1.

60\*60

3600

2.

Second\_per\_hour \*24

86400

3.

Seconds\_per\_day/second\_per\_hour

24.0

4.

Seconds\_per\_day//second\_per\_hour

24

5.

Years

6.

|  |
| --- |
| # 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 |