 **Seconds in an hour:**

* There are 60 seconds in a minute and 60 minutes in an hour. So, the total seconds in an hour is:

python

Copy code

seconds\_in\_a\_minute = 60

minutes\_in\_an\_hour = 60

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

print(seconds\_per\_hour) # Outputs 3600

 **Assign the result to seconds\_per\_hour:**

* This was already done in the previous step. So:

python

Copy code

seconds\_per\_hour = 3600

 **Seconds in a day:**

* There are 24 hours in a day, so:

python

Copy code

hours\_per\_day = 24

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

print(seconds\_per\_day) # Outputs 86400

 **Calculate seconds\_per\_day and save the result:**

python

Copy code

hours\_per\_day = 24

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

print(seconds\_per\_day) # Outputs 86400

 **Divide seconds\_per\_day by seconds\_per\_hour using floating-point division:**

python

Copy code

division\_floating\_point = seconds\_per\_day / seconds\_per\_hour

print(division\_floating\_point) # Outputs 24.0

 **Divide seconds\_per\_day by seconds\_per\_hour using integer division:**

python

Copy code

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

print(division\_integer) # Outputs 24

* The integer division will not include the decimal point, so it will be 24, which agrees with the floating-point result, except for the .0.

 **Write a generator for prime numbers:**

Here is a simple prime number generator using the Sieve of Eratosthenes approach:

python

Copy code

def genPrimes():

D = {}

q = 2

while True:

if q not in D:

# q is a new prime number

yield q

D[q \* q] = [q]

else:

# Move each factor to its next multiple

for p in D[q]:

D.setdefault(p + q, []).append(p)

del D[q]

q += 1

# Example usage:

prime\_generator = genPrimes()

for \_ in range(10): # Get the first 10 primes

print(next(prime\_generator))

* This generator yields prime numbers in sequence: 2, 3, 5, 7, 11, and so on.