**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).**

**Answer :**

To calculate the number of seconds in an hour, you can multiply the number of seconds in a minute (60) by the number of minutes in an hour (60) using the interactive interpreter. Here's the calculation:

seconds\_in\_minute = 60

minutes\_in\_hour = 60

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

seconds\_in\_hour

The result is:

3600

sol. 60

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

**Answer :**

To assign the result from the previous calculation (3600 seconds in an hour) to a variable called seconds\_per\_hour, you can do the following:

seconds\_per\_hour = 3600

Now the variable seconds\_per\_hour holds the value 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.

**Answer :**

To calculate the number of seconds in a day, we can make use of the variables seconds\_per\_hour (which we previously assigned as 3600 seconds) and the number of hours in a day (24 hours).

Since there are 60 minutes in an hour, and 60 seconds in a minute, we can multiply seconds\_per\_hour by the number of hours in a day to obtain the total number of seconds in a day. Here's the calculation:

hours\_per\_day = 24

seconds\_per\_minute = 60

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

seconds\_per\_day

The result will be:

86400

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

**Answer :**

The number of seconds in a day and save the result in a variable called seconds\_per\_day, you can use the following code:

seconds\_per\_day = seconds\_per\_hour \* 60 \* 24

This will multiply the value of seconds\_per\_hour by 60 (minutes per hour) and 24 (hours per day) to get the total number of seconds in a day. The result will be stored in the variable seconds\_per\_day.

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

**Answer :**

To divide seconds\_per\_day by seconds\_per\_hour using floating-point division, you can use the / operator. Here's the calculation:

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

result

The result will be:

24.0

Therefore, dividing the total number of seconds in a day by the number of seconds in an hour yields 24.0, indicating that there are 24 hours in a day.

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?

**Answer :**

To divide seconds\_per\_day by seconds\_per\_hour using integer division, you can use the // operator. Here's the calculation:

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

result

The result will be:

24

When using integer division (//), the fractional part is discarded, resulting in an integer value. In this case, the result is 24, which agrees with the floating-point value from the previous question, aside from the final .0. Therefore, both the floating-point and integer division yield the same value of 24, confirming that there are 24 hours in a day.

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

**Answer :**

This is the example of a generator function called genPrimes that generates the sequence of prime numbers on successive calls to its next() method:

def genPrimes():

yield 2 # 2 is the first prime number

primes = [2] # Store the prime numbers generated so far

num = 3 # Start checking for primes from 3

while True:

is\_prime = True

for prime in primes:

if num % prime == 0:

is\_prime = False

break

if is\_prime:

yield num

primes.append(num)

num += 2 # Only check odd numbers for primality

# Usage example:

primes\_generator = genPrimes()

# Generate and print the first few prime numbers

print(next(primes\_generator)) # Output: 2

print(next(primes\_generator)) # Output: 3

print(next(primes\_generator)) # Output: 5

print(next(primes\_generator)) # Output: 7

print(next(primes\_generator)) # Output: 11

The genPrimes function uses a while loop to continuously generate prime numbers. It maintains a list primes to keep track of the prime numbers generated so far. The function checks if a number num is prime by dividing it with all the prime numbers in the primes list. If num is not divisible by any prime number, it is considered a prime and yielded by the generator. The generator continues to generate the next prime number on each successive call to its **next()** method.