PYTHON MINOR ASSIGNMENT- 6 (NUMPY AND PANDAS)

```
'''Q1) Create a 2x3 array of ones, a 3x3 array of zeros, and a 2x5 array filled with sevens.'''

import numpy as np
arr1 = np.ones((2, 3))#np.ones(shape):creates an array filled with the value 1.0
arr2 = np.zeros((3, 3))#np.zeros(shape): array filled with the value 0.0
arr3 = np.full((2, 5), 7)#np.full(shape, value):array filled with a specified value (7 in this case)
print(arr1)
print(arr2)
print(arr3)
```

```
'''Q2) Use broadcasting to:
Cube each element.
Add 7 to each element.
Multiply each element by 2.'''

import numpy as np
arr=np.arange(1,5).reshape(2,2)#arange: is used to set a range:(start,end-1) and reshapde for 2*2 array
cube=arr**2 #will mul 2 to every element in the array.
add=arr+7 # similarly will add 7 to every element
prod=arr*2 #mul 2 to every element
print(arr)
print(cube , add , prod)
```

```
'''Q3) Create arrays and perform element-wise multiplication.'''

import numpy as np
arr1=np.arange(1,10).reshape(3,3) #range: 1 - 9 and 3*3 matrix
arr2=np.arange(1,10).reshape(3,3) # same
mul=arr1*arr2
print(mul)
```

```
'''Q4) Task: Create a 2D array and swap the first two rows and columns.'''

import numpy as np
arr=np.array([[1,2,3],[4,5,6]])
print(arr) #Before
arr[[0,1],:]=arr[[1,0],:] #Rows 0,1 = 1,0
arr[:,[0,1]]=arr[:,[1,0]] #Column 1,0 = 0,1
print(arr) #After
```

```
'''Q5) Use two lists to create a 2x5 array.'''

import numpy as np
arr=np.array([[1,2,3,4,5],[6,7,8,9,10]])
print(arr)
```

```
'''Q6) Create an array of powers of 2, then flatten it using flatten and ravel.'''

import numpy as np
arr=2**np.array([[1,2,3],[4,5,6]])
flattened = arr.flatten() #Creates a new 1D array that is a copy of the original array , Changes do not
affect the original.
raveled = arr.ravel() #Creates a view (if possible) of the original array in 1D format., Changes can
affect the original (if view).
print(arr)
print(flattened)
print(raveled)
```

```
'''Q7) Task: Find the most frequent values in an array.'''

import numpy as np
from collections import Counter
arr = np.array([1,2,2,2,2,3,4,4,5])
value, frequency = Counter(arr).most_common(1)[0] #(1): means we want the single most common element
print("Value:",value,"Frequency: ",frequency) #[0]: to extract the first tuple from the list
```

```
'''Q8) Use linspace to create an array and reshape it to a 2D array, then convert it to integers.'''

import numpy as np
arr= np.linspace(1.1, 6.6, 6).reshape(2, 3)#np.linspace(st,stop,num)creates an array of num evenly spaced
values between start and stop (inclusive).
int_arr = arr.astype(int)
print(int_arr)
```

```
'''Q9)Write a function format_2d_array (arr) that takes a two-dimensional array of positive integers
(represented as nested lists) and returns a formatted string that mimics NumPy's column-based format. In
this format, each element in the array should be right-aligned, and the width of each column should be
determined by the number of characters required to display the largest element in the array.'''

import numpy as np
def format_2d_array(arr):
    return "\n".join(" ".join(f"{x:3}" for x in row) for row in arr)#x:3 for right alignment ,
arr=np.array([[1,2,3],[4,5,6]]) .join: for single string
format=format_2d_array(arr)
print(format)
```

```
'''10.Create an array containing the values 1-15, reshape it into a 3-by-5 array, then use indexing and
slicing techniques to perform:'''

import numpy as np
arr=np.arange(1,16).reshape(3,5)
print(arr[1]) #(a)Select row 2.
print(arr[:,4]) #(b)Select column 5.
print(arr[:2]) #(c)Select rows 0 and 1.
print(arr[1,4]) #(d)Select columns 2-4.
print(arr[:, 2:5]) #(e)Select the element that is in row 1 and column 4.
print(arr[1:, [0, 2, 4]]) #(f) Select all elements from rows 1 and 2 that are in columns 0, 2 and
```

```
Q11) import numpy as np

array1 = np.array([[0, 1], [2, 3]])
array2 = np.array([[4, 5], [6, 7]])

# (a)Use vertical stacking to create a 4-by-2 array named array 3, with arrayl stacked on top of array2.
array3 = np.vstack((array1, array2))
print(array3)

# (b)Use horizontal stacking to create a 2-by-4 array named array4, with array2 to the right of arrayl.
array4 = np.hstack((array1, array2))
print(array4)

# (c)Use vertical stacking with two copies of array4 to create a 4-by-4 array named array 5.
array5 = np.vstack((array4, array4))
print(array5)

# (d)Use horizontal stacking with two copies of array3 to create a 4-by-4 array named array6.
array6 = np.hstack((array3, array3))
print(array6)
```

```
'''12.  Use NumPy's concatenate Function to reimplement the previous problem.'''
import numpy as np
array1 = np.array([[0, 1], [2, 3]])
array2 = np.array([[4, 5], [6, 7]])

# (a)
array3 = np.concatenate((array1, array2), axis=0)
print(array3)
# (b)
array4 = np.concatenate((array1, array2), axis=1)
print(array4)
# (c)
array5 = np.concatenate((array4, array4), axis=0)
print(array5)
# (d)
array6 = np.concatenate((array3, array3), axis=1)
print(array6)
```

```
'''13.  Use NumPy's tile function to create a checkerboard pattern of dashes and asterisks'''

import numpy as np
pattern = np.array([['-','*'], ['*','-']])
checkerboard = np.tile(pattern, (4, 4)) #np.tile function replicates (tiles) the pattern along the rows
and columns.
for row in checkerboard:
    print(" ".join(row))
```

```
'''14)Use the NumPy bincount function to count the number of occurrences of each non-negative integer in
a 5-by-5 array of random integers in the range O - 99'''

import numpy as np
import random
arr = np.random.randint(0, 100, (5, 5))# Generate a 5-by-5 array of random integers in the range 0 - 99
#np.bincount takes an array of non-negative integers as input
counts = np.bincount(arr.flatten())# Use the bincount function to count the number of occurrences of each
non-negative integer
print(counts)
```

```
'''15.Write functions median and mode that use existing NumPy capabilities to determine the median
(middle) and mode (most frequent) of the values in an array. Your functions should determine the median
and mode regardless of the array's shape. Test your function on three arrays of different shapes.'''

import numpy as np
import statistics as st
def median(arr):
    return st.median(arr.flatten())

def mode(arr):
    return st.mode(arr.flatten())

print(median(np.array([1,2,3,4,5]))) #similary for mode
print(mode(np.array([[1, 3, 5], [7, 2, 5]])))#similarly for median
print(median(np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]]))) #similarly for mode
```

```
'''16.Write a NumPy program to create a 9*9*2 array with random values and extract any array of shape
(5,5,2) from the said array.
'''
import numpy as np
array = np.random.randint(0, 10, size=(9, 9, 2))
subarray = array[2:7, 2:7, :]
print("Original 9x9x2 Array:")
print(array)
print("Extracted 5x5x2 Subarray:")
print(subarray)
```

```
'''17.Write a code to create a 4*4 array with random values and sort each column.'''

import numpy as np
ar1 = np.random.randint(1, 11, size=(4, 4))
sorted_array = np.sort(ar1, axis=0) #will sort the coloumn i.e axis=0
print("Sorted Array (each column sorted):")
print(sorted_array)
```

```
'''18.  Write a Pandas program to convert the first column of a DataFrame as a Series.'''

import pandas as pd
data = {'A': [1, 2, 3, 4],'B': [5, 6, 7, 8],'C': [9, 10, 11, 12]}
df = pd.DataFrame(data)
firstCol=df['A']
print(df) #Original
print(firstCol) #First Column Series
```

```
'''19.  Convert sl=[1,2,3,4,2] and s2=[3,4,5,6] to two series objects. Find elements in sl, which are not
present in s2.'''

import pandas as pd
s1=pd.DataFrame([1,2,3,4,2])
s2=pd.DataFrame([3,4,5,6])
print(s1[~s1.isin(s2).any(axis=1)])
```

```
'''Q20) Write a Pandas program to find the index of the first occurrence of the smallest and largest
value of a given series. If the input is [1,1, 3, 7,88, 12, 88, 23, 3, 1, 9, O], the output should be O
and 4'''

import pandas as pd
s = pd.Series([1, 1, 3, 7, 88, 12, 88, 23, 3, 1, 9, 0])
print(s.idxmin()) #min index = 11
print(s.idxmax()) #max index = 4
```

```
'''21.  Convert L=['Cry', 'Apple', 'Orange', 'Sky', 'Banana'] to a pandas series. Create a new series
with the elements which has a vowel. Create another series which starts with a vowel.'''

import pandas as pd
L = ['Cry', 'Apple', 'Orange', 'Sky', 'Banana']
series= pd.Series(L)
vowels= ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
has_vowel= series[series.str.contains(f"[{''.join(vowels)}]")]
start= series[series.str[0].isin(vowels)]

print(series) #Original
print(has_vowel) #at least one vowel
print(start) #starts with vowel
```

```
'''22.  Perform the following tasks using the pandas Series object:'''

import pandas as pd
import numpy as np

print(pd.Series([7, 11, 13, 17]))#(a)Create a Series from the list [ 7, 11, 13, 1 7] .
print(pd.Series([100.0] * 5))#(b)Create a Series with five elements where each element is 10 0 . 0.

print(pd.Series(np.random.randint(0, 101, size=20)))#(c)Create a Series with 20 elements that are all
random numbers in the range Oto 100. Use the describe method to produce the Series' basic descriptive
statistics.

print(pd.Series([98.6, 98.9, 100.2, 97.9], index=['Julie', 'Charlie', 'Sam', 'Andrea']))#(d) Create a
Series called temperatures with the following floating-point values: 98. 6, 98. 9, 1 0 0 . 2, and 9 7 .
9. Use the index keyword argument to specify the custom indices 'Julie', 'Charlie', 'Sam', and 'Andrea'.

temp_dict = {'Julie': 98.6, 'Charlie': 98.9, 'Sam': 100.2, 'Andrea': 97.9}
print(pd.Series(temp_dict))#( e)form a dictionary from the names and values in Part ( d), then use it to
initialize a Series.
```

```
'''23.  Perform the following tasks using the pandas DataFrame object:
(a) Create a DataFrame named temperatures from a dictionary of three temperature readings each for
'Maxine', 'James', and ' Amanda'.'''

import pandas as pd
temperature_data = {
    'Maxine': [22.5, 24.0, 19.5],
    'James': [21.0, 23.5, 20.0],
    'Amanda': [23.0, 25.0, 21.5]}

temperatures = pd.DataFrame(temperature_data)

# (b)Recreate the DataFrame temperatures in Part (a) with custom indices using the index keyword argument
and a list containing 'Morning',' Afternoon', and 'Evening'.
temperatures_custom_index = pd.DataFrame(temperature_data, index=['Morning', 'Afternoon', 'Evening'])

# (c)Select from temperatures the column of temperature readings for 'Maxine'.
maxine_temperatures = temperatures['Maxine']

# (d)Select from temperatures the row of 'Morning' temperature readings.
morning_temperatures = temperatures_custom_index.loc['Morning']

# (e)Select from temperatures the rows for 'Morning' and 'Evening' temperature readings.
morning_evening_temperatures = temperatures_custom_index.loc[['Morning', 'Evening']]

# (f)Select from temperatures the columns of temperature readings for ' Amanda' and 'Maxine'.
amanda_maxine_temperatures = temperatures_custom_index.loc[:, ['Amanda', 'Maxine']]

# (g)Select from temperatures the elements for ' Amanda' and 'Maxine' in the 'Morning' and 'Afternoon'.
morning_afternoon_amanda_maxine = temperatures_custom_index.loc[['Morning', 'Afternoon'], ['Amanda',
'Maxine']]

# (h)Use the describe method to produce temperatures' descriptive statistics.
descriptive_stats = temperatures_custom_index.describe()

# (i)Transpose temperatures.
transposed_temperatures = temperatures_custom_index.T

# (j)Sort temperatures so that its column names are in alphabetical order. '''
sorted_temperatures = temperatures_custom_index[sorted(temperatures_custom_index.columns)]
```