**Python 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).**

=> Using the interactive interpreter, you can perform this calculation as follows:

seconds\_in\_minute = 60

minutes\_in\_hour = 60

seconds\_in\_hour = seconds\_in\_minute \* minutes\_in\_hour

seconds\_in\_hour

output : 3600

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

=> 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.**

=> we can multiply the number of seconds per hour by the number of hours in a day.

hours\_in\_day = 24

seconds\_per\_day = seconds\_per\_hour \* hours\_in\_day

seconds\_per\_day

output: 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 \* minutes\_in\_hour \* hours\_in\_day

seconds\_per\_day

output : 86400

**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, you can perform the following calculation:

result = seconds\_per\_day / seconds\_per\_hour

result

output: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?**

=> result\_integer\_division = seconds\_per\_day // seconds\_per\_hour

result\_integer\_division

output:24

The result of the integer division is 24. This value agrees with the floating-point division result of 24.0 from the previous question, aside from the absence of the decimal point and zero. Both calculations indicate 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, ...**

=>

def genPrimes():

primes = []

num = 2

while True:

is\_prime = True

for prime in primes:

if num % prime == 0:

is\_prime = False

break

if is\_prime:

yield num

primes.append(num)

num += 1

primes\_generator = genPrimes()

print(next(primes\_generator)) # Output: 2

print(next(primes\_generator)) # Output: 3

print(next(primes\_generator)) # Output: 5

print(next(primes\_generator)) # Output: 7

print(next(primes\_generator)) # Output: 11