

# Numpy Assignment no #02, AIC\_Quarter #02, PIAIC

```
In [1]: import numpy as np
```

## Task 01

```
In [27]: # Task no 1
def function1():
    # create 2d array from 1,12 range
    # dimension should be 6row 2 columns
    # and assign this array values in x values in x variable
    # Hint: you can use arange and reshape numpy methods
    x = np.arange(1,13).reshape((6,2))
    return x

print(function1())
```

```
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]]
```

## Task 02

```
In [26]: #Task 02
def function2():
    #create 3D array (3,3,3)
    #must data type should have float64
    #array value should be satart from 10 and end with 36 (both included)
    # Hint: dtype, reshape
    x = np.arange(10,37,dtype=np.float64).reshape((3,3,3))
    return x

function2()
```

```
Out[26]: array([[10., 11., 12.],
                [13., 14., 15.],
                [16., 17., 18.]],

               [[19., 20., 21.],
                [22., 23., 24.],
                [25., 26., 27.]],

               [[28., 29., 30.],
                [31., 32., 33.],
                [34., 35., 36.]])
```

## Task 03

```
In [40]: #Task 03
def function3():
    #extract those numbers from given array. those are must exist in 5,7 Table
    #example [35,70,105,..]
    a = np.arange(1, 100*10+1).reshape((100,10))
    #print(a)
    x=a[(a%5==0) & (a%7==0)]
    return x

print(function3())
```

```
[ 35  70 105 140 175 210 245 280 315 350 385 420 455 490 525 560 595 630
 665 700 735 770 805 840 875 910 945 980]
```

## Task 04

```
In [24]: #Task 04
def function4():
    #Swap columns 1 and 2 in the array arr.
    arr = np.arange(9).reshape(3,3)
    #print(arr)

    b=(arr[:,[1,0,2]])
    return b

function4()
```

```
Out[24]: array([[1, 0, 2],
               [4, 3, 5],
               [7, 6, 8]])
```

## Task 05

```
In [6]: #Task 05
def function5():
    #Create a null vector of size 20 with 4 rows and 5 columns with numpy function
    a= np.full((4,5),0)
    return a

#second way to do same thing

#a=np.zeros((4,5),dtype=int)
#a

function5()
```

```
Out[6]: array([[0, 0, 0, 0, 0],
               [0, 0, 0, 0, 0],
               [0, 0, 0, 0, 0],
               [0, 0, 0, 0, 0]])
```

## Task 06

```
In [23]: #Task 06
def function6():
    # Create a null vector of size 10 but the fifth and eighth value which is 10,
    a= np.zeros(10,dtype=int)
    print(a)
    a[4]=10
    a[7]=20
    return a

function6()
```

```
[0 0 0 0 0 0 0 0 0 0]
```

```
Out[23]: array([ 0,  0,  0,  0, 10,  0,  0, 20,  0,  0])
```

## Task 07

```
In [22]: #Task 07
def function7():
    # Create an array of zeros with the same shape and type as X. Dont use reshape
    #x = np.arange(4, dtype=np.int64)
    #x = np.zeros(4, dtype=np.int64)

    #or

    x= np.zeros(4, dtype="int64")
    return x

function7()
```

```
Out[22]: array([0, 0, 0, 0])
```

## Task 08

```
In [21]: #Task 08
def function8():
    # Create a new array of 2x5 uints, filled with 6.
    a=np.full((2,5),6, dtype="int32")
    return a

function8()
```

```
Out[21]: array([[6, 6, 6, 6, 6],
                [6, 6, 6, 6, 6]], dtype=int32)
```

## Task 09

```
In [20]: #Task 09
def function9():
    # Create an array of 2, 4, 6, 8, ..., 100.
    a=np.arange(2,101,2)
    return a

function9()
```

```
Out[20]: array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26,
                28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52,
                54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78,
                80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100])
```

## Task 10

```
In [11]: #task10
def function10():
    # Subtract the 1d array brr from the 2d array arr, such that each item of brr
    # is subtracted from each row of arr.

    arr = np.array([[3,3,3],[4,4,4],[5,5,5]])
    brr = np.array([1,2,3])
    subt = arr.T-brr
    return subt

#or
#subt = np.subtract(arr.T, brr)
#subt

function10()
```

```
Out[11]: array([[2, 2, 2],
                [2, 2, 2],
                [2, 2, 2]])
```

## Task 11

```
In [19]: #Task11
def function11():
    # Replace all odd numbers in arr with -1 without changing arr.

    arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    ans=np.where(arr%2==1,-1,arr)
    return ans

function11()
```

```
Out[19]: array([ 0, -1,  2, -1,  4, -1,  6, -1,  8, -1])
```

```
In [13]: #Task12
def function12():
    # Create the following pattern without hardcoding. Use only numpy functions
    # HINT: use stacking concept

    arr = np.array([1,2,3])
    ans=np.hstack((arr.repeat(3), np.tile(arr,3)))
    return ans

    #or

    #ans=np.r_[np.repeat(arr,3), np.tile(arr,3)]
    #ans

#or, we can use this, but ans is without commas

    #arr1=np.repeat(arr,3)
    #print(arr1)

    #arr2=np.tile(arr,3)
    #print(arr2)

    #ans=np.concatenate((arr1,arr2),axis=0)
    #print(ans)

function12()
```

```
Out[13]: array([1, 1, 1, 2, 2, 2, 3, 3, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3])
```

## Task 13

```

In [14]: #Task13
def function13():
    # Set a condition which gets all items between 5 and 10 from arr.

    arr = np.array([2, 6, 1, 9, 10, 3, 27])
    arr1=np.extract((5<arr) & (arr<10), arr)
    return arr1

    #or

    #arr1=arr[np.all([arr>5,arr<10], axis= 0)]
    #arr1

    #or
    #arr1=arr[(arr>5) & (arr<10)]
    #arr1

function13()

```

Out[14]: array([6, 9])

## Task 14

```

In [17]: #Task 14
def function14():
    # Create an 8X3 integer array from a range between 10 to 34 such that the dig
    # Hint use split method

    arr= np.arange(10, 34, 1).reshape(8,3)
    a=np.split(arr, 4, axis=0)
    return a

function14()

```

Out[17]: [array([[10, 11, 12],  
[13, 14, 15]]),  
array([[16, 17, 18],  
[19, 20, 21]]),  
array([[22, 23, 24],  
[25, 26, 27]]),  
array([[28, 29, 30],  
[31, 32, 33]])]

## Task 15

```
In [18]: #Task 15
def function15():
    #Sort following NumPy array by the second column

    arr = np.array([[ 8,  2, -2],[-4,  1,  7],[ 6,  3,  9]])
    sorted_array = arr[np.argsort(arr[:, 1])]
    return sorted_array

#or

#arr_sort = arr[arr[:,1].argsort()]
#arr_sort

function15()
```

```
Out[18]: array([[ -4,  1,  7],
               [  8,  2, -2],
               [  6,  3,  9]])
```

## Task 16



```

In [32]: #Task 16
def function16():
    #Write a NumPy program to join a sequence of arrays along depth.
    #x = np.array([[1], [2], [3]])
    #y = np.array([[2], [3], [4]])
    """
    Expected Output:
        [[1 2]]
        [[2 3]]
        [[3 4]]
    """

    x = np.array([[1], [2], [3]])
    y = np.array([[2], [3], [4]])
    ans = np.concatenate((x,y)).reshape((3,1,-1) , order="F")
    return ans

#or

#z=np.dstack((x,y))    (Ans with commas)
#z

#or

#z=np.concatenate([x,y],axis=1)    (Ans with commas)
#z

print(function16())

```

```

[[[1 2]]
 [[2 3]]
 [[3 4]]]

```

## Task 17

```

In [33]: #Task 17
def function17():
    #replace numbers with "YES" if it divided by 3 and 5
    # otherwise it will be replaced with "NO"
    # Hint: np.where

    arr = np.arange(1,10*10+1).reshape((10,10))
    #print(arr)

    b=arr[(arr%3==0) & (arr%5==0)]
    print(b)
    c=np.where(arr%15 == 0, "Yes", "No")
    return c

function17()

```

```
[15 30 45 60 75 90]
```

```

Out[33]: array([[ 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No'],
 [ 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No'],
 [ 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes'],
 [ 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No'],
 [ 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No'],
 [ 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes'],
 [ 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No'],
 [ 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No'],
 [ 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes'],
 [ 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No']],
 dtype='<U3')

```

## Task 18

```
In [34]: #Task 18
def function18():
    #count values of "students" are exist in "piaic"

    piaic = np.arange(100)
    students = np.array([5,20,50,200,301,7001])

    #1          to know which values are present

    #values_present=piaic[np.in1d(piaic,students)]
    #print(values_present)

    #2          to count how many values are present?

    #a=np.count_nonzero(students < 100)
    #a

    #3          we can also use these

    #b=piaic[np.in1d(piaic,students)].size
    #b

    #4

    c=np.intersect1d(piaic,students).size
    return c

    #or

    #5
    #x=np.count_nonzero(np.isin(students, piaic))
    #x

function18()
```

Out[34]: 3

## Task 19

```
In [36]: #Task19
def function19():

    #Create variable "X" from 1,25 (both are included) range values
    #Convert "X" variable dimension into 5 rows and 5 columns
    #Create one more variable "W" copy of "X"
    #Swap "W" row and column axis (like transpose)
    # then create variable "b" with value equal to 5
    # Now return output as "(X*W)+b:
    X = np.arange(1,26).reshape(5,5)
    print(X)
    w=X.T
    print(w)
    b=5
    output=(X*w)+b
    return output

    #or may be

    #w = np.array(X).swapaxes(1,0)
    #or
    #w = np.copy(X).T
    #or
    #w = X.copy().transpose()

function19()
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
[[ 1  6 11 16 21]
 [ 2  7 12 17 22]
 [ 3  8 13 18 23]
 [ 4  9 14 19 24]
 [ 5 10 15 20 25]]
```

```
Out[36]: array([[ 6, 17, 38, 69, 110],
 [ 17, 54, 101, 158, 225],
 [ 38, 101, 174, 257, 350],
 [ 69, 158, 257, 366, 485],
 [110, 225, 350, 485, 630]])
```

## Task 20

```
In [39]: #Task20
def function20():
    #apply fuction "abc" on each value of Array "X"
    x = np.arange(1,11)
    def abc(x):
        return x*2+3-2
    return abc(x)

function20()
```

```
Out[39]: array([ 3,  5,  7,  9, 11, 13, 15, 17, 19, 21])
```

```
In [ ]:
```