HONG KONG INSTITUTE OF VOCATIONAL EDUCATION

**Laboratory 5: NumPy for Data Analysis**

**TASKS :**

1. Write Python code with NumPy to implement to following tasks:
   1. Check the version of NumPy.

import numpy as np  
np.\_\_version\_\_

* 1. Convert a list storing integers from 1 to 100 into an array, “arr”.

arr = np.arange(1, 101)  
arr

* 1. Extract all the even numbers in “arr” to form “arr1”.

arr1 = arr[arr%2 == 0]  
arr1

* 1. Extract all the number ending with a zero or a five digit in “arr” to form “arr2”.

arr2 = arr[arr%5 == 0]  
arr2

1. Write Python code with NumPy to implement to following tasks:
   1. Create an arbitrary zero-dimensional array called “arr”.

arr = np.array(99)  
arr

* 1. Create a one-dimensional array called "arr2" with length 18, storing integers from 51 to 68.

arr2 = np.arange(51, 69)  
arr2

* 1. Reshape the array “arr2” as a 3×6 array called “arr3”.

arr3 = arr2.reshape(3, 6)  
arr3

* 1. Reverse the order of the columns in “arr3” to form an array called “arr4”.

arr4 = arr3[::,::-1]  
arr4

1. Write Python code with NumPy to implement to following tasks:
   1. Create a 3×4 array called “wed”, storing all the dates of Wednesday from 5 June 2019 to 28 August 2019.

wed = np.arange(np.datetime64('2019-06-05'), np.datetime64('2019-08-28'),7).reshape(3,4)  
wed

* 1. Take alternate elements from “wed” to form a 2×3 array called “wed2”.

wed2 = wed.flatten()[::2].reshape(2,3)  
wed2

* 1. Form an array “wed3” by extracting the second and fourth columns from “wed”.

wed3 = wed[:,[1,3]]  
wed3

* 1. Extract the diagonal elements from “wed” to form one-dimensional array “wed4”. Note that you are NOT allowed to use the diagonal function of numpy arrays.

wed4 = wed.flatten()[::wed.shape[1]+1]  
wed4

1. Given the following array,

num = np.array([5, 2, 1, 1, 3, np.nan, 4, 3, 1, 4, 3, 1, 2, np.nan, np.nan, 1, 1, 2])

Write Python code with NumPy to implement to following tasks:

1. Find the index of last repetition of number 1 in num. Hint: You may use np.where.

np.where(num == 1)[0][-1]

1. Count of number of occurrences of np.nan in num.

np.where(np.isnan(num))[0].size

1. Remove all the nan values in num to form another array, “num2”.

idx = np.where(np.isnan(num))[0]  
num2 = np.delete(num, idx)  
num2

1. Take a square root of each value in num to form another array, “num3”. You are NOT allowed to use math.sqrt function.

um3 = num \*\* 0.5  
num3

1. Write Python code with NumPy to implement to following tasks:
   1. Use two different examples of two-dimensional arrays to demonstrate *fancy indexing*.

# Task 5a - First Example  
arr1 = np.arange(11, 23).reshape(3, 4)  
arr1  
#%%  
# Extract the first and last rows from arr1  
arr1[[0, -1]]  
#%%  
# Task 5a - Second Example  
arr2 = np.arange(101, 117).reshape(4, 4)  
arr2  
#%%  
# Extract the second and third rows from arr2  
arr2[:, [1, 2]]

* 1. Create a 2×3×4 array “arr” with randomly generated numbers from 1 to 24. Use transpose function on this array to mimic T function.

from random import randint  
arr = np.array([randint(1,25) for n in range(24)]).reshape(2,3,4)  
arr  
#%%  
arr.T  
#%%  
arr.transpose(2,1,0)

* 1. You are given an array stored in the file "Lab5.5c.npy”. Retrieve the array into “farr” and then reverse the order of the elements in “farr”.

farr = np.load('Lab5.5c.npy')  
farr  
#%%  
farr2 = farr[::-1, ::-1]  
farr2

* 1. Create a 2×3×4 array of randomly generated floats and use numpy.save to store the array into a file.

rom numpy.random import rand  
arr3 = rand(2, 3, 4)  
arr3  
#%%  
np.save('Lab5.5d', arr3)

1. Compare the results of using numpy.matrix object and numpy.array object to solve the following set of linear equations:

A = np.matrix([[1,2,3,-2], [3,2,-1,2], [2,-1,-2,-3], [2,-3,2,1]])  
B = np.matrix([[6], [4], [2], [8]])  
X = A\*\*(-1) \* B  
X

1. Write Python code with NumPy to implement to following tasks:
2. Create two different arrays, r1 and r2, of 10 randomly generated integers between 0 to 9 inclusively.

r1 = np.array([np.random.randint(10) for \_ in range(10)])  
r2 = np.array([np.random.randint(10) for \_ in range(10)])

1. Write a function, maxx that returns the maximum value between its two integer arguments.

def maxx(x, y):  
 return x if x >= y else y

1. Convert the function maxx that works on two scalars, to work on two arrays. Name the new function as pair\_max.

pair\_max = np.vectorize(maxx, otypes=[int])

1. Apply the function pair\_max to arrays r1 and r2 for extracting the maximum values from pairwise comparisons on these arrays.

pair\_max(r1, r2)

1. Given an array of floats in file “iris2d.npy”. Write Python code with NumPy to implement to following tasks:
2. Load the file into an array named “iris” and show the dimensions of the array.

iris = np.load('iris2d.npy')  
iris.shape

1. Create a new array iris2 by extracting the first 25 rows and the first four columns from iris and then reshaping it into a one-dimensional array.

iris2 = iris[:25,:-1].flatten()  
iris2.shape

1. Randomly assign several np.nan values to the array iris2 and then count the number of np.nan values in the array.

iris2[np.random.randint(50, size=20)] = np.nan  
iris2

1. Replace all np.nan values in iris2 by 999.

np.set\_printoptions(suppress=True)  
iris2[np.isnan(iris2)] = 999  
iris2