None
HW1 Solutions

Note:

1. Only one possible right answer is shown. All possible right answers will be given full credit.
2. Only the final solution is shown, and the details of actual code is not shown.
3. You may come to the office hours or the help sessions to discuss the HW solutions.
4. If you find any typos or issues, kindly contact your section instructor, or send a text @ **smujahid** on MS teams.

Table of Contents

Problem #A

```
----------------------------------------------Problem #A----------------------------------------------
A-1:

To create a list consider using LIST COMPREHENSION.

The given list is: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]
The required list is: [-1, -1.0, -3, -2.0, -5, -3.0, -7, -4.0, -9, -5.0]
------------------------------------------------------------------------------------------------------
A-2:

To create a list consider using LIST COMPREHENSION.
You can have Nested for statements in the list comprehension.

To create the required list, you may create the following list using List Comprehension:
 [(1, -1), (2, -2), (3, -3), (4, -4), (5, -5), (6, -6), (7, -7), (8, -8), (9, -9)]

 Use the above list to obtain the required following list using List Comprehension:
 [1, -1, 2, -2, 3, -3, 4, -4, 5, -5, 6, -6, 7, -7, 8, -8, 9, -9]
------------------------------------------------------------------------------------------------------
A-3:

To break the string into list of string use str.split() method.

The required list of contiguous words is:
 ['Vjg', 'dguv', 'rtqitcou', 'ctg', 'ytkvvgp', 'uq', 'vjcv', 'eqorwvkpi', 'ocejkpgu', 'ecp', 'rgthqto', 'vjgo'
, 'swkemn{0', 'Cnuq.', 'vjg', 'dguv', 'rtqitcou', 'ctg', 'ytkvvgp', 'uq', 'vjcv', 'jwocp', 'dgkpiu', 'ecp', 'w
pfgtuvcpf', 'vjgo', 'engctn{0', 'C', 'iqqf', 'guuc{kuv', 'cpf', 'c', 'iqqf', 'rtqitcoogt', 'jcxg', 'c', 'nqv',
'kp', 'eqooqp0']
------------------------------------------------------------------------------------------------------
A-4:

There were no \n in the list from A-3. However, the way you read the given string in A-3 may create \n charact
ers.

When you have \n characters, you can use str.replace() method.

The required list (without \n) is:
 ['Vjg', 'dguv', 'rtqitcou', 'ctg', 'ytkvvgp', 'uq', 'vjcv', 'eqorwvkpi', 'ocejkpgu', 'ecp', 'rgthqto', 'vjgo'
, 'swkemn{0', 'Cnuq.', 'vjg', 'dguv', 'rtqitcou', 'ctg', 'ytkvvgp', 'uq', 'vjcv', 'jwocp', 'dgkpiu', 'ecp', 'w
pfgtuvcpf', 'vjgo', 'engctn{0', 'C', 'iqqf', 'guuc{kuv', 'cpf', 'c', 'iqqf', 'rtqitcoogt', 'jcxg', 'c', 'nqv',
'kp', 'eqooqp0']
------------------------------------------------------------------------------------------------------
A-5:

Enumerating over words gives you the letters. Function ord() and chr() can be used to manipulate the letters.
For example, try:
chr(ord('b')-1).

The required list of words after ord/chr transformation is:
 ['The', 'best', 'programs', 'are', 'written', 'so', 'that', 'computing', 'machines', 'can', 'perform', 'them'
, 'quickly.', 'Also,', 'the', 'best', 'programs', 'are', 'written', 'so', 'that', 'human', 'beings', 'can', 'u
nderstand', 'them', 'clearly.', 'A', 'good', 'essayist', 'and', 'a', 'good', 'programmer', 'have', 'a', 'lot',
'in', 'common.']
```

```
-----------------------------------------------------------------------------------------------------
A-6:

To join list of strings into a single string use str.join() method.

The required string is:
 The best programs are written so that computing machines can perform them quickly. Also, the best programs ar
e written so that human beings can understand them clearly. A good essayist and a good programmer have a lot i
n common.
-----------------------------------------------------------------------------------------------------
A-7:

Consider using dictionaries when the scenario hints towards key-value pairs.

 To solve this task, the following lists were created:

[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
 ['X', 'XX', 'XXX', 'XL', 'L', 'LX', 'LXX', 'LXXX', 'XC', 'C']
 ['Ten', 'Twenty', 'Thirty', 'Forty', 'Fifty', 'Sixty', 'Seventy', 'Eighty', 'Ninety', 'Hundred']

 Then the following dictionary was created (you can have multiple connected dictionaries):

{10: ('X', 'Ten'), 20: ('XX', 'Twenty'), 30: ('XXX', 'Thirty'), 40: ('XL', 'Forty'), 50: ('L', 'Fifty'), 60: (
'LX', 'Sixty'), 70: ('LXX', 'Seventy'), 80: ('LXXX', 'Eighty'), 90: ('XC', 'Ninety'), 100: ('C', 'Hundred')}

Please enter a number from the following choices:  10, 20, 30, 40, 50, 60, 70, 80, 90, 100:  10
The roman numerals form is X, and the word form is Ten.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Problem #B

```
-----------------------------------------------Problem #B-----------------------------------------------
B-1:

Whenever you use random number in your code, consider using np.random.seed() method to set the seed. This will
help in reproducing the same random numbers.

The required random array is:
 [[6 6 6 4 0 7 2 3 7]
 [3 0 1 3 2 2 3 0 7]
 [8 6 6 0 4 7 7 8 7]
 [3 3 1 0 1 2 5 8 3]
 [7 2 3 0 7 1 6 1 5]
 [6 3 6 7 8 5 5 1 2]
 [2 5 5 1 7 4 2 4 8]
 [4 5 1 5 8 7 7 7 6]]
-----------------------------------------------------------------------------------------------------
B-2:

Numpy array has attributes, like array.dim, array.shape, array.size and many more, that can be handy during co
ding tasks.

The required information for S is:
Dimension: 2
    Shape: (8, 9)
     Size: 72
-----------------------------------------------------------------------------------------------------
B-3:

Numpy array indexing uses single square brackets with two inputs separated by comma: [input1,input2] where inp
ut1 corresponds to rows, and input2 corresponds to columns.

The required row elements are:
[ 4 25 25  1 49 16  4 16 64]
```

B-4:

Numpy array indices: [:,input2] indicates all rows, and selected columns based on input2.

The second column from the end always has index as -2 (or for this particular case index as 7 can be used too)
.

The required elements are:
[0.3 0. 0.8 0.8 0.1 0.1 0.4 0.7]

B-5:

Numpy array index inputs can be of: start:endBefore:steps type, where steps shows how to move from one index t
o another index.

The required alternate row elements from row index 1 are:
[[3 0 1 3 2 2 3 0 7]
 [3 3 1 0 1 2 5 8 3]
 [6 3 6 7 8 5 5 1 2]
 [4 5 1 5 8 7 7 7 6]]

B-6:

Numpy array index inputs can be of: start:endBefore:steps type, where steps shows how to move from one index t
o another index.

The required sliced elements starting from column index 2 are:
[[6 0 2 7]
 [1 2 3 7]
 [6 4 7 7]
 [1 1 5 3]
 [3 7 6 5]
 [6 8 5 2]
 [5 7 2 8]
 [1 8 7 6]]

B-7:

Numpy array indices: start:endBefore:steps type, can have negative steps.

The required reversed rows are:
[[4 5 1 5 8 7 7 7 6]
 [2 5 5 1 7 4 2 4 8]
 [6 3 6 7 8 5 5 1 2]
 [7 2 3 0 7 1 6 1 5]
 [3 3 1 0 1 2 5 8 3]
 [8 6 6 0 4 7 7 8 7]
 [3 0 1 3 2 2 3 0 7]
 [6 6 6 4 0 7 2 3 7]]

B-8:

Numpy array indices: start:endBefore:steps type, can have negative steps.

The required reversed columns are:
[[7 3 2 7 0 4 6 6 6]
 [7 0 3 2 2 3 1 0 3]
 [7 8 7 7 4 0 6 6 8]
 [3 8 5 2 1 0 1 3 3]
 [5 1 6 1 7 0 3 2 7]
 [2 1 5 5 8 7 6 3 6]
 [8 4 2 4 7 1 5 5 2]
 [6 7 7 7 8 5 1 5 4]]

B-9:

Numpy array indexing uses single square brackets with two inputs separated by comma: [input1,input2] where input1 and input2 can be list of indices.
Having both inputs as list of indices (same size list) will result in selecting corresponding elements.
Having one of the inputs as list of indices will result in selecting entire rows/columns.

```
The required slice is:
[[0 7 6]
 [0 6 2]
 [7 5 3]]
```

B-10:

Numpy methods like split and stack (or insert) can be used for this task.

Note copying full aray can be done using .copy() method.

```
The given nd-array S is:
[[6 6 6 4 0 7 2 3 7]
 [3 0 1 3 2 2 3 0 7]
 [8 6 6 0 4 7 7 8 7]
 [3 3 1 0 1 2 5 8 3]
 [7 2 3 0 7 1 6 1 5]
 [6 3 6 7 8 5 5 1 2]
 [2 5 5 1 7 4 2 4 8]
 [4 5 1 5 8 7 7 7 6]]

The new & updated nd-array P is:
[[ 6  6  6  4 -9  7  2  3  7]
 [ 3  0  1  3 -7  2  3  0  7]
 [ 8  6  6  0 -5  7  7  8  7]
 [ 3  3  1  0 -8  2  5  8  3]
 [ 1  1  1  1  1  1  1  1  1]
 [ 7  2  3  0 -2  1  6  1  5]
 [ 6  3  6  7 -1  5  5  1  2]
 [ 2  5  5  1 -2  4  2  4  8]
 [ 4  5  1  5 -1  7  7  7  6]]
```

~-~~