**Assignment 15**

**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**

**To calculate the number of seconds in an hour using Python, you can use the interactive interpreter and multiply the number of seconds in a minute (60) by the number of minutes in an hour (also 60) as follows:**

**python**

**Copy code**

**>>> 60 \* 60**

**3600**

**Therefore, there are 3600 seconds in an hour.**

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

**To assign the result from the previous task (seconds in an hour) to a variable called seconds\_per\_hour, you can simply use the assignment operator = as follows:**

**makefile**

**Copy code**

**seconds\_per\_hour = 60 \* 60**

**Now the variable seconds\_per\_hour contains the value 3600, which is the number of seconds in an hour.**

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

**To calculate the number of seconds in a day using the seconds\_per\_hour variable that we defined in the previous task, we can use the following formula:**

**makefile**

**Copy code**

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

**This is because there are 24 hours in a day. Now, substituting the value of seconds\_per\_hour that we calculated earlier, we get:**

**makefile**

**Copy code**

**seconds\_per\_day = 3600 \* 24**

**Simplifying this expression, we get:**

**makefile**

**Copy code**

**seconds\_per\_day = 86400**

**Therefore, there are 86,400 seconds in a day, assuming a standard 24-hour day.**

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

**To calculate the number of seconds in a day and save the result in a variable called seconds\_per\_day, we can use the following code:**

**makefile**

**Copy code**

**seconds\_per\_hour = 60 \* 60**

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

**Now the variable seconds\_per\_day contains the value 86400, which is the number of seconds in a day, assuming a standard 24-hour day.**

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

**To divide seconds\_per\_day by seconds\_per\_hour using floating-point division, we can use the following code:**

**makefile**

**Copy code**

**seconds\_per\_hour = 60 \* 60**

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

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

**This will output the result of the division, which is 24.0. This means that there are 24 hours in a day.**

**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?**

**To divide seconds\_per\_day by seconds\_per\_hour using integer division, we can use the double slash (//) operator as follows:**

**makefile**

**Copy code**

**seconds\_per\_hour = 60 \* 60**

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

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

**This will output the result of the integer division, which is 24. This agrees with the floating-point value from the previous question, aside from the final .0, which is because the floating-point division returns a float value while the integer division returns an integer value. So, both calculations confirm that there are 24 hours in a day.**

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

**To generate an infinite sequence of prime numbers using a generator function in Python, we can use the following code:**

**python**

**Copy code**

**def genPrimes():**

**"""**

**Generator that yields the sequence of prime numbers on successive calls to its next() method.**

**"""**

**primes = [2] # initialize the list of primes with the first prime number**

**yield 2 # yield the first prime number**

**current = 3 # initialize the current number to check for primality**

**while True: # loop forever**

**is\_prime = True # assume current is prime until proven otherwise**

**for p in primes:**

**if current % p == 0: # current is divisible by a previous prime number**

**is\_prime = False # current is not prime**

**break**

**if is\_prime: # current is prime**

**primes.append(current) # add current to the list of primes**

**yield current # yield current as the next prime number**

**current += 2 # move on to the next odd number (even numbers other than 2 are not prime)**

**This generator works by keeping a list of previously generated prime numbers and checking if the next odd number is divisible by any of them. If not, the number is added to the list of primes and yielded as the next prime number.**

**To use this generator, you can create an instance of it and call its next() method to get the next prime number in the sequence:**

**python**

**Copy code**

**primes = genPrimes()**

**print(next(primes)) # output: 2**

**print(next(primes)) # output: 3**

**print(next(primes)) # output: 5**

**print(next(primes)) # output: 7**

**print(next(primes)) # output: 11**

**# and so on...**

**Note that this generator will continue to yield prime numbers indefinitely until it is interrupted or stopped manually.**