**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**

In Python, you can use the interactive interpreter as a calculator to find the number of seconds in an hour by multiplying the number of seconds in a minute (60) by the number of minutes in an hour (60). Here's how you can do it:

```python

# Calculate the number of seconds in an hour

seconds\_in\_minute = 60

minutes\_in\_hour = 60

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

print(seconds\_in\_hour)

```

When you run this code, you'll get the result:

```

3600

```

So, there are 3600 seconds in an hour.

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

Sure! To assign the result (3600 seconds) from the previous task to a variable called `seconds\_per\_hour`, you can do the following:

```python

# Calculate the number of seconds in an hour

seconds\_in\_minute = 60

minutes\_in\_hour = 60

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

print(seconds\_per\_hour)

```

When you run this code, the variable `seconds\_per\_hour` will be assigned the value of `3600`, which represents 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, you can use the `seconds\_per\_hour` variable from the previous task and the number of hours in a day (24 hours). Here's how you can do it:

```python

# Calculate the number of seconds in an hour

seconds\_in\_minute = 60

minutes\_in\_hour = 60

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

# Calculate the number of seconds in a day

hours\_in\_day = 24

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

print(seconds\_per\_day)

```

When you run this code, you'll get the result:

```

86400

```

So, there are 86,400 seconds in a day.

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

Apologies for the confusion in my previous response. Let's calculate the number of seconds in a day and save the result in a variable called `seconds\_per\_day`:

```python

# Calculate the number of seconds in an hour

seconds\_in\_minute = 60

minutes\_in\_hour = 60

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

# Calculate the number of seconds in a day

hours\_in\_day = 24

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

print(seconds\_per\_day)

```

When you run this code, you'll get the correct result:

```

86400

```

So, there are 86,400 seconds in a day, and the value is stored in the variable `seconds\_per\_day`.

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

Sure, let's divide `seconds\_per\_day` by `seconds\_per\_hour` using floating-point division in Python:

```python

# Calculate the number of seconds in an hour

seconds\_in\_minute = 60

minutes\_in\_hour = 60

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

# Calculate the number of seconds in a day

hours\_in\_day = 24

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

# Divide seconds\_per\_day by seconds\_per\_hour using floating-point division

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

print(result)

```

When you run this code, you'll get the result:

```

24.0

```

So, the result of dividing `seconds\_per\_day` by `seconds\_per\_hour` is `24.0`. This indicates that there are 24 hours in a day, which is the expected outcome.

**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?**

Let's divide `seconds\_per\_day` by `seconds\_per\_hour` using integer (//) division in Python:

```python

# Calculate the number of seconds in an hour

seconds\_in\_minute = 60

minutes\_in\_hour = 60

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

# Calculate the number of seconds in a day

hours\_in\_day = 24

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

# Divide seconds\_per\_day by seconds\_per\_hour using integer (//) division

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

print(result\_integer\_division)

```

When you run this code, you'll get the result:

```

24

```

The result of the integer (//) division is `24`, which is the same as the floating-point division result, aside from the final `.0`. The floating-point division produces a result with a decimal point (24.0), but both results represent the number of hours in a day. The decimal point in the floating-point result doesn't affect the agreement with the integer result, as the actual value is still the same.

**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 create a generator that generates prime numbers on successive calls to its `next()` method, you can implement the `genPrimes` generator function in Python. Here's a possible implementation using a simple approach:

```python

def is\_prime(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

def genPrimes():

num = 2

while True:

if is\_prime(num):

yield num

num += 1

# Create the generator

prime\_generator = genPrimes()

# Generate and print the first few prime numbers

for \_ in range(10):

print(next(prime\_generator))

```

In this code, the `is\_prime` function checks whether a given number is prime. The `genPrimes` generator function initializes `num` to 2 and then keeps incrementing `num` while checking each number if it's prime using the `is\_prime` function. When a prime number is found, it is yielded using the `yield` keyword. The generator will continue generating prime numbers indefinitely when calling `next()` on it.

When you run this code, you'll get the first ten prime numbers:

```

2

3

5

7

11

13

17

19

23

29

```

The generator will keep producing prime numbers as you call `next()` on it.