of the previous task (3600 seconds in an hour) to a variable called seconds\_per\_hour, you can write the following code:

pythonCopy code

seconds\_per\_hour = 60 \* 60

Now, the variable **seconds\_per\_hour** holds the value 3600, which represents the number of seconds in an hour.

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

Ans : To calculate the number of seconds in a day, you can multiply the number of seconds per hour by the number of hours in a day. Since there are 24 hours in a day, you can use the variables seconds\_per\_hour and hours\_per\_day to perform the calculation. Here's the code:

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

The variable **seconds\_per\_day** now holds the result of the calculation, which is the number of seconds in a day.

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

Ans : Apologies for the confusion. Here's the corrected code to calculate the number of seconds in a day and save the result in a variable called seconds\_per\_day:

pythonCopy code

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

Now, the variable **seconds\_per\_day** holds the value 86400, which represents the number of seconds in a day.

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

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

seconds\_per\_day = 86400 seconds\_per\_hour = 3600 result = seconds\_per\_day / seconds\_per\_hour

The variable **result** will hold the value **24.0** since there are 24 hours in a day. The use of floating-point division (**/**) ensures that the result is a decimal number rather than an integer.

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

pythonCopy code

seconds\_per\_day = 86400 seconds\_per\_hour = 3600 result = seconds\_per\_day // seconds\_per\_hour

The variable **result** will hold the value **24** since integer division (**//**) returns the quotient as an integer, discarding any fractional part. In this case, the result is the same as the floating-point division result from the previous question, **24.0**, except for the absence of the decimal point and the **.0**.

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

Ans : Sure! Here's an implementation of the genPrimes generator that returns the sequence of prime numbers on successive calls to its next() method:

def genPrimes(): # Initialize the prime number sequence primes = [2] yield 2 # Start checking for primes from 3 num = 3 while True: is\_prime = True # Check if num is divisible by any prime number for prime in primes: if num % prime == 0: is\_prime = False break # If num is prime, yield it and add to the primes list if is\_prime: primes.append(num) yield num # Move to the next number for checking num += 2

You can use the **genPrimes** generator as follows:

prime\_generator = genPrimes() # Print the first few prime numbers print(next(prime\_generator)) # Output: 2 print(next(prime\_generator)) # Output: 3 print(next(prime\_generator)) # Output: 5 print(next(prime\_generator)) # Output: 7 print(next(prime\_generator)) # Output: 11

Each call to **next(prime\_generator)** will return the next prime number in the sequence.