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.

seconds\_in\_a\_minute = 60

minutes\_in\_an\_hour = 60

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

print(seconds\_in\_an\_hour)

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

Ans.

seconds\_per\_day = 86400

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.

hours\_in\_a\_day = 24

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

print(seconds\_in\_a\_day)

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

Ans.

seconds\_per\_day = 86400

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

Ans.

result\_floating = seconds\_per\_day / seconds\_per\_hour

print(result\_floating)

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.

result\_integer = seconds\_per\_day // seconds\_per\_hour

print(result\_integer)

Yes, the integer division result agrees with the floating-point value from the previous question, except for 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, …

Ans.

def is\_prime(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

def genPrimes():

num = 2

while True:

if is\_prime(num):

yield num

num += 1

prime\_generator = genPrimes()