எங்கள் வாழ்வும் எங்கள் வளமும்
மங்காத தமிழ் என்று சங்கே முழங்கு ... *புரட்சிக்கவி*

NOTICE

www.DataScienceInTamil.com

Day 36 - Batch 3 - Python Language

Chapter 018 Recursion, List of List, matrices and cubes

To watch the recorded Python and Data Science videos in YouTube:

Day 36- Batch 3 - Recursion, List of List, matrices and cubes

https://youtu.be/WHJ1B3sUSAw

Official Website:

https://DataScienceInTamil.com/

மேலும் முக்கிய கேள்விகள் பதில்களுக்கு :

https://www.DatascienceInTamil.com/#faq

To join DataScienceInTamil Telegram group:

இந்த குழுவில் உங்கள் நண்பர்களை இணைக்க விரும்பினால் அதற்கான லிங்க்

https://t.me/joinchat/lUZEsr-zidpjZjEx

To Join the class, please fill the form :

https://forms.gle/QFpLHwAoinFaX2cE6

Join Zoom Meeting (From Sep 26 2022 to Oct 26 2022)

https://us06web.zoom.us/j/88900302653?pwd=MVBFUlhqTTE1LzFFRUVpTzZ2S1Vsdz09

Meeting ID: 889 0030 2653

Passcode: 1234

Monday through Friday 8 PM to 10 PM IST (From Sep 26 2022 to Oct 26 2022)

We support open-source products to spread Technology to the mass.

- This is completely a FREE training course to provide introduction to Python language
- All materials / contents / images/ examples and logo used in this document are owned by the respective companies / websites. We use those contents for FREE teaching purposes only.
- We take utmost care to provide credits whenever we use materials from external source/s. If we missed to acknowledge any content that we had used here, please feel free to inform us at info@DataScienceInTamil.com.

- All the programming examples in this document are for FREE teaching purposes only.

Thanks to all the open-source community and to the below websites from where we take references / content /code example, definitions, etc., please use these websites for further reading:

Team, PCEP exam point of view the following topics are yet to be covered

Vanitha is going to cover the below

---

1. scientific notation
2. the accuracy of floating-point numbers

3. type casting
4. the print() and input() functions
5. the sep= and end= keyword parameters
6. the int() and float() functions

Melcose is going to cover the below

---

1. constructing vectors (marks = [4,5,6])
2. list comprehensions//DONE
3. copying and cloning// DONE
4. escaping using the \ character// DONE
6. Functions // DONE
7. Exceptions// DONE
8. defining and invoking user-defined functions //DONE

   8a) generators // DONE
9. the return keyword, returning results// DONE

10. the None keyword// DONE
11. recursion
12. parameters vs. arguments// DONE
13. positional, keyword, and mixed argument passing// DONE
14. name scopes, name hiding (shadowing), and the global keyword // DONE
15. Generators // DONE
16. lists in lists: matrices (list of list) and cubes

https://numpy.org/
https://www.tutorialspoint.com/numpy/index.htm
https://www.w3schools.com/python/python_datatypes.asp
Python Notes For Professionals.pdf – this is the book we follow
https://docs.python.org/3/tutorial/
https://docs.python.org/3.9/tutorial/index.html
https://blog.finxter.com/what-are-advantages-of-numpy-over-regular-python-lists/
https://towardsdatascience.com/lets-talk-about-numpy-for-datascience-beginners-b8088722309f
https://data-flair.training/blogs/numpy-applications/
https://data-flair.training/blogs/numpy-features/amp/
https://www.mathsisfun.com/algebra/scalar-vector-matrix.html

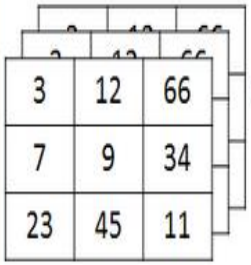
Scalar	Vector	Matrix	Tensor
1	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 7 \end{bmatrix} & \begin{bmatrix} 3 & 2 \\ 5 & 4 \end{bmatrix} \end{bmatrix}$

1	5	18	23

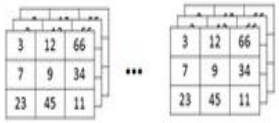
Vector (1D array)
Dimension = 1
(1 index required)

3	12	66
7	9	34
23	45	11

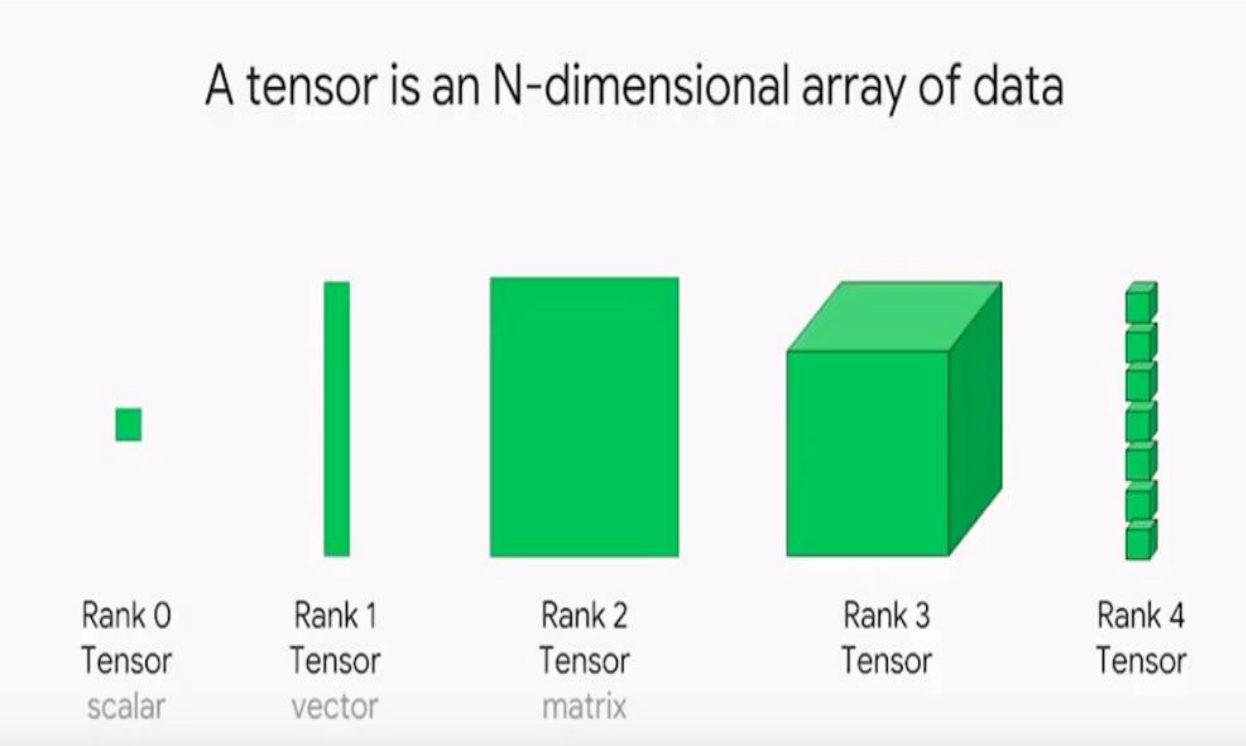
Matrix (2D array)
Dimension = 2
(2 indexes required)

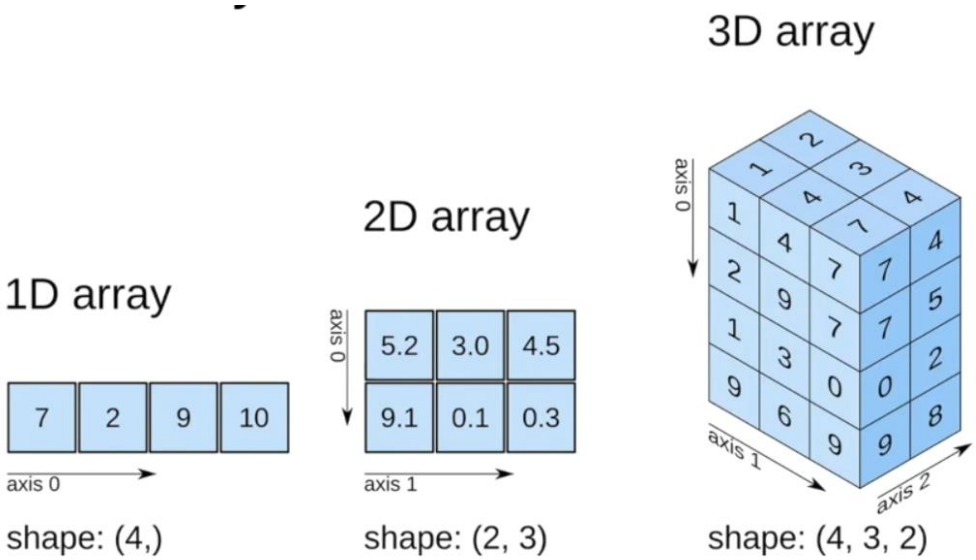


3D array (3rd order Tensor)
Dimension = 3
(3 indexes required)

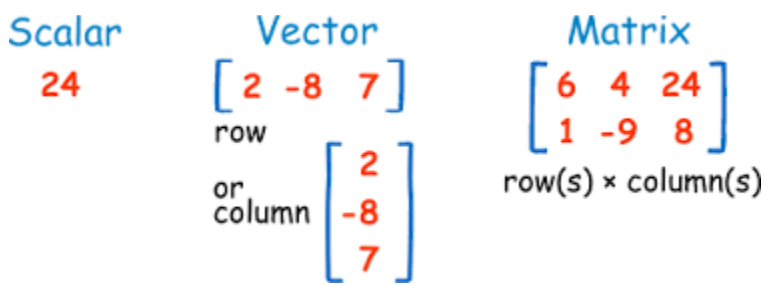


ND array
Dimension = N
(N indexes required)


A tensor is an N-dimensional array of data
Rank 0
Tensor
scalar
Rank 1
Tensor
vector
Rank 2
Tensor
matrix
Rank 3
Tensor
Rank 4
Tensor



constructing vectors – PCEP



A vector is a list of numbers (can be in a row or column), A matrix is an array of numbers (one or more rows, one or more columns).

==========

creates a rows and cols that is a 2D array, that is a matrices

```
# a = [[  [10, 80], [100, 800],[1,8]   ]]
a = [ [10, 80], [100, 800],[1,8] ]
print(type(a))
print(len(a))
for item in a:
    print(item)

print(a[0])
```

Output // it creates a rows and cols that is a 2D array, that is a matrice

```
<class 'list'>
3
[10, 80]
[100, 800]
[1, 8]
[10, 80]
```

==========

Create a cube / 3D list – using NumPy

```python
import numpy as np
a = [[ [10, 80, 800], [11, 81, 810],[12,82, 811] ], [ [10, 80, 800], [11, 81, 810],[12,82, 811] ] ]
print(a)
print("=" * 30)

print("Dim", np.ndim(a))
print("Shape" , np.shape(a))
print("=" * 30)

print("Axis 0")
sumAnswer=(np.sum(a,axis=0))
print(sumAnswer)
print("Dim", np.ndim(sumAnswer))
print("Shape", np.shape(sumAnswer))
```

output

[[[10, 80, 800], [11, 81, 810], [12, 82, 811]], [[10, 80, 800], [11, 81, 810], [12, 82, 811]]]

==========

Dim 3

Shape (2, 3, 3)

==========

Axis 0

```
[[  20   160 1600]
 [  22   162 1620]
 [  24   164 1622]]
```

Dim 2

Shape (3, 3)

Create a cube / 3D list – using list

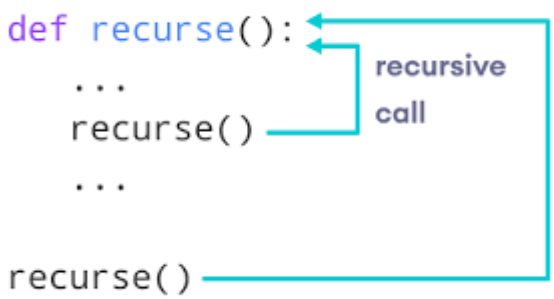
```
a = [[ [10, 80, 800], [11, 81, 810],[12,82, 811] ], [ [10, 80, 800], [11, 81, 810],[12,82, 811] ] ]

print(type(a))
print(len(a))

for item in a:
    print(item)

print(a[0])
```

output

```
<class 'list'>
2
[[10, 80, 800], [11, 81, 810], [12, 82, 811]]
[[10, 80, 800], [11, 81, 810], [12, 82, 811]]
[[10, 80, 800], [11, 81, 810], [12, 82, 811]]
```

Recursion

Python also accepts function recursion, which means a **defined function can call itself**. Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.



```python
def callMe():
    print("This fn calls itself")
    callMe()

callMe()
```

output

This fn calls itself

This fn calls itself

Traceback (most recent call last):

  File "C:\Users\Melcose\PycharmProjects\pythonProject\1_Machine_Learning_Melcose.py", line 5, in <module>

    callMe()

  File "C:\Users\Melcose\PycharmProjects\pythonProject\1_Machine_Learning_Melcose.py", line 3, in callMe

    callMe()

```
  File "C:\Users\Melcose\PycharmProjects\pythonProject\1_Machine_Learning_Melcose.py", line 3, in callMe
    callMe()
  File "C:\Users\Melcose\PycharmProjects\pythonProject\1_Machine_Learning_Melcose.py", line 3, in callMe
    callMe()
  [Previous line repeated 993 more times]
  File "C:\Users\Melcose\PycharmProjects\pythonProject\1_Machine_Learning_Melcose.py", line 2, in callMe
    print("This fn calls itself")
RecursionError: maximum recursion depth exceeded while calling a Python object
=======
```

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

1. Every recursive function must have a base condition that stops the recursion or else the function calls itself infinitely.
2. The Python interpreter limits the depths of recursion to help avoid infinite recursions, resulting in stack overflows.
3. By default, the maximum depth of recursion is **1000**. If the limit is crossed, it results in **RecursionError**

ex

```
def recursor():
    recursor()
recursor()
```

output

[Previous line repeated 996 more times]

RecursionError: maximum recursion depth exceeded

Advantages of Recursion

1. Recursive functions make the code look clean and elegant.
2. A complex task can be broken down into simpler sub-problems using recursion.
3. Sequence generation is easier with recursion than using some nested iteration.

Disadvantages of Recursion

1. Sometimes the logic behind recursion is hard to follow through.
2. Recursive calls are expensive (inefficient) as they take up a lot of memory and time.
3. Recursive functions are hard to debug.

==========

```
def count_down(start):
    """ Count down from a number  """
    print(start)
    count_down(start-1)
count_down(20)
```

output

run the code to see the output

==========

Points to note

- A recursive function is a function that calls itself until it doesn't.
- And a recursive function always has a condition that stops calling itself.

==========

Batch 3 Python course (for the purpose of writing PCEP) has been completed

Thanks to all our team mates

I wish every one of us would prepare, write and pass the PCEP exam

www.DataScienceInTamil.com

A.Melcose
Melcose@Gmail.com

எங்கள் வாழ்வும் எங்கள் வளமும்
மங்காத தமிழ் என்று சங்கே முழங்கு ... *புரட்சிக்கவி*