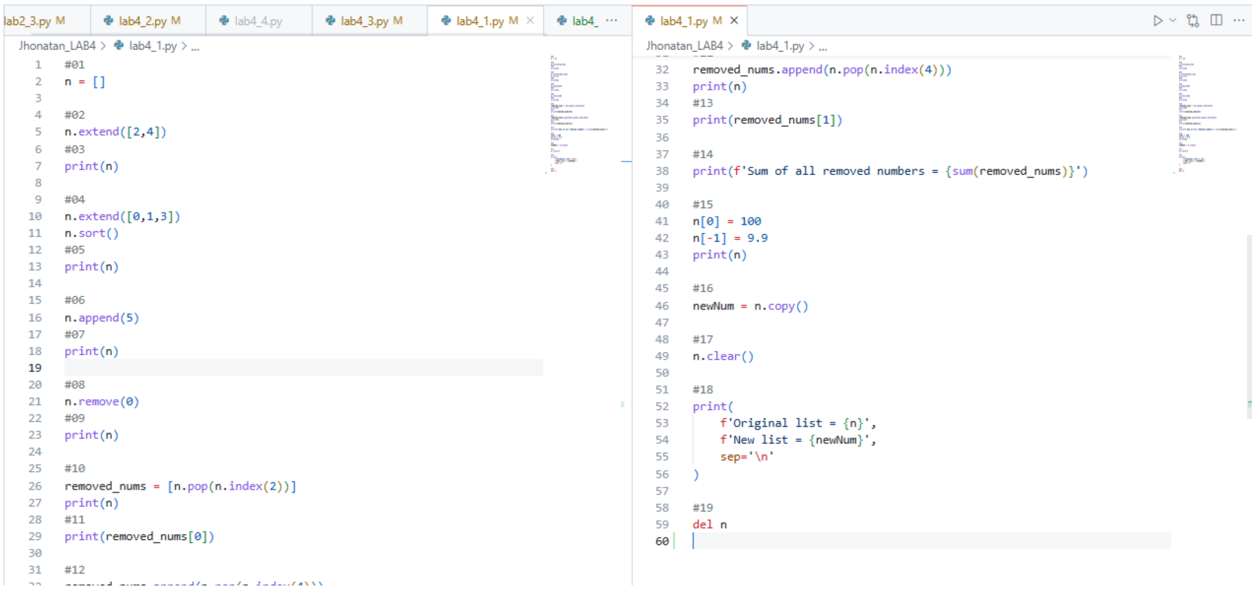
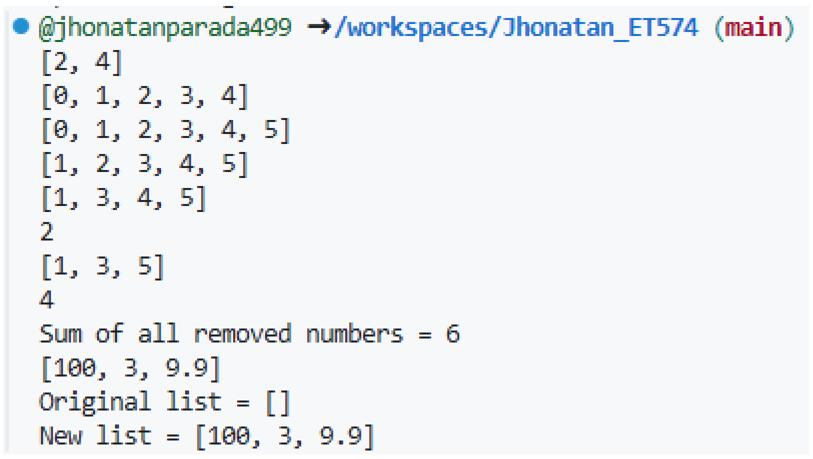
Lab 4 Summary

Jhonatan Parada Torres

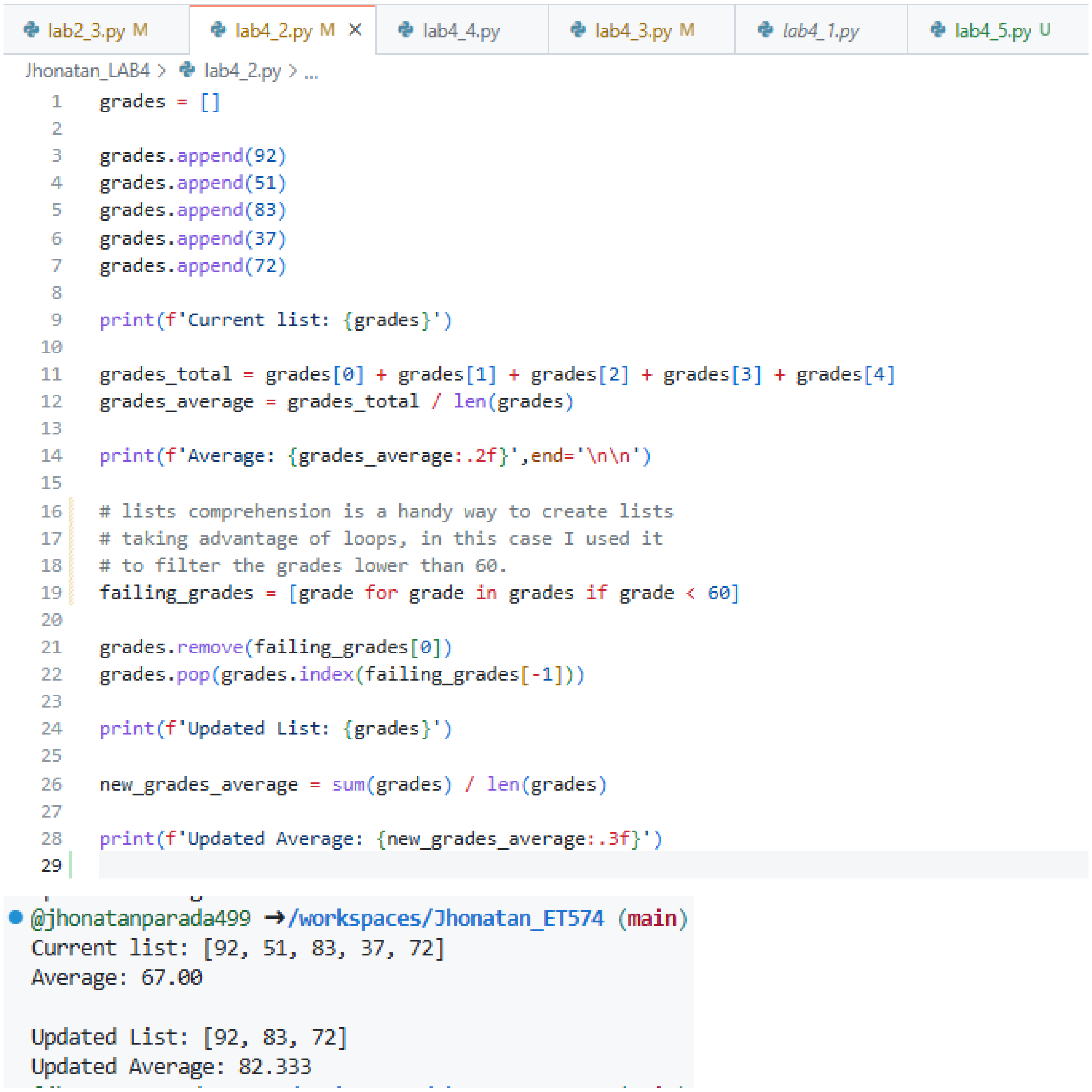
1.



```python
#01
n = []

#02
n.extend([2,4])
#03
print(n)

#04
n.extend([0,1,3])
n.sort()
#05
print(n)

#06
n.append(5)
#07
print(n)

#08
n.remove(0)
#09
print(n)

#10
removed_nums = [n.pop(n.index(2))]
print(n)
#11
print(removed_nums[0])

#12
removed_nums.append(n.pop(n.index(4)))
print(n)
#13
print(removed_nums[1])

#14
print(f'Sum of all removed numbers = {sum(removed_nums)}')

#15
n[0] = 100
n[-1] = 9.9
print(n)

#16
newNum = n.copy()

#17
n.clear()

#18
print(
    f'Original list = {n}',
    f'New list = {newNum}',
    sep='\n'
)

#19
del n
```



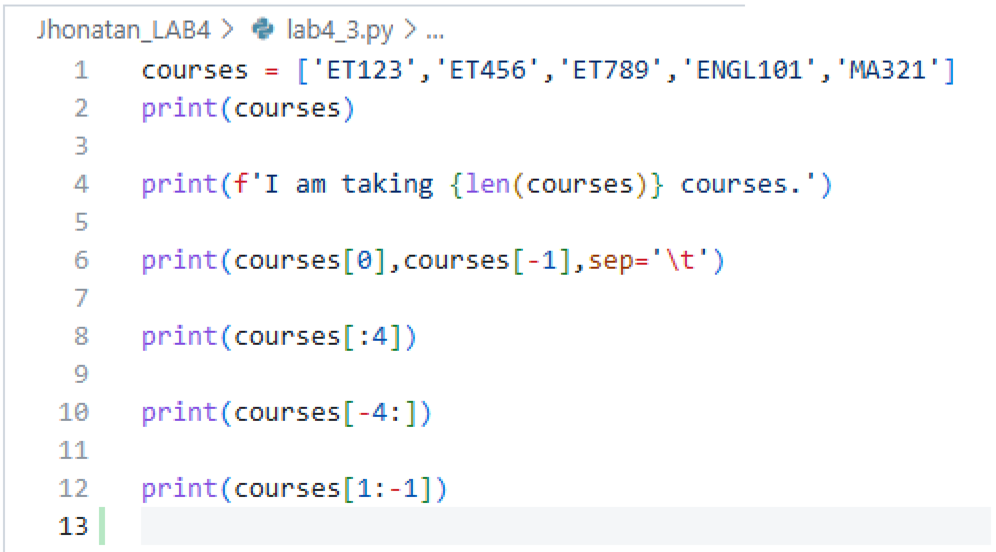
```
@jhonatanparada499 ➜/workspaces/Jhonatan_ET574 (main)
[2, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 3, 4, 5]
2
[1, 3, 5]
4
Sum of all removed numbers = 6
[100, 3, 9.9]
Original list = []
New list = [100, 3, 9.9]
```


lab2_3.py M | lab4_2.py M × | lab4_4.py | lab4_3.py M | lab4_1.py | lab4_5.py U

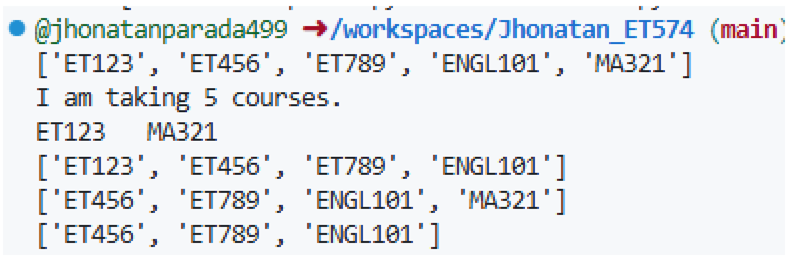
Jhonatan_LAB4 > lab4_2.py > ...

```python
grades = []

grades.append(92)
grades.append(51)
grades.append(83)
grades.append(37)
grades.append(72)

print(f'Current list: {grades}')

grades_total = grades[0] + grades[1] + grades[2] + grades[3] + grades[4]
grades_average = grades_total / len(grades)

print(f'Average: {grades_average:.2f}',end='\n\n')

# lists comprehension is a handy way to create lists
# taking advantage of loops, in this case I used it
# to filter the grades lower than 60.
failing_grades = [grade for grade in grades if grade < 60]

grades.remove(failing_grades[0])
grades.pop(grades.index(failing_grades[-1]))

print(f'Updated List: {grades}')

new_grades_average = sum(grades) / len(grades)

print(f'Updated Average: {new_grades_average:.3f}')

```

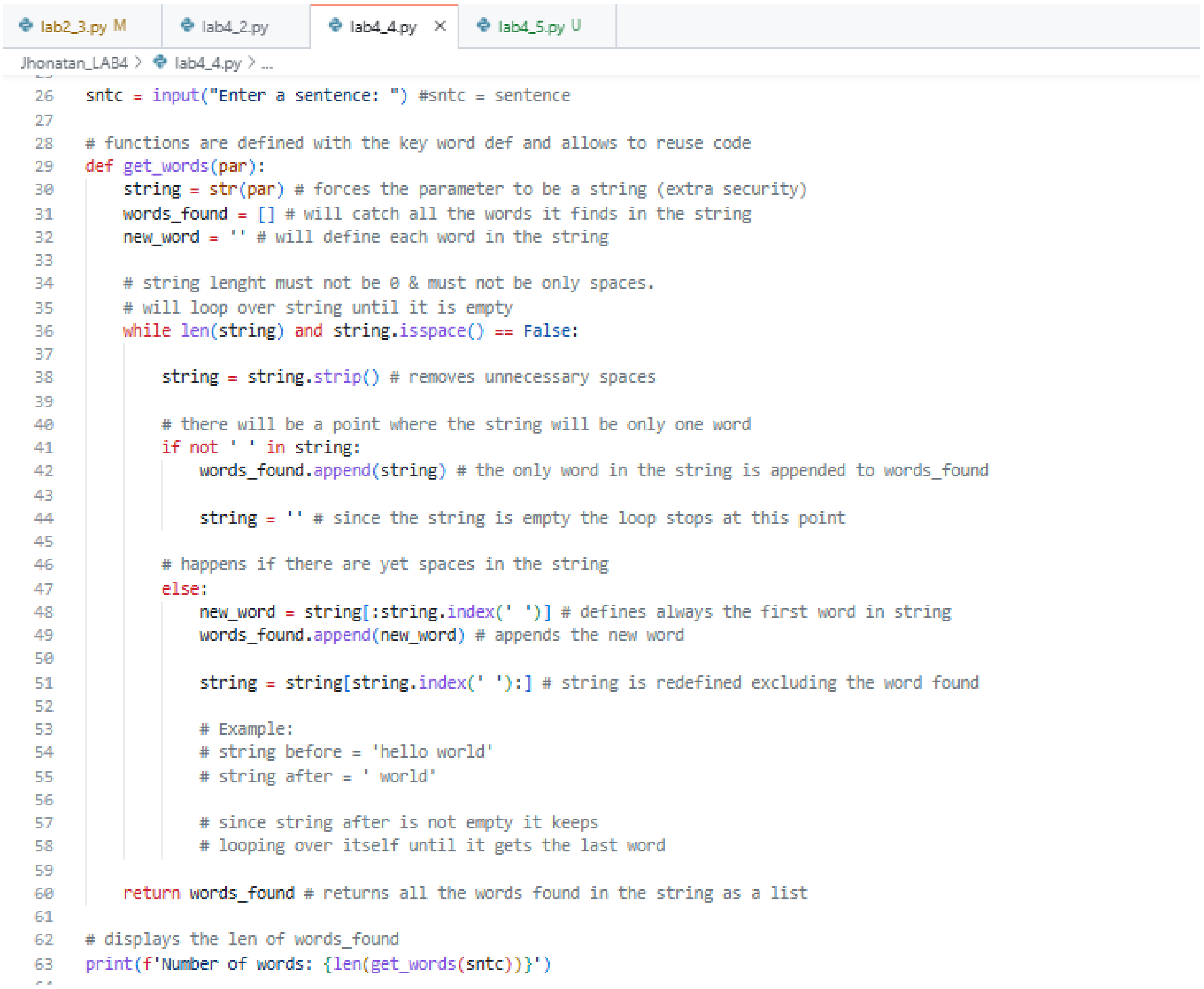
```
● @jhonatanparada499 ➜ /workspaces/Jhonatan_ET574 (main)
Current list: [92, 51, 83, 37, 72]
Average: 67.00

Updated List: [92, 83, 72]
Updated Average: 82.333
```


Jhonatan_LAB4 > lab4_3.py > ...

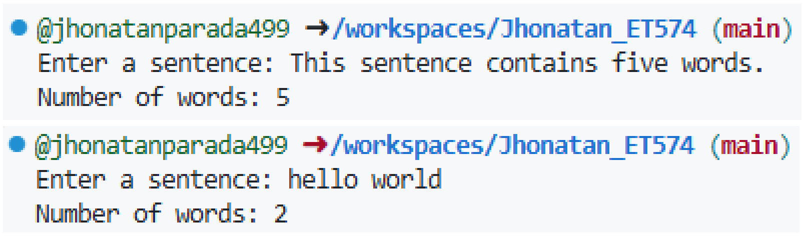
```python
courses = ['ET123','ET456','ET789','ENGL101','MA321']
print(courses)

print(f'I am taking {len(courses)} courses.')

print(courses[0],courses[-1],sep='\t')

print(courses[:4])

print(courses[-4:])

print(courses[1:-1])

```



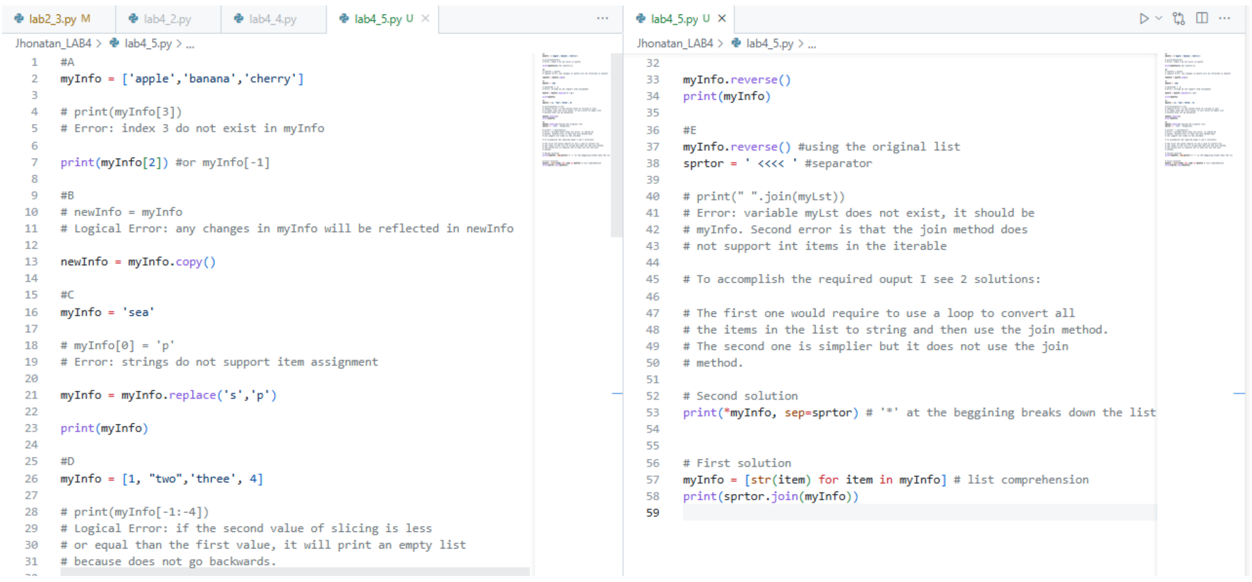
```
● @jhonatanparada499 ➜ /workspaces/Jhonatan_ET574 (main)
  ['ET123', 'ET456', 'ET789', 'ENGL101', 'MA321']
  I am taking 5 courses.
  ET123   MA321
  ['ET123', 'ET456', 'ET789', 'ENGL101']
  ['ET456', 'ET789', 'ENGL101', 'MA321']
  ['ET456', 'ET789', 'ENGL101']
```



```python
sntc = input("Enter a sentence: ") #sntc = sentence

# functions are defined with the key word def and allows to reuse code
def get_words(par):
    string = str(par) # forces the parameter to be a string (extra security)
    words_found = [] # will catch all the words it finds in the string
    new_word = '' # will define each word in the string

    # string lenght must not be 0 & must not be only spaces.
    # will loop over string until it is empty
    while len(string) and string.isspace() == False:

        string = string.strip() # removes unnecessary spaces

        # there will be a point where the string will be only one word
        if not ' ' in string:
            words_found.append(string) # the only word in the string is appended to words_found

            string = '' # since the string is empty the loop stops at this point

        # happens if there are yet spaces in the string
        else:
            new_word = string[:string.index(' ')] # defines always the first word in string
            words_found.append(new_word) # appends the new word

            string = string[string.index(' '):] # string is redefined excluding the word found

            # Example:
            # string before = 'hello world'
            # string after = ' world'

            # since string after is not empty it keeps
            # looping over itself until it gets the last word

    return words_found # returns all the words found in the string as a list

# displays the len of words_found
print(f'Number of words: {len(get_words(sntc))}')
```

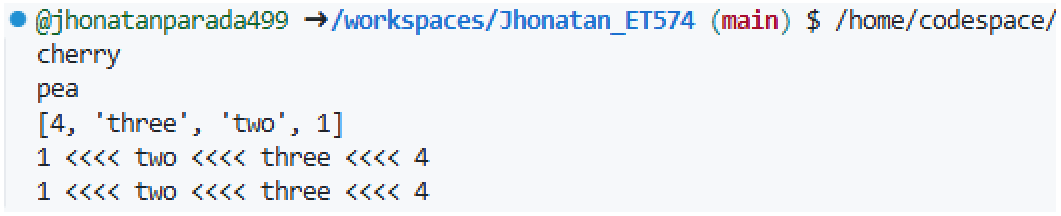


```
● @jhonatanparada499 ➜ /workspaces/Jhonatan_ET574 (main)
  Enter a sentence: This sentence contains five words.
  Number of words: 5
● @jhonatanparada499 ➜ /workspaces/Jhonatan_ET574 (main)
  Enter a sentence: hello world
  Number of words: 2
```



```python
#A
myInfo = ['apple','banana','cherry']

# print(myInfo[3])
# Error: index 3 do not exist in myInfo

print(myInfo[2]) #or myInfo[-1]

#B
# newInfo = myInfo
# Logical Error: any changes in myInfo will be reflected in newInfo

newInfo = myInfo.copy()

#C
myInfo = 'sea'

# myInfo[0] = 'p'
# Error: strings do not support item assignment

myInfo = myInfo.replace('s','p')

print(myInfo)

#D
myInfo = [1, "two",'three', 4]

# print(myInfo[-1:-4])
# Logical Error: if the second value of slicing is less
# or equal than the first value, it will print an empty list
# because does not go backwards.
```

```python

myInfo.reverse()
print(myInfo)

#E
myInfo.reverse() #using the original list
sprtor = ' <<<< ' #separator

# print(" ".join(myLst))
# Error: variable myLst does not exist, it should be
# myInfo. Second error is that the join method does
# not support int items in the iterable

# To accomplish the required ouput I see 2 solutions:

# The first one would require to use a loop to convert all
# the items in the list to string and then use the join method.
# The second one is simplier but it does not use the join
# method.

# Second solution
print(*myInfo, sep=sprtor) # '*' at the beggining breaks down the list


# First solution
myInfo = [str(item) for item in myInfo] # list comprehension
print(sprtor.join(myInfo))

```



```
@jhonatanparada499 ➜/workspaces/Jhonatan_ET574 (main) $ /home/codespace/
cherry
pea
[4, 'three', 'two', 1]
1 <<<< two <<<< three <<<< 4
1 <<<< two <<<< three <<<< 4
```

2. Lab4_4.py was the hardest question this time in terms of logic. I could have been able to write one line of code to accomplish the function of word counting only if the input sentences had one space character between words. For example:

‘hello world’ or ‘this is a sentence.’

In this case the number of words would be the number of white spaces + 1 (that’ s assuming that there is at least one character in the sentence, if the sentence had no characters, according to that formula the result would be one word, making no sense.)

But it all can go down by an extra white space such as in:

‘hello world’ or ‘this   is   a sentence’

Now we cannot say that the number of words is the number of white spaces + 1, because there is no correlation between the characters and the white spaces. In the source code I explain further the logic I used.

I also went through a hard time when I was using the while loop, working with it is not a game as I recall from previous experiences, one mistake and the program crashes. So, there was this time during the testing of lab4_4.py where the program crashed every time I ran it. In short, while I was troubleshooting it, I noticed that the logical error was based on a particular principle of built-in methods in python: Some of them return a new value and others return none but change the object on which the method is applied.

The line of code that was causing the logical error was in line 38:

```
string.strip()
```

The fix:

```
string = string.strip()
```