**1. How many seconds are in an hour?**

You can calculate the seconds in an hour by multiplying the number of seconds in a minute (60) by the number of minutes in an hour (60):

60 \* 60 # Result: 3600 seconds

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

seconds\_per\_hour = 60 \* 60 # 3600 seconds

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

There are 24 hours in a day, so to calculate the seconds in a day, multiply seconds\_per\_hour by 24:

seconds\_per\_day = seconds\_per\_hour \* 24 # 3600 \* 24 = 86400 seconds

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

seconds\_per\_day = seconds\_per\_hour \* 24 # Result: 86400 seconds

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

seconds\_per\_day / seconds\_per\_hour # Result: 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?**

seconds\_per\_day // seconds\_per\_hour # Result: 24

Yes, this number agrees with the floating-point value (24.0), except for the decimal part.

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

Here's a simple generator function to yield prime numbers:

def genPrimes():

primes = []

n = 2

while True:

if all(n % p != 0 for p in primes):

primes.append(n)

yield n

n += 1

This generator will yield prime numbers on each call to next(). You can test it as follows:

prime\_gen = genPrimes()

# Get the first 5 primes:

for \_ in range(5):

print(next(prime\_gen)) # Output: 2, 3, 5, 7, 11