**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).**

Ans-To calculate the number of seconds in an hour, we can multiply the number of seconds in a minute (60) by the number of minutes in an hour (60).

Python Example: seconds\_per\_hour = 60 \* 60

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

Ans-The result from the previous calculation, which is the number of seconds in an hour, can be assigned to a variable called ‘seconds\_per\_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 estimate the number of seconds in a day, we can multiply the number of seconds per hour by the number of hours in a day (24).

Python Example: 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**

Ans- We can calculate the number of seconds per day again and save the result in a variable called seconds\_per\_day.

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

Ans- To find out how many times the number of seconds per hour fits into the number of seconds per day, we can divide seconds\_per\_day by seconds\_per\_hour using floating-point division(/).

floating\_point\_division = 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- To calculate the integer division between seconds\_per\_day and seconds\_per\_hour, you can use integer division (//):

integer\_division = seconds\_per\_day // seconds\_per\_hour

The result of the integer division should agree with the floating-point value from the previous question, except for the final ".0" that is present in the floating-point division result.

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

def genPrimes():

prime\_numbers = []

num = 2

while True:

is\_prime = True

for prime in prime\_numbers:

if num % prime == 0:

is\_prime = False

break

if is\_prime:

prime\_numbers.append(num)

yield num

num += 1

# Usage example:

prime\_generator = genPrimes()

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

# and so on...