**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**

seconds\_in\_a\_minute = 60

minutes\_in\_an\_hour = 60

seconds\_in\_an\_hour = seconds\_in\_a\_minute \* minutes\_in\_an\_hour

seconds\_in\_an\_hour =3600

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

seconds\_per\_hour = 3600

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

minutes\_in\_an\_hour = 60

hours\_in\_a\_day = 24

seconds\_in\_a\_day = seconds\_per\_min \* minutes\_in\_an\_hour \* hours\_in\_a\_day

seconds\_in\_a\_day=86400

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

minutes\_in\_an\_hour = 60

hours\_in\_a\_day = 24

seconds\_per\_day = seconds\_per\_min \* minutes\_in\_an\_hour \* hours\_in\_a\_day

seconds\_per\_day=86400

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

86400/3600=24.0

**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?**

86400//3600=24

This agrees 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, ...**

A generator is a special type of function in Python that allows us to iterate over a sequence of values without storing them all in memory at once. It generates values on the fly as they are needed, making it memory-efficient and suitable for generating large or infinite sequences.

def genPrimes():

primes = []

last = 1

while True:

last += 1

for p in primes:

if last % p == 0:

break

else:

primes.append(last)

yield last

prime\_generator = genPrimes()

print(next(prime\_generator ))

print(next(prime\_generator ))

print(next(prime\_generator ))

print(next(prime\_generator ))