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

2. Assign the result from the previous task (seconds in an hour) 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.

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

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

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?

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

ChatGPT

Sure, let's go through each step:

python

Copy code

# Task 1

seconds\_in\_minute = 60

minutes\_in\_hour = 60

seconds\_per\_hour = seconds\_in\_minute \* minutes\_in\_hour

print(seconds\_per\_hour) # Output: 3600

# Task 3

hours\_in\_day = 24

seconds\_per\_day = seconds\_per\_hour \* minutes\_in\_hour \* hours\_in\_day

print(seconds\_per\_day) # Output: 86400

# Task 4

seconds\_per\_day = seconds\_per\_hour \* minutes\_in\_hour \* hours\_in\_day

print(seconds\_per\_day) # Output: 86400

# Task 5

floating\_point\_result = seconds\_per\_day / seconds\_per\_hour

print(floating\_point\_result) # Output: 24.0

# Task 6

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

print(integer\_result) # Output: 24

# Task 7

def genPrimes():

primes = [2]

current\_number = 3

while True:

is\_prime = all(current\_number % prime != 0 for prime in primes)

if is\_prime:

primes.append(current\_number)

yield current\_number

current\_number += 2 # Optimization: Check only odd numbers (except 2)

# Example of using the generator

primes\_generator = genPrimes()

for \_ in range(5): # Print the first 5 prime numbers

print(next(primes\_generator))

In the above code:

Task 1 calculates the number of seconds in an hour.

Task 3 calculates the number of seconds in a day using the variables from Task 1 and additional variables.

Task 4 recalculates the number of seconds in a day, without changing the result.

Task 5 performs floating-point division to find out how many hours are in a day.

Task 6 performs integer division to find out the same but without the decimal part.

Task 7 defines a generator genPrimes that yields prime numbers on successive calls to its next() method. It starts with 2 and generates odd prime numbers thereafter. The example at the end shows how to use this generator to get the first 5 prime numbers.