Date : 7/9/25
MON TUE WED THU FRI SAT SUN

Module - 3 (Numpy)

Assignment Questions :-

Q1 What is a Numpy - N-dimensional Array (ndarray)? Explain its importance in numerical data analysis with an example.

Ans :-

- A Numpy ndarray (N-dimensional array) is a powerful data structure that stores homogeneous (same type) data in a multidimensional, continuous memory block.
- It is faster and more memory - efficient than normal python lists.
- Supports vectorized operations (performing computations on entire arrays without loops.)

★ Importance in data Analysis :-

- Provides high-speed computations.
- Used in mathematical modeling, machine learning, and scientific computing
- Handles large datasets efficiently.

Date : __/__/____
MON TUE WED THU FRI SAT SUN

☆ Example :

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

#output :
# [[1 2 3]
# [4 5 6]]
```

Here "arr" is a 2D ndarray (2 Rows x 3 columns).

Q2 What is indexing in Numpy arrays? How does it differ from normal Python list indexing? Give a suitable example.

Answer

- Indexing means accessing specific elements of an array using their position.
- In Numpy arrays, indexing can be done with integers, slices, boolean masks, or arrays.

☆ Difference from Python list indexing :

- Python lists allow only simple indexing
- Numpy allows advanced indexing (multi-dimensional, boolean, fancy indexing).

Date : __/__/____
MON TUE WED THU FRI SAT SUN

* Example:

```
import numpy as np
arr = np.array([10, 20, 30, 40, 50])

#python list-styling indexing.
Print(arr[2])    #output : 30

#numpy advance indexing.
print(arr[[1,3]])  #output : [20 40]
```

Q3. Define slicing in Numpy. How can slicing be used to access a range of elements in an array? Explain with an example.

Answer

- Slicing means accessing a subset of elements from an array using "start:end:step".
- Works similar to Python lists but supports multi-dimensional slicing.

→ General form : "array[start:end:step]"

* Example :

```
import numpy as np
arr = np.array([10, 20, 30, 40, 50, 60])

print(arr[1:5])   #output : [20 30 40 50]
print(arr[::2])   #output : [10 30 50]
```

Date : __/__/____
MON TUE WED THU FRI SAT SUN

★ For 2D Array:

```
mat = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(mat[0:2, 1:3])  #Output: [[2 3] [5 6]]
```

Q4 What do you mean by reshaping in Numpy arrays? State the rules of reshaping and give a practical use case.

Answer

- Reshaping means changing the shape (rows × columns × Dimensions) of an array without changing its data.

★ Rules of Shaping:

① Total number of elements must remain ~~cont~~ constant.
- Example: "(2×6)" → "(3×4)" allowed (12 elements).

② New shape must be compatible with existing data.

③ Can use -1 to let Numpy auto-calculate dimension.

★ Example:

```
import numpy as np
~~arr = np.array~~
```

Date : __/__/____
MON TUE WED THU FRI SAT SUN

```
arr = np.arange(12)  #[0, 1, 2. . . . , 11]

reshaped = arr.reshape(3,4)
print(reshaped)

#output:
#[[0  1  2  3]
#  [4  5  6  7]
#  [8  9  10  11]]
```

☆ Practical use case: converting a flat dataset into a 2D matrix for machine learning models.

Q5. What is ND-array arithmetic in Numpy? Explain element-wise operations (addition, substraction, multiplication, division) with examples.

Answer:

- ND-Array arithmetic means performing mathematical operations directly on arrays.
- Operations are applied element-wise, i.e., same position elements are operated together.

☆ Example:

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
```

Date : __/__/____
MON TUE WED THU FRI SAT SUN

```
print (a+b)   # Addition → [5  7  9]
print (a-b)   # subtraction → [-3  -3  -3]
print (a*b)   # Multiplication → [4  10  18]
print (b/a)   # Division → [4.  2.5  2.]
```

★ Importance :
- Removes need for loops.
- Makes numerical computing much faster and concise.

★★ Programming Questions :→

Q1 Consider a situation, where a programmer is asked to take as input the ages of 5 students from the user using a loop and save it to a list. Now, at the time of printing the list of ages, "# Index error : list index out of range." The above error has shown. Deduce the possible reason for the error and write the corrected code to carry out the job.

Ans → Reason : Error occurs when we try to access indices beyond the valid length of the list (e.g., wrong loop range or using index more than "len(list) - 1").

Correction : Use "len()" while looping to avoid going out of range.

Date : __/__/____
MON TUE WED THU FRI SAT SUN

correct code:

```
ages = []
for i in range (5):
    age = int (input ("Enter age of student: "))
    ages. append (age)

# printing correctly using len ().

for i in range (len(ages)):
    print ("student", i + 1, "Age:", ages[i])
```

Q2. Consider a program with a recursive function to calculate the factorial of a given number and save the values of the recursive call to an array using the Numly library. Measure the number of recursive function calls for $n = 5$.

Ans
- Concept → Factorial uses recursion. ($n! = n \times (n-1)!$)
- Each recursive call stores the current value in an array.
- For $n = 5$, recursion tree has 5 calls ($5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$).

Code:

```
import numpy as np

calls = []
```

Date : __/__/____
MON TUE WED THU FRI SAT SUN

```
def factorial (n) :
    calls.append (n)    # save recursive call
    if n == 1 :
        return 1
    else :
        return n * factorial (n-1)

n=5
result = factorial (n)

calls_array = np.array (calls)
print ("factorial of", n, "=", result)
print ("Recursive calls Array:", calls_array)
print ("Number of recursive calls :",
        len (calls_array))
```

Output :

Factorial of 5 = 120
Recursive calls Array = [5 4 3 2 1]
Number of recursive calls = 5

Q3 Say, you are given a list of numbers ~~to~~ 10.45, 20.66, 30.85, 40.55 and calculate the Hash values and store them in list given below

ls = [value_1, hash_value (value1), value_2, hash_value (value_2), value_3, hash_value (value_3), ---- value_5, hash_value (value_5)]

write a code in python to carry out this task and show middle element

Date : __/__/____
MON TUE WED THU FRI SAT SUN

value with its corresponding hash value.

Answer

- Concept : "hash()" → generates unique integers.
- Store [value, hash(value)] sequentially in list.
- Middle element = 3rd value (30.85) and its hash.

Code:

```
numbers = [10.45, 20.66, 30.85, 40.55, 50.75]
ls = []

for val in numbers:
    ls.extend([val, hash(val)])

print("list with values and hash:", ls)

# Middle term / element
mid_index = len(ls)//2
print("Middle value & Hash:", ls[mid_index-1],
      ls[Mid_index])
```

Q4. Suppose, you are given two arrays of 5 by 5 each and asked to perform matrix multiplication taking input from two separate files like Matrix_A.txt and Matrix_B.txt. But you got the error "FilenotFoundError"

Date : __/__/____
MON TUE WED THU FRI SAT SUN

Deduce the possible reason for the error and write the corrected code to carry out the Job.

Answer

- Reason: Error occurs if file does not exist in working dictionary or wrong path is given.
- Correction: Place the file in same folder as program or use full file path.

Correct Code:

```
import numpy as np

# Ensure files exist in same dictionary

A = np.loadtxt("Matrix_A.txt", dtype = int)
B = np.loadtxt("Matrix_B.txt", dtype = int)

result = np.dot(A, B)
print("Matrix Multiplication Result: \n"
      , result)
```

Q5. Consider a program containing two arrays first_array = np.array([13, 19, 25, 10]), second_array = np.array([12, 14, 18, 16]), Try to perform addition and subtraction elementwise. Measure the number of addition and subtraction operations performed.

Date : __/__/____
MON TUE WED THU FRI SAT SUN

Answer :-

- Perform elementwise Operation
- Count operations with counter variables.

Code:

```
import numpy as np

first_array = np.array([13, 19, 25, 10])
second_array = np.array([12, 14, 18, 16])

add_count = 0
subtraction = 0
sub_count = 0

addition = []
subtraction = []

for i in range(len(first_array)):
    addition.append(first_array[i]
    + second_array[i])
    add_count += 1

    subtraction.append(first_array[i]
    - second_array[i])
    sub_count += 1

print("Addition Result:", addition)
print("subtraction Result:", subtraction)
print("Total Addition Ops:", add_count)
print("Total Subtraction Ops:", sub_count)
```

Output :

Addition Result : [25, 33, 43, 26]
Subtraction Result : [1, 5, 7, -6]
Total Addition Ops : 4
Total Subtraction Ops : 4

Q6 :- Create a program to compose a sequential array with a negative step.

Answer → Use np.arange (start, stop, step) with negative step.

Code:

```
import numpy as np
arr = np.arange (20, 0, -3)   # start = 20,
                              # stop = 0,
                              # step = -3
print ("Sequential Array with negative Step:", arr)
```

Output : [20 17 14 11 8 5 2]