**Source code for Ques(1) to (6):**

**seconds\_per\_hour = 3600**

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

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

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

**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).**

**Ans:** 60\*60

3600

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

**Ans:** seconds\_per\_hour = 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.**

**Ans:** seconds\_per\_hour \* 24

86400

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

**Ans:** seconds\_per\_day = seconds\_per\_hour \* 24

seconds\_per\_day

86400

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

**Ans:** 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?**

**Ans:** Yes, I agree

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, ...**

**Ans: The code is here:**

def genPrimes():

primes = [] *# primes generated so far*

last = 1 *# last number tried*

while True:

last += 1

for p in primes:

if last % p == 0:

break

else:

primes.append(last)

yield last

p = genPrimes()