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).

In [1]:
```
60*60
```

Out[1]:
```
3600
```

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

In [2]:
```
seconds_per_hour = 60 * 60
```

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 86,400 seconds in a day.

To arrive at this answer using the given variables:

There are 60 minutes in an hour

There are 60 seconds in a minute

Therefore, there are 60 x 60 = 3600 seconds in an hour

To get the seconds in a day, we can multiply the seconds per hour by the number of hours in a day, which is 24

In [4]:
```
seconds_per_hour = 60 * 60
minutes_per_hour = 60
```

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

In [5]:
```
seconds_per_hour = 60 * 60
seconds_per_day = seconds_per_hour * 24
print(seconds_per_day)
```

```
86400
```

5. Divide seconds_per_day by seconds_per_hour. Use floating-point (/) division.

In [6]:
```
seconds_per_hour = 60 * 60  # 3600
seconds_per_day = seconds_per_hour * 24  # 86400
print(seconds_per_day / seconds_per_hour)
```

```
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?

No, the result of integer division (//) will be different from the floating-point division (/) even if the final .0 is ignored. This is because integer division discards any remainder and returns only the whole number portion of the result.

In [7]:
```
print(seconds_per_day // seconds_per_hour)
```

```
24
```

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

In [9]:
```python
def genPrimes():
    primes = []  # to store the prime numbers generated so far
    num = 2  # start with the first prime number

    while True:
        is_prime = True  # assume num is prime

        # check if num is divisible by any of the prime numbers generated so far
        for p in primes:
            if num % p == 0:
                is_prime = False
                break

        if is_prime:
            primes.append(num)  # add num to the list of primes
            yield num  # yield the prime number

        num += 1  # move on to the next number


# create a generator object
prime_gen = genPrimes()

# print the first 10 prime numbers
for i in range(10):
    print(next(prime_gen))
```

```
2
3
5
7
11
13
17
19
23
29
```

In [ ]: