**Name : Channabasav Angadi**

**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).**

60 \* 60

**3600**

**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**

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

**seconds\_per\_hour \* 24**

**=86400**

**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

Print(seconds\_per\_day)

=86400

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

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?**

seconds\_per\_day // seconds\_per\_hour

**24**

**Yes**

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

def genPrimes():

    primes = []

    n = 2

    last = n

    while True:

        for i in primes:

            if n % i == 0:

                n += 1

                break

        else:

            primes.append(n)

            last = n

            n += 1

            yield last

genPrimes()

**<generator object genPrimes at 0x0000025AFFEBB0B0>**

**None**

def genPrimes():

    primes = [2]

    yield primes[0]

    guess = 3

    while True:

        if all(guess%x != 0 for x in primes):

            primes.append(guess)

        if guess == primes[-1]:

            yield primes[-1]

        guess += 2

genPrimes()

**<generator object genPrimes at 0x0000025AFFEBB4A0>**