

## ▼ Python Workshop 3 Exercises

Name: Akriti Kumari Dev University Id: 2329584

Examine and test the example code below, then design and write your code to answer the 7 questions that follow.

### ▼ Lists and indexing in Python

Consider a list of numbers: `nums = [90, 80, 70, 60, 50, 40, 30, 20, 10]`

90	80	70	60	50	40	30	20	10
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

Each character in the list has a notional position called [the index](#),

to access one of the elements, the name of the list with the index (in square brackets) must be provided E.g:

```
print(nums[2]) # index from the front => 70
```

In Python lists may also be [indexed from the back](#) by using a negative number for the index so

```
print(nums[-2]) # index from the back => 20
```

Of course where you use a number in a program, you can replace with a variable to produce interesting results

### ▼ Example 1

Write a Python function that returns the sum of two given vectors (represented as lists of the same length).

### ▼ Solution

```
def sumVectors(x, y): vect = len(x)*[0] # create a list of the same length as x, filled with 0s
i = 0 while i < len(x): vect[i] = x[i] + y[i] i += 1 return vect
```

Note that we have to create a vector of the required length before we can store the resulting values in it. Try a call such as `sumVectors([1,2,3], [6,5,4])` to test the working of the function.

### ▼ Example 2

Write a Python function that creates and returns a table with a given numbers of rows and columns, filled with None values.

### ▼ Solution

```
def buildTable(nRows, nCols): table = [] for r in range(nRows): table.append(nCols*[None]) return table
```

We start with an empty list of rows `table` and append a new row to it `nRows` times. `nCols*[None]` creates one row: a list of `nCols` elements, all set to `None`. *You may want to experiment with the example functions in the space below:*

Now try to write the Python code to answer the following questions in the cells below

1. Write and test a function that returns the index of the largest element in the list (or, if several elements have the largest value, the index of the first one of them). Write your own algorithm do not use built-in functions `max` or `index` or any library functions.

```
#Akriti Kumari Dev
```

```
def search_element(A, search_element):
```

```

    for i in range(len(A)):
        if A[i]==search_element:
            return i

def largest_element(A):
    max=A[0]
    for i in range(len(A)):
        if (A[i]>=max):
            max=A[i]
    result=search_element(A,max)
    return result

A=[100,24,99,211]
index=largest_element(A)
print(index)

3

```

2. Write a python script (ie. does not need to be a function) that swaps the values of three variables x, y, and z, so by the end x gets the value of y, y has the value of z, and z gets the value that was in x. *Note: the swapping can be done in a single line of code if you use good pythonic design.*

```

#Akriti Kumari Dev
#assigning the values to x,y and z
x=4
y=5
z=6
#swapping the values of x,y and z
x,y,z=y,z,x
print(x,y,z)

5 6 4

```

3. Pretend that list's reverse method does not exist and write your own function reverseList(lst) that reverses the list in place and returns a None. *Hints: proceed from both ends, but not too far..., the swapping (from Q2) should be useful and no need for a return statement.*

```

#Akriti Kumari Dev
def reverseList(lst):
    n=len(lst)
    for i in range(n//2):
        j=n-i-1
        lst[i],lst[j]=lst[j],lst[i]
lst=[4,5,6,7,8,9]
reverseList(lst)
print(lst)

[9, 8, 7, 6, 5, 4]

```

4. Write and test a function that takes a list with four or more elements and returns the index of the element that makes the greatest sum with its left and right neighbors. *Hint: a good solution here might be similar to your answer to Q1.*

```

#Akriti Kumari Dev
def maxsum_index(lst):
    max_sum=lst[0]+lst[1]+lst[2]
    maxsum_index=1
    for i in range(2,len(lst)-1):
        curr_sum=lst[i-1]+lst[i]+lst[i+1]
        if curr_sum>max_sum:
            max_sum=curr_sum
            maxsum_index=i
    return maxsum_index
lst=[3,4,5,6,7]
print(maxsum_index(lst))

2

```

5. Write a function called *intersection* that takes 2 lists of integers and returns a third list values that occur in both lists ie.

- ▼ *intersection*([2, 3, 4], [1, 2, 5]) should return [2]. Write your solution in pure Python to show the logic in your solution, do not use any library functions.¶

```
#Akriti Kumari Dev
def intersection(lst1,lst2):
    result=[]
    for values in lst1:
        if values in lst2 and values not in result:
            result.append(values)
    return result
lst1=[2,3,4]
lst2=[1,2,5]
intersection(lst1,lst2)
```

[2]

6. Let  $\vec{x} = (x_1, x_2, \dots, x_n)$  and  $\vec{y} = (y_1, y_2, \dots, y_n)$  be two n dimensional vectors. Their *dot product*, denoted  $\vec{x} \cdot \vec{y}$ , is defined

- ▼ as  $\vec{x} \cdot \vec{y} = x_1y_1 + x_2y_2 + x_3y_3 + \dots + x_ny_n$ . Write and test a Python function that returns the dot product of two given vectors (represented as lists of the same length).

```
#Akriti Kumari Dev
def dot_product(x,y):
    result=0
    for i in range(len(x)):
        result+=x[i]*y[i]
    return result
x=[1,2,3,4,5]
y=[4,5,6,7,8]
dot_product(x,y)
```

100

7. Write and test a Python function that returns the sum of the elements on the main diagonal (upper left to lower right) of a

- ▼ square matrix (represented as a list of lists eg. [[9, 8, 7], [6, 5, 4], [3, 2, 1]]). In linear algebra, this value is called the trace of the matrix. Note: your solution should work for any sized square matrix passed to it.

```
#Akriti Kumari Dev
def traced_matrix(matrix_A):
    if all(len(row)==len(matrix_A)for row in matrix_A):
        sum=0
        for i in range(len(matrix_A)):
            sum+=matrix_A[i][i]
        return sum
    else:
        print("Input value must be square")
matrix_A=[[9,8,7],[6,5,4],[3,2,1]]
traced_matrix(matrix_A)
```

15

## ▼ Downloading your workbook

Please note even if you are Jupyter on your computer (ie. not on a network) the Jupyter notebook runs as a server, so you are editing and running a file that is not easily accessed on your filing system. So to obtain your workbook (to use on a different computer or upload as an assignment, etc.) it must be downloaded to your file system. To do this...

Goto the **"File"** menu and select the **"Download as"** item. You can then select **"notebook (ipynb)"** to download.

