Question 1:

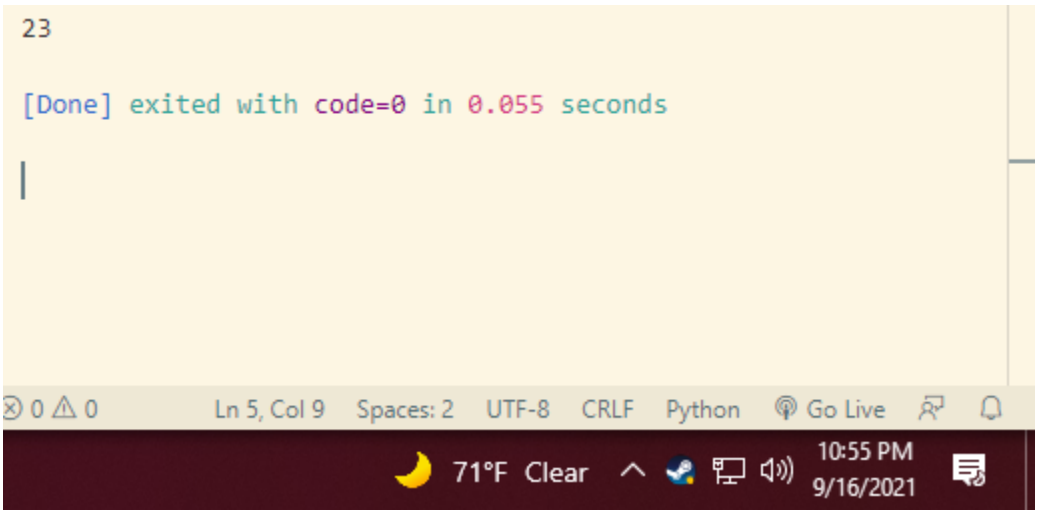
Code:

```
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:

```
38

[Done] exited with code=0 in 0.054 seconds
```

0 0 Ln 7, Col 9 Spaces: 2 UTF-8 CRLF Python Go Live

71°F Clear 11:00 PM 9/16/2021

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:

```
17

[Done] exited with code=0 in 0.063 seconds
```

0 0 Ln 7, Col 22 Spaces: 2 UTF-8 CRLF Python Go Live

71°F Clear 11:03 PM 9/16/2021

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, $50n^2$ will run slower than 3^n.

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 $10n^3$ to run faster, but after that, it's 16.

Question 6:

	5 seconds	10 seconds
log n	$10^{5,000,000}$	$10^{10,000,000}$
n	5,000,000	10,000,000
n^2	2236	3162
n^3	170	215
2^n	22	23

Sean

	5 mil ms	10 mil mics
log n	$\log n = 5mil = 10^{5mil}$	
n	5 million	10 million
n^2	$n^2 = 5mil = 2236$	3162
n^3	$n^3 = 5mil = 170$	215
2^n	$2^n = 5mil$	

$\log n = 5mil$
$n = 10^{5mil}$ ✓

$n = n$ ✓

$n^2 = 5mil \Rightarrow n = \sqrt{5mil}$

$n^3 = 5mil \Rightarrow n = \sqrt[3]{5mil}$

$2^n = 5million$

$n = \frac{\ln(5mil)}{\ln(2)} = 22$ $n = \frac{\ln(10mil)}{\ln(2)}$

Question 7:

```
log(n): 6.907755278982137
n^1: 1000
n^2: 1000000
n^3: 1000000000
2^n: 1.07e+301

[Done] exited with code=0 in 0.051 seconds
```

0 0 Ln 4, Col 22 Spaces: 2 UTF-8 CRLF Python Go Live

71°F Clear 11:28 PM 9/16/2021

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)}')
```