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, ...

Answers:

1. 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):

60 \* 60 = 3600

Therefore, there are 3600 seconds in an hour.

1. We can assign the result from the previous task to a variable called seconds\_per\_hour like this:

seconds\_per\_hour = 3600

1. To calculate the number of seconds in a day, we can multiply the number of seconds in an hour (3600) by the number of hours in a day (24):

3600 \* 24 = 86400

Therefore, there are 86400 seconds in a day.

1. We can assign the result from the previous task to a variable called seconds\_per\_day like this:

seconds\_per\_day = 86400

1. To calculate the number of hours in a day, we can divide seconds\_per\_day by seconds\_per\_hour using floating-point division (/):

seconds\_per\_day / seconds\_per\_hour = 24.0

Therefore, there are 24.0 hours in a day.

1. We can also divide seconds\_per\_day by seconds\_per\_hour using integer division (//):

seconds\_per\_day // seconds\_per\_hour = 24

This result agrees with the floating-point value from the previous question, except that it is an integer value without a decimal point.

1. Here is a generator function genPrimes that returns the sequence of prime numbers on successive calls to its next() method:

def genPrimes():

"""

A generator function that yields prime numbers on successive calls to its next() method.

"""

primes = [] # list to store prime numbers

num = 2 # start with the first prime number

while True:

is\_prime = True

# check if the current number is divisible by any of the previous prime numbers

for prime in primes:

if num % prime == 0:

is\_prime = False

break

if is\_prime:

primes.append(num) # add the current number to the list of primes

yield num # yield the current number as the next prime number

num += 1 # move on to the next number

We can use this generator as follows:

primes = genPrimes() # create a generator object

print(next(primes)) # output: 2

print(next(primes)) # output: 3

print(next(primes)) # output: 5

print(next(primes)) # output: 7

print(next(primes)) # output: 11

# and so on...