Name :- Deepawali . B. Mhaisagar
Assignment no 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).

ANS

To calculate the number of seconds in an hour, you can multiply the number of seconds in a minute (60) by the number of minutes in an hour (60). Using the interactive interpreter as a calculator, you can perform this calculation directly:

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.

ANS

To assign the result of the calculation, which is the number of seconds in an hour (3600), to a variable called `seconds_per_hour`, you can do the following in Python:

seconds_per_hour = 60 * 60

After executing this code, the value 3600 will be assigned to the variable `seconds_per_hour`. You can then use this variable throughout your program to refer to the number of seconds in an hour.

3. What is a box tuple, and how does it work?

ANS

To calculate the number of seconds in a day, you can utilize the variables `seconds_per_hour` (which is 3600) and the number of minutes in an hour (60). Since there are 24 hours in a day, you can multiply the number of seconds per hour by the number of hours in a day.

Considering 60 minutes per hour and 24 hours per day, you can calculate the number of seconds in a day as follows:

seconds_per_hour = 60 * 60

minutes_per_hour = 60

hours_per_day = 24

seconds_per_day = seconds_per_hour * minutes_per_hour * hours_per_day

After executing this code, the variable `seconds_per_day` will contain the number of seconds in a day, which is 86,400.

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

ANS

To calculate the number of seconds in a day and save the result in a variable called `seconds_per_day`, you can use the previously defined variables `seconds_per_hour`, `minutes_per_hour`, and `hours_per_day`. Here's the code:

seconds_per_hour = 60 * 60

minutes_per_hour = 60

hours_per_day = 24

seconds_per_day = seconds_per_hour * minutes_per_hour * hours_per_day

After executing this code, the variable `seconds_per_day` will store the value 86,400, which represents the number of seconds in a day. You can then use the `seconds_per_day` variable in your program to refer to this value as needed.

5. What method would you call to get Image object for a 100×100 image, excluding the lower-left quarter of it?

ANS

To divide the `seconds_per_day` variable by the `seconds_per_hour` variable using floating-point division (/), you can perform the calculation as follows:

```
seconds_per_day = 86400

seconds_per_hour = 3600



result = seconds_per_day / seconds_per_hour
```

After executing this code, the `result` variable will contain the result of the division, which is 24.0. The use of floating-point division allows for the result to be a decimal value representing the number of hours in a day.

6. After making changes to an Image object, how could you save it as an image file?

ANS

To divide the `seconds_per_day` variable by the `seconds_per_hour` variable using integer division (//), you can perform the calculation as follows:

```
seconds_per_day = 86400

seconds_per_hour = 3600



result = seconds_per_day // seconds_per_hour
```

After executing this code, the `result` variable will contain the result of the integer division, which is 24. Integer division discards the decimal part of the division result, resulting in a whole number.

In this case, the integer division result of 24 matches the floating-point division result of 24.0 from the previous question, aside from the presence of the decimal point. Both calculations represent the whole number of hours in a day, but with different data types (integer vs. floating-point).

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

Certainly! Here's an implementation of the `genPrimes` generator that returns the sequence of prime numbers on successive calls to its `next()` method:

```
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:
            primes.append(num)
            yield num

        num += 1
```

In this implementation, the `genPrimes` generator initializes an empty list called `primes` and starts with the number 2. It enters an infinite loop, continually generating and yielding prime numbers.

For each number `num`, it checks if it is divisible by any previously found prime numbers in the `primes` list. If it is not divisible by any of them, it is considered a prime number and added to the `primes` list. Then, the `num` is yielded as the next prime number in the sequence.

To use the `genPrimes` generator, you can create an instance and call its `next()` method in a loop to obtain successive prime numbers:

```
primes_generator = genPrimes()



# Get the first 10 prime numbers

for _ in range(10):

    prime = next(primes_generator)

    print(prime)
```

In the example above, the loop runs 10 times, and in each iteration, it calls `next(primes_generator)` to obtain the next prime number from the generator and prints it.