In [3]:

```
import numpy as np
```

Q-1 Create a null vector of size 10 (zeros array) but the fifth value which is 1.

In [4]:

```
#Code here
a=np.zeros(10)
a[4]=1
a
```

Out[4]:

```
array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])
```

Q-2 Write a function to create a 2d array with 1 on the border and 0 inside. Take 2-D array shape as (a,b) as parameter to function.

hint Use slicing technique to replace 0 with 1 in zeros matrix or nested for loops

Eg.-

```
[[1,1,1,1],
[1,0,0,1],
[1,0,0,1],
[1,1,1,1]]
```

In [4]:

```
#Code here
```

In [ ]:

Q-3: Arrays

You are given a space separated list of numbers. Your task is to print a reversed NumPy array with the element type float.

Input Format:

A single line of input containing space separated numbers.

Output Format:

Print the reverse NumPy array with type float.

Example 1:

Input:

```
1 2 3 4 -8 -10
```

Output:

```
[-10.  -8.   4.   3.   2.   1.]
```

In [ ]:

`Q-4:` Softmax function

Create a Python function to calculate the Softmax of the given numpy 1D array. The function only accepts the numpy 1D array, otherwise raise error.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=i}^{K} e^{z_j}}$$

https://en.wikipedia.org/wiki/Softmax_function

Example 1:

Input:

```
[86.03331084 37.7285648  48.64908087 87.16563062 38.40852563 37.20006318]
```

Output:

```
[2.43733249e-01, 2.56112115e-22, 1.41628284e-17, 7.56266751e-01,
       5.05514197e-22, 1.50974911e-22]
```

Example 2:

Input:

```
[33.17344305 45.61961654 82.05405781 80.9647098  68.82830233 91.52064278]
```

Output:

```
[4.57181035e-26, 1.16249923e-20, 7.73872596e-05, 2.60358426e-05,
       1.39571531e-10, 9.99896577e-01]
```

In [ ]:

```

```

`Q-5` Answer below asked questions on given array:

1. Fetch Every alternate column of the array
2. Normalise the given array

https://en.wikipedia.org/wiki/Normalization_(statistics)

There are different form of normalisation for this question use below formula.

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

`arr1=np.random.randint(low=1, high=10000, size=40).reshape(8,5)`

In [ ]:

```python
# Given
arr1=np.random.randint(low=1, high=10000, size=40).reshape(8,5)
arr1
```

```
Out[ ]: array([[5004, 6405, 1638, 2845, 3510],
               [6966, 3244, 2711, 4785, 7351],
               [ 689, 8978,  572, 3661,  245],
               [5895, 4114, 4716, 2627, 4822],
               [ 994, 2038, 5624, 5577, 2819],
               [9362, 7754, 7941, 4262, 2240],
               [5707, 5418, 9980,  255, 4026],
               [8057, 3557, 8667, 7068, 8319]])
```

```
In [7]: #Output
        """
        Normalized Array:
         [[0.62896664 0.25437347 0.80461758 0.41405614 0.20829943]
         [0.72975997 0.36950773 0.58696094 1.         0.4310415 ]
         [0.13374695 0.59143613 0.23535395 0.20341741 0.92890561]
         [0.80543124 0.76688365 0.66924329 0.31468674 0.44711147]
         [0.04759967 0.54485354 0.88537429 0.48179414 0.55522783]
         [0.81519528 0.09947111 0.21196094 0.26932465 0.        ]
         [0.62754272 0.80288853 0.57790887 0.34855574 0.33238405]
         [0.00101709 0.92707486 0.64524003 0.32262002 0.77542718]]"""
```

```
Out[7]: '\nNormalized Array:\n [[0.62896664 0.25437347 0.80461758 0.41405614 0.20829943]\n [0.72975997 0.36950773 0.586960
        94 1.         0.4310415 ]\n [0.13374695 0.59143613 0.23535395 0.20341741 0.92890561]\n [0.80543124 0.76688365 0.66
        924329 0.31468674 0.44711147]\n [0.04759967 0.54485354 0.88537429 0.48179414 0.55522783]\n [0.81519528 0.09947111
        0.21196094 0.26932465 0.        ]\n [0.62754272 0.80288853 0.57790887 0.34855574 0.33238405]\n [0.00101709 0.92707
        486 0.64524003 0.32262002 0.77542718]]'
```

```
In [13]: #Code here
         #1
```

```
In [15]: #2
```

`Q-6:` Find the nearest element in the array to a given integer.

```
Ex:-
a=23 and array - [10 17 24 31 38 45 52 59].
Nearest element is 24
```

Hint: Read about this function `argmin()` and abs()

In [ ]:

In [ ]:

`Q-7 Write a NumPy program

- to find the missing data in a given array. Return a boolean matrix.
- also try to fill those missing values with 0. For that, you can use `np.nan_to_num(a)`

```
import numpy as np

np.array([[3, 2, np.nan, 1],
          [10, 12, 10, 9],
          [5, np.nan, 1, np.nan]])
```

In [ ]:

In [ ]: