Instructions:

Practice w/ list comprehensions and dicts.

Suggestions:

1) Here are some functions you might find useful: any, map, sum, zip
2) You are welcome to write your own auxiliary functions too.

Write each of these functions without any for-loops or while-loops.

1.) Write a function alnum_fn(x) that takes a string and returns True if the string is alphanumeric, otherwise False.

```
print ("1. ", "is '3' alphanumeric?  ", alnum_fn('3'))

print ("   ", "is 'x$z' alphanumeric?", alnum_fn('x$z'))

print()
```

2.) Write a function is_noun(x) that takes a part of speech such as 'NNP' and returns True if the POS represents a noun phrase, otherwise False. A POS represents a noun if its first two letters are 'NN'.

```
print ("2. ", "is NNP a noun?", is_noun('NNP'))

print ("   ", "is VP a noun? ", is_noun('VP'))

print ("   ", "is N a noun?  ", is_noun('N'))

print()
```

3.) Write a function is_even(x) that returns True if x is an even number, otherwise False. You can assume that the function will only be called with integer arguments.

```
print ("3. ", "is 3 even?  ", is_even(3))

print ("   ", "is 0 even?  ", is_even(0))

print ("   ", "is -2 even? ", is_even(-2))
```

print()

4.) Given a list such as v1 = [1, 3, 5, 7, 9], write a function add_one(x) that returns a list where each element is one greater than the original list, e.g., [2, 4, 6, 8, 10] for this example. You can assume that all of the elements in the original list are numeric.

v1 = [2, 0, -2, 4, 6]

v2 = add_one(v1)

print("4. ", "if the input is ", v1)

print("   ", "the output is   ", v2)

print()

5.) Given a list such as v2 = ['N', 'V', "JJ", "NS", "N$"], write a function drop_bad(x) that returns a list that contains only the alphabetic elements in x. In this example the result would be ['N', 'V', "JJ", "NS"].

v1 = ['abc', 'x$z', '3']

v2 = drop_bad(v1)

print("5. ", "if the input is ", v1)

print("   ", "the output is   ", v2)

print()

6.) Given a list such as

v2 = [['boy', 'N'], ['is', 'V'], ['of', 'PP'], ['xyz', 'NS'], ['abc', 'N$']]

write a function show_nouns(x) that returns a list of the same format that contains only the nouns in the input list.

v1 = [['man', 'NN'], ['man', 'VB'], ['flowers', 'NNS'], ['flowers', 'VBZ']]

```
v2 = show_nouns(v1)

print("6. ", "if the input is ", v1)
print("    ", "the output is    ", v2)
print()
```

7.) Given a list of words and parts of speech such as

v2 = [['book', ['NN', 'VB']], ['is', ['VBZ']], ['of', ['PP']]]

write a function show_nouns2(x) that returns a list of the same format containing only the nouns in the original list.

```
v1 = [['man', ['NN', 'VB']], ['flour', ['NN', 'VB']], ['the', ['DT']]]
v2 = show_nouns2(v1)

print("7. ", "if the input is ", v1)
print("    ", "the output is    ", v2)
print()
```

8.) Given a list of the format above, write a function show_nouns3(x) that returns a simple list containing only the nouns, e.g. ['book', 'x'].

```
v1 = [['man', ['NN', 'VB']], ['flour', ['NN', 'VB']], ['the', ['DT']]]
v2 = show_nouns3(v1)

print("8. ", "if the input is ", v1)
print("    ", "the output is    ", v2)
print()
```

9.) Given a list of the format

a_list = [3, 3, 4, 4, 5, 5, 6, 6, 9, 9]

write a function select_numbers(x) that returns a list containing one greater than than each even number in the list, e.g., [5, 5, 7, 7].

```
v1 = [2, 0, -1, 3, 6]

v2 = select_numbers(v1)


print("9. ", "if the input is ", v1)

print("   ", "the output is   ", v2)

print()
```

10.) Given a list of lists of numbers where the sublists do not necessarily have the same lengths

e.g., v1 = [[1, 2, 3], [5, 4]],

write a function show_count(x) that produces the total number of entries in the sublists, e.g., 5 for the list above.

```
v1 = [[1, 2, 3], [5, 4]]

v2 = show_count(v1)


print("10. ", "if the input is ", v1)

print("    ", "the output is   ", v2)

print()
```

11.) Write a function show_totals(x) that produces a list where each sublist has been summed, e.g., [6, 9] for the list above.

```
v1 = [[1, 2, 3], [5, 4]]

v2 = show_totals(v1)
```

```
print("11. ", "if the input is ", v1)
print("    ", "the output is   ", v2)
print()
```

12.) Write a function show_total(x) that produces the sum of the values in the sublists e.g. 15 for the list above.

```
v1 = [[1, 2, 3], [5, 4]]
v2 = show_total(v1)


print("12. ", "if the input is ", v1)
print("    ", "the output is   ", v2)
print()
```

13.) Given a list containing two sublists of the same length e.g., v2 = [[1, 2, 3], [5, 4, 7]],

write code to produce the dot product of the sublists, e.g. 34 for the list above (= 1*5 + 2*4 + 3*7)

```
v1 = [[1, 2, 3], [5, 4, 7]]
v2 = dot_product(v1)


print("13. ", "if the input is ", v1)
print("    ", "the output is   ", v2)
print()
```

14.) Given a sentence represented as a list of words where the last word represents the final punctuation mark, e.g., v1 = ['The', 'big', 'black', 'horse', 'is', 'black', '.'], write a function remove_dot(x) that produces a list of words without the final punctuation mark.

```
v1 = ['The', 'big', 'black', 'horse', 'is', 'black', '.']
v2 = remove_dot(v1)
```

```
print("14. ", "if the input is ", v1)
print("    ", "the output is    ", v2)
print()
```

15.) Given a sentence of the form above, write a function produce_lower(x) that contains the same sentence in lower case. Hint: Use the 'lower' function of the string class.

```
v1 = ['The', 'big', 'black', 'horse', 'is', 'black', '.']
v2 = produce_lower(v1)

print("15. ", "if the input is ", v1)
print("    ", "the output is    ", v2)
print()
```

16.) Given a sentence of the form above, write a function count_words(x) write that returns the number of distinct words in the sentence, e.g., 5 for the sentence above.

```
v1 = ['The', 'big', 'black', 'horse', 'is', 'black', '.']
v2 = count_words(v1)

print("16. ", "if the input is ", v1)
print("    ", "the output is    ", v2)
print()
```

17.) Prof. Untel (that is French for "So-and-so") represents student grades in a list, as follows: [['Aaa', 'g', [2, 4, 5]],['Bbb', 'u', [4, 5, 6]], etc.] means that student Aaa got 2 on HW1 (not HW0!), 4 on HW2, etc. The second argument represents grad/undergrad. Write a function avg_grade(x) giving the average grade for HW1.

```
v1 = [['Aaa', 'g', [2, 4, 5]],
      ['Bbb', 'u', [4, 5, 6]],
      ['Ccc', 'g', [7, 8, 9]],
      ['Ddd', 'u', [1, 2, 3]],
      ['Eee', 'u', [4, 5, 7]]]

v2 = avg_grade(v1)

print("17. ", "if the input is ", v1)
print("    ", "the output is   ", v2)
print()
```

18.) Write a function ugrad_points(x)giving the total number of points earned on HW2 by undergraduates.

```
v2 = ugrad_points(v1)

print("18. ", "if the input is ", v1)
print("    ", "the output is   ", v2)
print()
```

Test the answers:

This is how I tested my answer:

1.) make a temp file containing your answers followed by my driver

```
cat hw02.py driver3.py > temp.py
```

2.) run the temp file

```
python3 temp.py > out.txt
```

3.) compare against my output

```
diff output3.py out.txt
```

4.) Note that your answers (in a file containing ONLY your answers, NOT my driver) should precede my driver, so that the functions will be defined before they are called.

5.) I encourage you to write a script for this so you don't have to key in all of the commands every time.