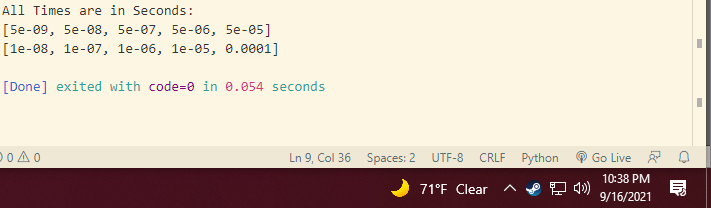
**Question 1:**



Code:

numsToSort = [100, 1000, 10000, 100000, 1000000]

aSpeed, bSpeed = 20 \* 10\*\*9, 10 \* 10\*\*9

aTime, bTime = [], []

for num in numsToSort:

  aTime.append(num / aSpeed)

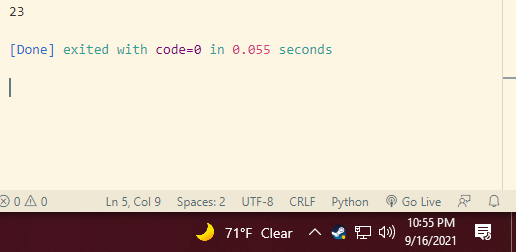
  bTime.append(num / bSpeed)

print('All Times are in Seconds: ')

print(aTime)

print(bTime)

**Question 2:**



Code:

import math

n = 2 *#math.log2(1) == 0, so we start at 2*

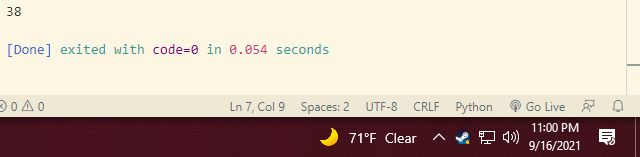
while (5 \* math.pow(n, 2)) <= (25 \* n \* math.log2(n)):

  n += 1

print(n)

The answer is: for 2 <= n < 23, insertion sort will beat merge sort.

**Question 3:**



Code:

import math

n = 2 *#math.log2(1) == 0, so we start at 2*

insertion = **lambda** n: 7 \* n\*\*2

merge = **lambda** n: 50 \* n \* math.log2(n)

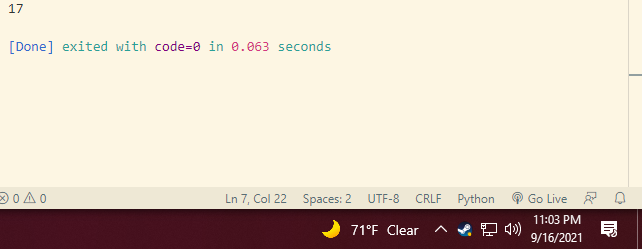
while insertion(n) <= merge(n):

  n += 1

print(n)

The answer is: for 2 <= n < 38, insertion sort will be faster.

**Question 4:**



Code:

import math

n = 0

insertion = **lambda** n: 50 \* n\*\*2

merge = **lambda** n: 3 \* n\*\*3

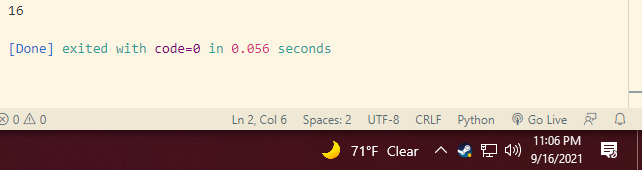
while insertion(n) >= merge(n):

  n += 1

print(n)

For 0 <= n < 17, 50n2 will run slower than 3n.

**Question 5:**



Code:

n = 1

insertion = **lambda** n: 10 \* n\*\*3

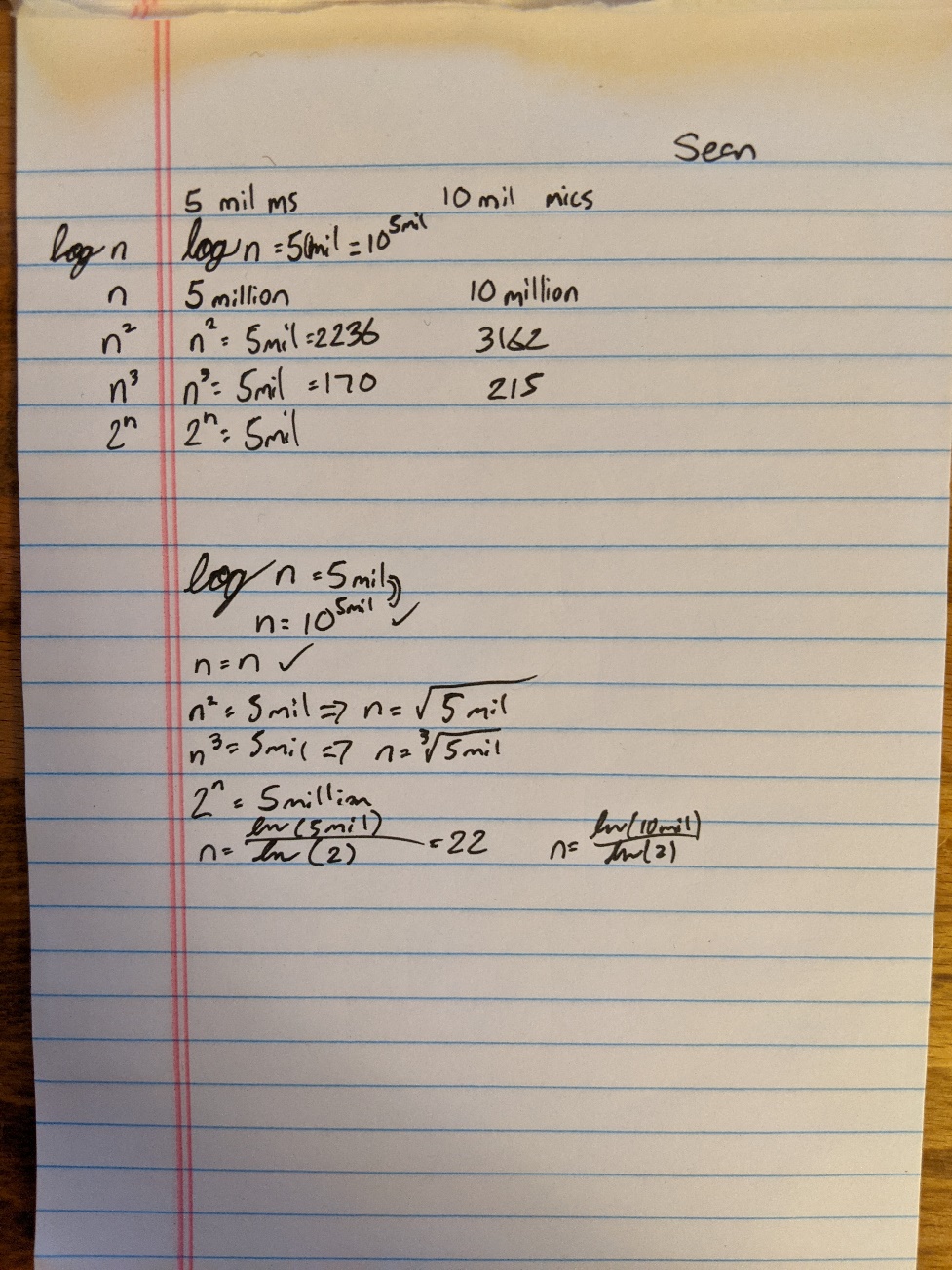
merge = **lambda** n: 2\*\*n

while insertion(n) >= merge(n):

  n += 1

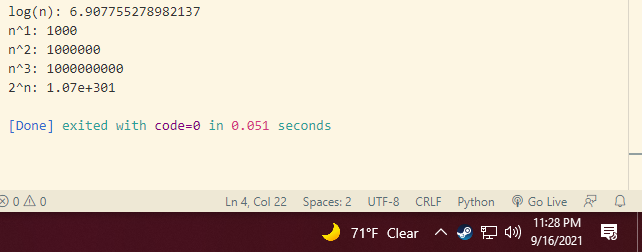
print(n)

Technically, 0 is the smallest value that will cause 10n3 to run faster, but after that, it’s 16.

**Question 6:**

|  |  |  |
| --- | --- | --- |
|  | 5 seconds | 10 seconds |
| log n | 105,000,000 | 1010,000,000 |
| n | 5,000,000 | 10,000,000 |
| n2 | 2236 | 3162 |
| n3 | 170 | 215 |
| 2n | 22 | 23 |

**Question 7:**

Code:

import math

n = 1000

print(**f**'log(n): {math.log(n)}')

for i in range(1, 4):

  print(**f**'n^{i}: {n\*\*i}')

print(**f**'2^n: {"{**:.2e**}".format(2\*\*n)}')