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).

sol. 60

ANS seconds\_per\_hour = 3600

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

seconds\_per\_day / seconds\_per\_hour

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

years\_list = [2000, 2001, 2002, 2003, 2004, 2005]

years\_list[2]

years\_list[-1]

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

seconds\_per\_hour.

ANS seconds\_per\_hour = 3600

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

seconds\_per\_day / seconds\_per\_hour

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

years\_list = [2000, 2001, 2002, 2003, 2004, 2005]

years\_list[2]

years\_list[-1]

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 There are 24 hrs in a day. So there are 2460 mins in a day. ( 1hr=60 mins) So there are 246060 seconds in a day. (1 min = 60 seconds) Therefore 86,400 seconds in a day.

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

ANS seconds\_per\_hour = 3600

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

seconds\_per\_day / seconds\_per\_hour

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

years\_list = [2000, 2001, 2002, 2003, 2004, 2005]

years\_list[2]

years\_list[-1]

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

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 ython provides two types of division – float and integer. The / operator is used for float division, while the // operator is used for integer division. For example, 10 / 3 results in 3.3333333333333335 (float division), while 10 // 3 results in 3 (integer division).

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 **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()

FINISH\*\*\*\*