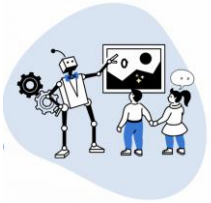


List Comprehensions - Assignment

List Comprehensions are a very powerful tool that allow us to **create a list** with one **single line of code.** **Please see the videos on www.walzl1.com for a full details.**

Examples:

```
squares = [x**2 for x in range(10)]
print (squares)

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Every time you have to use a **`for`** loop to create a list you should think:

"can I do this with a list comprehension?"

List Comprehensions can also utilize conditional statements (like an **`if`** statement to modify existing lists):

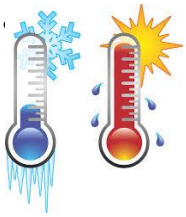
```
old_list = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
number_list = [ x for x in old_list if x % 2 == 0]
print(number_list)

[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Exercises

For the following exercises use the following list as an **input list** if one is not mentioned or given.

```
List1=[1,2,3,4,5,6,7,8,9]
```

1. Use a list comprehension to double each value in the example list shown
2. Use a list comprehension to give the Celsius value for a list of Fahrenheit temperatures. (look up the conversion online).



3. Let's say I give you the following list.
   `a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100].`
   Write one line of Python that takes this list and makes a new list that has only the odd elements of the original list in it.

4. See if you can **predict** what the following list comprehension will produce…. *then* run the code.

```
New_list= [i**2 if i%2==0 else i**3 for i in [1,2,3,4,5]]
```

5. See if you can **predict** what the following list comprehension will produce…. *then* run the code.

```
words = ['hello', 'and', 'goodbye']
new_words = [ s.upper() + '!!!' for s in words ]
```

6. For a given list make all elements which are between 3 and 8 negative values using a list comprehension.

Example:

```
mylist = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Output = [1, 2, -3, -4, -5, -6, -7, -8, 9, 10]
```

7. Given a list of numbers, return the list with all even numbers doubled, and all odd numbers turned negative.

```
 input list=[72, 26, 79, 70, 20, 68, 43, -71, 71, -2]
output_list=[144, 52, -79, 140, 40, 136, -43, 71, -71, -4]
```

8. See if you can **predict** what the following list comprehension will produce…. *then* run the code.

```
[x for x in 'MATHEMATICS' if x in ['A','E','I','O','U']]
```

9. Use a list comprehension to create a list of all the W's and T's in a word

10. Use a list comprehension to create a list of all the periods in a sentence and then uses `len()` to count them.

11. Use a list comprehension to count the number of spaces in a string.

12. Given a list of numbers, write a list comprehension that produces a list of only the positive numbers in that list.

13. Given a list of numbers, write a list comprehension that produces a list that contains the word "Even" or "Odd" in place of each number (depending on weather or not the number is even or odd.

```
Output=['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd']
```

14. Given a sentence, produce a list of the lengths of each word in the sentence, but only if the word is not 'the'.

```
Sentence = 'the quick brown fox jumps over the lazy dog'
Output = [5, 5, 3, 5, 4, 4, 3]
```

15. Let's say you want to create a program that will sent information in a coded form that others can't read.

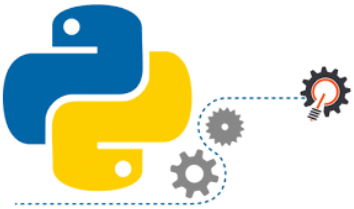
16. Given a sentence, return the sentence will all its letter transposed by 1 in the alphabet, but only if the letter is a-y.

```
inputList  = ('the quick brown fox jumps over the lazy dog')
outputList = 'uif rvjdl cspxo gpy kvnqt pwfs uif mbzy eph'
```

Challenging Exercises: (work with a friend if you want)

17. See if you can use a single list comprehension to generate the following `Output_list` form the `Input_list` shown:

```
Input_list=[1,2,3]
Output_list = [1,2,3,1,4,9,1,8,27,1,16,81,1,32,243]
```



18. Write a list comprehension that uses **nested for statements** to create a single list with all 36 different dice combinations from (1,1) to (6,6)….whaaaaat?...look it up!

19. Let's assume that you are responsible for analyzing the outcome of a referendum organized to decide whether or not BC should secede from the rest of Canada. You are being given the data in the form of the string below, which is a set of "yes/no" votes, where **"y" and "Y" both mean "yes"** and **"n" and "N" both mean "no"**.

```
Input_votes = "yynNYYnnNyYnY"
```

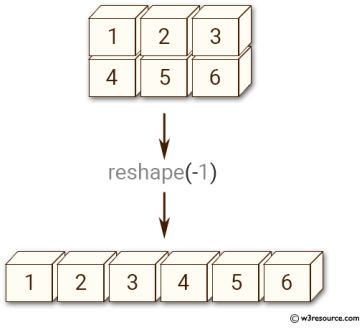
Use a list comprehension to output a list of the number of yes votes and number of no votes.

```
Output_list=[7,5]
```

Now see if you can make your program work while taking input from list (string) that is from an external file!...you'll be a pro if you can do this one!

Super Pro Exercise:

20. A common procedure in programming is call "flattening a matrix" this is taking a two dimensional list (or array) and putting it into a single (linear) list.



Example:

```
Matrix=[[1,2,3],[2,4,6],[7,8,9]]

Ouput_list=[1,2,3,2,4,6,7,8,9]
```

Use a single list comprehension to turn a 2-D list into the `Output_list` shown above. You might have to look it up...but that's OK!