1 Easy Exercises

1.1 Question 1

Marco has a home library and he has many books. He wants to know how many different authors of books he has. He write down the name of all authors book by book. Given a list of authors, make a function that returns the number of unique authors.

```
def unique_authors(author_list):
    ...
# Example:
author_list = ['Hugo', 'Nietzsche', 'shakespeare', 'Walker', 'Camus', 'Moliere', 'Hugo',
    'Nietzsche', 'Camus', 'Moliere', 'Walker', 'Nietzsche']

# Solution:
unique_authors(author_list) # 6
```

1.2 Question 2

A musician has several musical instruments. He is also a passionate of consonants and vowels. He would like to know which of his musical instruments contain at least three vowels. Given a list of musical instruments, make a function that returns the number of instruments containing at least three vowels.

```
def at_least_three_vowels(instruments_list):
    ...
# Example:
instruments_list = ['Guitar', 'Violin', 'Drums', 'Piano', 'Bass', 'Saxophone', 'Trumpet'
    , 'Maracas', 'Banjo', 'Harp']

# Solution:
at_least_three_vowels(instruments_list) # 5
```

1.3 Question 3

Make a function that takes in a string and a letter, then returns the same string but sets every lower case instance of the letter to upper case and every upper cease instance to lower case.

```
def switch_letter_size(sentence,letter):
    ...
# Example:
sentence = 'Programming Is Absolutely Awesome.'
letter = 'a'

# Solution:
switch_letter_size(sentence)
#  'ProgrAmming Is absolutely awesome.'
```

1.4 Question 4

Make a function that returns the following sum:

$$\sum_{n=4}^{k} \frac{2\pi n^3}{n-3}$$

```
def my_sum(k):
    ...
# Example:
k = 45

# Solution:
my_sum(k) # 219677.8898
```

1.5 Question 5

Make a function that takes in a positive integer and returns the list of positive squares that are inferior to it including that integer.

```
def square_numbers(n):
    ...
# Example:
n = 25

# Solution:
square_numbers(n) # [1, 4, 9, 16, 25]
```

1.6 Question 6

Make two functions. The first one takes two strings as input: a sentence and a file name then saves it as a txt file. The second function takes in a txt file and counts the number of letters in it.

```
def save_string_to_file(sentence, filename):
    ...
def count_letters_in_file(filename):
    ...
# Example:
sentence = "Hello, this is a test string!"
filename ="test_file.txt"

# Solution:
save_string_to_file(sentence, filename)
count = count_letters_in_file(filename) # 22
```

1.7 Question 7

Given a list of strings containing name of animals make a function that returns a dictionary with the number of each animal in the list.

```
def get_animal_dictionary(animal_list):
    ...
# Example:
animal_list = ['Dog', 'Tiger', 'Lion', 'Dolphin', 'Flamingo', 'Monkey', 'Lion', 'Lion',
    'Tiger', 'Dolphin']

# Solution:
get_animal_dictionary(animal_list)
# {'Dog': 1, 'Tiger': 2, 'Lion': 3, 'Dolphin': 2, 'Flamingo': 1, 'Monkey': 1}
```

1.8 Question 8

A statistician interested in football said that after studying multiple games the following is true: If a team scores less then seventeen goals in the season then they will get relegated. If they score more than fifty goals, then they will get promoted unless they scored less than four goals in three consecutive games. Given a list of goals scored by a team every game of a season, make a function that would output: "The team is relegated." if relegated, "The team is maintained." if maintained and "The team is promoted." if the team is promoted. Note: if for example you get the following list: `[0,1,2,0,1,0,1,0,0,0,3,4,0,3,1,0]` the sum of goals is 16. Therefore the team gets relegated.

```
def football_result(goals_list):
    ...
# Example:
goals_list = [2,7,2,3,5,0,1,2,4,6,3,3,4,2,3,2,6]

# Solution:
football_result(goals_list) # "The team is maintained."
```

1.9 Question 9

Given a string of a sentence, make a function that counts the number of consonants in it.

```
def consonants_count(sentence):
    ...
# Example:
sentence = "I didn't get this far just to get this far :)"

# Solution:
consonants_count(sentence)
# 23
```

1.10 Question 10

Make a class *Circle* that takes as parameters the center of the circle as a list: $[x_0, y_0]$ and the radius r. Then make a method named *get_area* that computes the area of the circle. Finally declare a new circle object and use the area method to obtain it's area:

```
class Circle:
    def __init__(...):
    ...
    def get_area(...):
    ...

# Example:
center = [10,5]
radius = 8

# Solution:
my_circle = Circle(center,radius)
my_circle.get_area()
# 201.06192
```

1.11 Question 11

In the world of mathematics, the Fibonacci numbers are very famous. They can be found with the following equation:

$$F(n) = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

Make a function that takes in a positive integer k and returns a list with the first k Fibonacci numbers.

```
def Fibonacci(k):
    ...
# Example:
k = 11

# Solution:
Fibonacci(k)
# [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

1.12 Question 12

Tom is the kind of guy that wears shorts every day of the year, unless he has an important event. He also likes to wear boots only when it snows, however due to fashion constraints, he only wears his boots with pants. Make a function that takes two boolean values as input: one for if Tom is going to an event and the other if it is snowing. The function should return a tuple of booleans: the first one is for if Tom will wear his shorts and the second is if he'll wear his boots.

```
def Tom_fashion(event,snowing):
    ...
# Example:
event = True
snowing = True

# Solution:
Tom_fashion(event,snowing)
# (False,True)
```

2 Medium Exercises

2.1 Question 1

A night guard walks around every night in the offices of a company to make sure that all employees have gone home, and that there are no intruders in the buildings. While on patrol, his boss has also asked him to check how many rooms still have the lights turned on. In fact the company needs to cut expenses and would like to avoid wasting energy and money on unused lighting.
In each room, if the combined energy used by the light bulbs is higher than 200 Watts/H, the room uses too much energy and should be noted down by the guard.
Given three lists, `room_nb` the room ids in the buildings, `bulb_type` the type of light bulb in the room in Watts/h, and `number_of_bulbs` the number of bulbs in the room, return a list with the room ids of the rooms that represent an energy wasting risk! If no room is above the limit, the function should return -1.

room number	110	112	114	131	132	133	245	246
bulb type	25	40	40	25	80	25	25	100
nb of bulbs	6	4	12	4	4	2	9	3

In the example above, room number 110 has 6 light bulbs with a power of 25 Watts/h, which gives a total of $6 \times 25 = 150$ Watts/h, which is below the limit of 200 Watts/h. However for room 246, that total goes to 300 Watts/h, which is too much!

```
1  def save_energy(room_nb, bulb_type, number_of_bulbs):
2      ...
3  # Example:
4  room_nb = [110,112,114,131,132,133,245,246]
5  bulb_type = [25,40,40,25,80,25,25,100]
6  number_of_bulbs = [6,4,12,4,4,2,9,3]
7
8  # Solution:
9  save_energy(room_nb, bulb_type, number_of_bulbs)
10 # [114, 132, 245, 246]
```

2.2 Question 2

Consider two inputs: `list_of_numbers` a list of numbers and `k` an integer. Assume that all numbers in the list are sorted in increasing order, and that all the numbers inside the list are positive integers. Find the smallest sublist from `list_of_numbers` whose product is bigger than `k`. You can assume that `k` is always larger than any number in the list.

`list_of_numbers = [1,2,3,4,5,6,8,9,10,11],k = 2345`

In the example above, to get to the number 2345, one needs to multiply $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 8 = 5760$ to get a number above 2345. However we need the smallest sublist, and that is attained by $8 \times 9 \times 10 \times 11 = 7920$, which only has 4 numbers! If the entire list product doesn't give a number higher than `k`, return -1.

```
1  def smallest_product(list_of_numbers,k):
2      ...
3  # Example:
4  list_of_numbers = [1,2,3,4,5,6,8,9,10,11]
5  k = 2345
6
7  # Solution:
8  smallest_product(list_of_numbers,k)
9  # returns [8,9,10,11]
```

2.3 Question 3

A swimming pool maintainer needs to figure out how much chlorine to put into his client's pool every hour. To do so he installs a device that measures the pH in the swimming pool every hour. If the pH is below or equal to 7 the pool is acidic, and the maintainer should put an amount equal to:

$$chlorine\_mass = \frac{0.015 * V}{pH} \tag{2.1}$$

However if the pH level is above 7, the water is too basic and the maintainer should add an amount equal to:

$$chlorine\_mass = 2 * \frac{0.037 * V}{pH} \tag{2.2}$$

in the equations, pH is the pH level in the water, V is the volume in cubic meters of water in the swimming pool. The $chlorine\_mass$ is in kg.
Given two inputs, `pH_level` a list with the hourly pH level, and an integer `pool_size` the volume in cubic meters of the pool, return the number of bags of chlorine necessary for the pool to be maintained throughout the day. The weight of a bag of chlorine is 12 kg.

```
def total_chlorine(pH_level, pool_size):
    ...
# Example:
pH_level = [7.6,8.4,5.5,12,6.7,8.2,7.2]
pool_size = 2998

# Solution:
total_chlorine(pH_level, pool_size)
# 13
```

2.4 Question 4

Two friends have come up with a secret code to send messages to each other without anyone else understanding! Each vowel is replaced with a number, and every space is replaced with "!" or "?" signs!

a	e	i	o	u	y
1	2	3	4	5	6

Given `code_path` the path to the file with the secret message, create a function that decodes the string and saves it one line below the secret message in the same file. For example, if the original txt file contains:

```
th3s?m2ss1g2!d4s2n't?m1k2?1n6!s2ns2.
this message doesn't make any sense.
```

```
def decoder(code_path):
    ...
# Example:
path/to/secret_code.txt # contains:
# l2ts?m22t?45ts3d2!3n?th3rt6!m3n5t2s.

# Solution:
decode(code_path)
path/to/secret_code.txt # now contains:
# l2ts?m22t?45ts3d2!3n?th3rt6!m3n5t2s.
# lets meet outside in thirty minutes.
```

2.5 Question 5 - Part 1

A traveler is planning out his next trip in the mountains, but would like to know how many peaks and valleys there are on his route. He will soon get a list of altitudes spread over his itinerary, and will be able to prepare for his trip. To help him out, create a function that takes one input `altitudes`, a list of all the altitudes on his itinerary, and returns a dictionary with two keys and values: the `nb_peaks` and the `nb_valleys`. For example, if his trip consists of:

$$[430, 706, 120, 500, 560, 780, 230]$$

The function should return: `{"nb_peaks":2, "nb_valleys":1}`

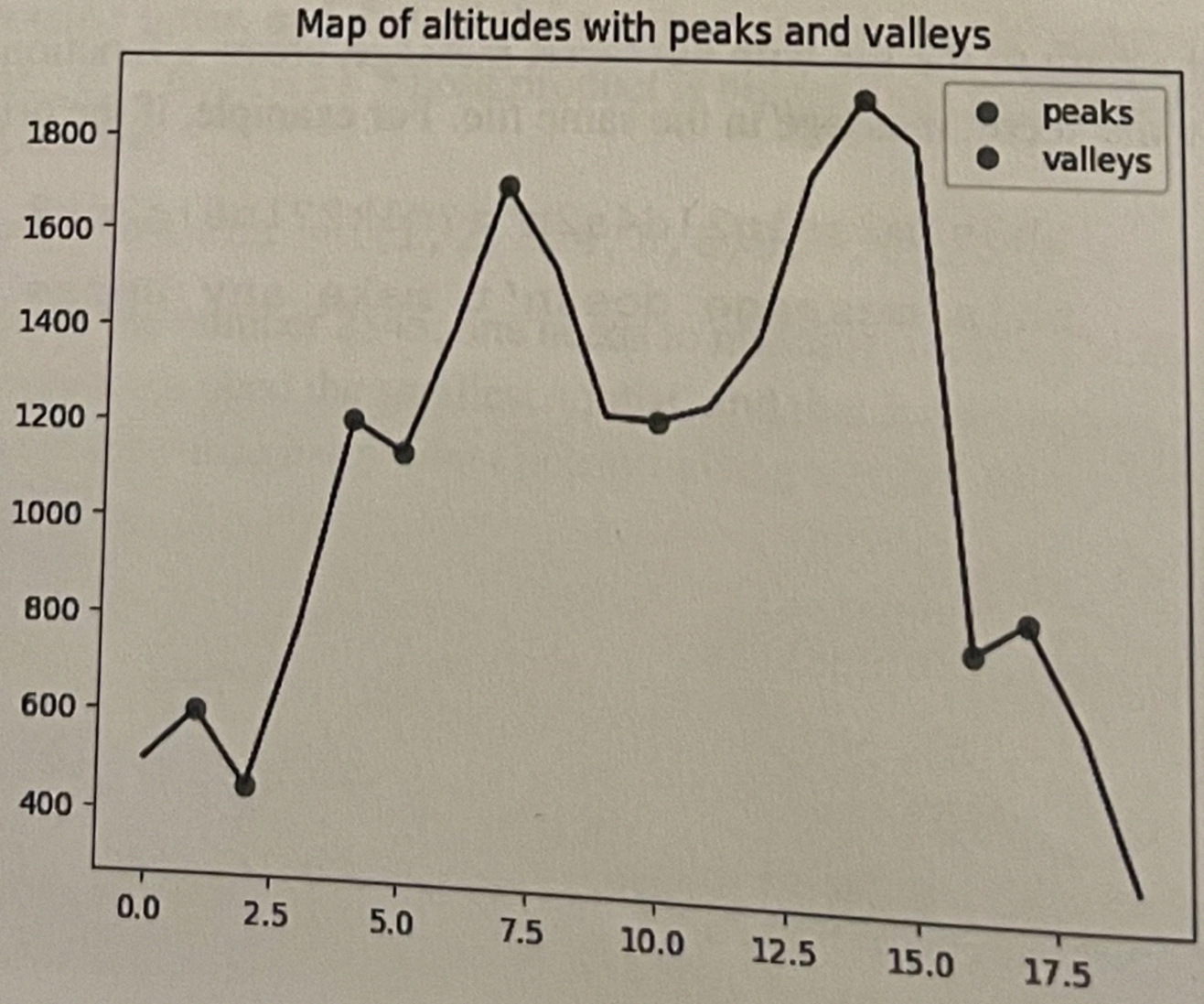
```
def mountains(altitudes):
    ...
# Example:
altitudes = [500,600,450,780,1203,1140,1400,1700,1530,1230,1224,1256,1400,1736,
             1895,1803,790,855,650,345]
# Solution:
mountains(altitudes)
# {'nb_peaks':5,'nb_valleys':4}
```

2.6 Question 5 - Part 2

The traveler is very grateful for the function that has been provided to him! He now knows how many peaks and valleys he will encounter during his trip and can prepare how much food and water he must bring on his voyage. However, He would also like to have a visual overview of the route! Create a function that takes one input `altitudes`, a list of all the altitudes on his itinerary, and plots the altitudes in a 2d plot.

To highlight the peaks and valleys, the traveler would also like the peaks to be visible with a red point on the plot, and the valleys visualized with a green point on the plot.

```
def map_altitude(altitudes):
    ...
# Example:
mountains = [500,600,450,780,1203,1140,1400,1700,1530,1230,1224,1256,1400,1736,
             1895,1803,790,855,650,345]
# Solution:
# plot with change in altitude (lines), points for peaks (red) and valleys (green).
```



3 Hard Exercises

3.1 Question 1

Write a Python program where you create a class `Vector2D` representing a 2D vector and overwrite the addition operator to add two vectors. The class should have two attributes, x and y, representing the x and y coordinates of the vector. Overwrite the `__add__` method to allow adding two Vector2D objects using the + operator. The addition should return a new Vector2D object whose x and y values are the sums of the x and y values of the two vectors being added. Add a `__str__` method to display the vector in the format "(x, y)".

```
class Vector2D:
    def __init__(...):
        ...

# Example:
v1 = Vector2D(1, 2)
v2 = Vector2D(3, 4)
v3 = v1 + v2

# Solution:
print(v3)
# (4, 6)
```

3.2 Question 2

The number of ice creams sold by the ice cream maker depends on two things, the weather and the number of ice creams sold the day before. If the temperature is equal to 25 degrees Celsius or above, the ice cream maker will sell $Sales(n) = \lceil (1.5 * Sales(n-1) - 5) \rceil$, but if the temperature is below 25 degrees Celsius, they will only sell $Sales(n) = \lfloor (0.9 * Sales(n-1) + 2) \rfloor$.
Given two inputs, a list of temperatures and the initial ice cream sales from the previous day, how many ice creams will the maker sell during that period (remember, do not count the `sales_init` in the period)?

```
def icecream_sales(temperature,sales_init):
    ...
# Example:
temperature = [17,26,28,15,18,24,26,26,27,19,13,20]
sales_init = 12

# Solution:
icecream_sales(temperature,sales_init)
# returns 231
```

3.3 Question 3

A scientist likes to organize their data into long strings. Inside the string the scientist has used commas to indicate separations between elements in a row, and semicolons to indicate the end of the row. The data should only contain integer and strings, but something wrong happened during the data recording and introduced some Booleans in the data. An input example is shown below, where the given string represents a 3×5 matrix.

`"54,2,'black',True,-3;54,1,'gray',99,-3;36,0,'black',0,-3"`

54	2	'black'	True	-3
54	1	'gray'	99	-3
36	0	'black'	0	-3

The objective of this exercise is to create a function that returns a list with the number of unique elements in each column of the data structure. However, if there is at least one Boolean or more in the column, the scientist cannot use that column with data and should indicate -1 in the list. In the example above, since the first column has two unique values, 54 and 36, the first element of the list is `2`. Repeating all columns, the output is `[2,3,2,-1,1]`.

```
def unique_elements(data):
    ...
# Example:
data = "54,2,'black',True,-3;54,1,'gray',99,-3;36,0,'black',0,-3"
# Solution:
unique_elements(data)
# returns [2,3,2,-1,1]
```

3.4 Question 4

A professor would like to make calculations for how well her class performed in different subjects in this semester's exams. She has a dictionary with students, their courses and their respective grades. However, there are some students that were absent for some exams, and are labeled 'DNA' for 'did not attend'. She would like to replace these values with the integer closest to the mean of the class.
Create a function that takes two inputs, the dictionary of students and a subject the professor is interested in knowing more about, and return the average of the class in that subject. You can assume that there is always at least one student in the class. Furthermore, if none of the students came to the exam, the function should return 'DNA'.

```
def grades(students,subject):
    ...
# Example:
students = {'Emma' : {'Math' : 7, 'Programming' : 10},
        'Thomas': {'Math' : 12, 'Programming' : 12},
        'Philip': {'Math' : 'DNA', 'Programming' : 2},
        'Judie': {'Math' : 2, 'Programming' : 0},
        'Mike': {'Math' : 12, 'Programming' : 'DNA'},
        'Daniel': {'Math' : 12, 'Programming' : 'DNA'},
        'Elizabeth': {'Math' : 10, 'Programming' : 12},
        'Ollie': {'Math' : 'DNA', 'Programming' : 7},
        'Cathy': {'Math' : 12, 'Programming' : 12},
        'Louise': {'Math' : 'DNA', 'Programming' : -3}
        }
# Solution:
subject = 'Programming'
grades(students,subjects) # 6.4
```

3.5 Question 5

The letters A and E are the most common in the English dictionary. You inherit a text file with many many words in it. Your task is to make a function that reads this text file and counts the number of words containing the letters A and E at the same time in it. Your function should save all results in a csv file called `english_ae_count` with the count of words from each rows. Each cell in the csv file should represent one row. Make it so the values are stored column wise in the csv file.

```
def count_of_words(filepath):
    ...
# Example:
# "./english_text.txt" with:
# table,banana,peace,coffee
# language,break,english,great
# dictionary,leather,seal,phone

# Solution:
# english_ae_count.csv with the values:
# 2
# 3
# 2
```

1 Exam

1.1 Question 1

Warren has an old sticky keyboard. He wrote some text, however he couldn't help but notice that sometimes the key of the equal sign was activating on it's own. Help him by making a function that would reconstruct his text by eliminating the equal symbols form his texts.

```
def Warren_helper(sentence):
    ...
# Example:
sentence = "=Hel=lo my so=n=, =it has been a whi=le=."

# Solution:
Warren_helper(sentence)
# "Hello my son, it has been a while."
```

1.2 Question 2

You want to study the dictionary in another language to find the shortest word that exists! Create a function that takes a list of words and returns the length of the shortest word.

```
def shortest_string(word_list):
    ...
# Example:
word_list = ['Bonjour', 'Bonsoir', 'Formidable', 'Genial', 'Imprimeur', 'Musculation', '
    Dejeuner', 'Tableau']

# Solution:
shortest_string(word_list) # 6
```

1.3 Question 3

A tournament has just been completed, where teams with male and female athletes played singles to win points and finish in 1st place! The data with points has been stored in a list of tuples, where each tuple represents a team with `(male_points,female_points,team_name)`.
The organiser would like to know which team came first, second and third. To do so he simply needs to add together the points for the male and female players in the same team, and find the top three combined scores. Create a function which takes as input `scores`, a list of tuples with the points for the males and females of each team, as well as the name of the team, and returns a tuple with the name of the top three teams in winning order (winner is the 1st element, runner up is the 2nd element etc.). One can assume that the points in a team can never be equal to another team, and that there will always be 3 or more teams.

```
def podium(scores):
    ...
# Example:
scores = [(10,12,'Lyngby'),(8,22,'Aarhus'),(26,23,'Helsingoer'),(10,13,'Roskilde')
    ,(16,4,'Randers'),(2,23,'Silkeborg'),(19,34,'Aalborg')]
# Solution:
podium(scores)
# ('Aalborg','Helsingoer','Aarhus')
```

1.4 Question 4

The M3 metro in Copenhagen is a relatively new line which goes in a full circle! There are 17 different stops on this line, with the corresponding names and travel times below:

F	AMH	NP	NR	NB	SP	VR	PHP	T	OE	M	KN	GS	R	KH	EP	FA
95	120	134	85	82	99	93	101	106	92	110	87	92	103	127	97	123

Table 1: Stops and travel times in the M3 line. The time under each stop is the time from that stop to the next (going right). For example it takes 85s to go from NR to NB.

The time from one station to the next is in **seconds**, and travelers can take the metro in both directions! You can assume that the time to go from in one direction or the other is the same. Going from NR (Nørrebros Runddel) to N (Nørrebro) or from N to NR takes 85s in both cases.
Given `stations` a string with all the metro stations separated by a space, `times` a list with the time in seconds it takes to go from one station to the next, `station1` and `station2` the start and destination respectively, find the fastest way to get to the desired location in Copenhagen using the M3 line.

```
def fastest_metro(stations,times,station1,station2):
    ...
# Example:
stations = 'F AMH NP NR N SP VR PHP T OE M KN GS R KH EP FA'
times = [95,120,134,85,82,99,93,101,106,92,110,87,92,103,127,97,123]
station1 = 'KH'
station2 = 'NP'

# Solution:
fastest_metro(stations,times,station1,station2)
# returns 562
```

1.5 Question 5

You need to help the developers of the new spider-man game. The task at hand is to handle the 3d coordinates of the spider-man. In order to do so, you need to make a `class` Spider-Man that has the initial 3d position of the character as a tuple as attribute. It should also have a method that takes series of command in a form of a list of strings ("right","left","forward","backward","jump","down"). Each command is a move in one step in the given direction:

- "right": would move the x coordinate one step higher.
- "left": would move the x coordinate one step lower.

The same applies for "forward" and "backward" in the y-direction and finally "jump" and "down" in the z-direction. Finally overwrite the string operator to display the current position of the spider-man.

```
class Spider_Man:
    def __init__(self, x, y, z):
        ...

    def update_position(self, list_of_commands):
        ...

    def __str__(self):
        ...
# Example:
Peter_Parker = Spider_Man(0, 0, 0)
list_of_commands = ["forward","right","jump","right","jump","right","forward","down","
    left","jump"]
Peter_Parker.update_position(list_of_commands)

# Solution:
print(Peter_Parker)
# (2, 2, 2)
```