NumPy

NUMERICAL PYTHON

این کتابخانه ابزاریست بسیار سریع برای محاسبات علمی که به ما در محاسبه عملیات ریاضی و منطقی(True,False) کمک میکند

با استفاده از کتابخانه NumPy در کنار Mathplotlib(برای مصور سازی داده ها) و pandas(برای کار با داده ها)
در کنار هم میتوان بسیاری از مسائل فیزیکی و کار های آزمایشگاهی را حتی در مقیاس بسیار بزرگ
شبیه سازی و آزمایش کرد
مسائلی مانند تبدیلات فوریه و جبر خطی و ماتریس ها و...

و به همین دلیل و سرعت بالا و هزینه های کمتر و راحتی در برنامه نویسی این کتابخانه در حوضه data science کاربرد وسیع و نیاز به مختصص دارد

NumPy از آرایه (لیست) های چند بعدی برای پیشبرد عملیات خود استفاده می‌کند

لیست ها در NumPy به عنوان آرایه ها (array) شناخته می‌شود

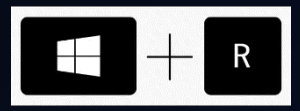
تفاوت هایی که لیست ها و آرایه ها دارند باعث شده تا آرایه ها تقریبا پنجاه برابر سریع تر از لیست ها عمل کنند

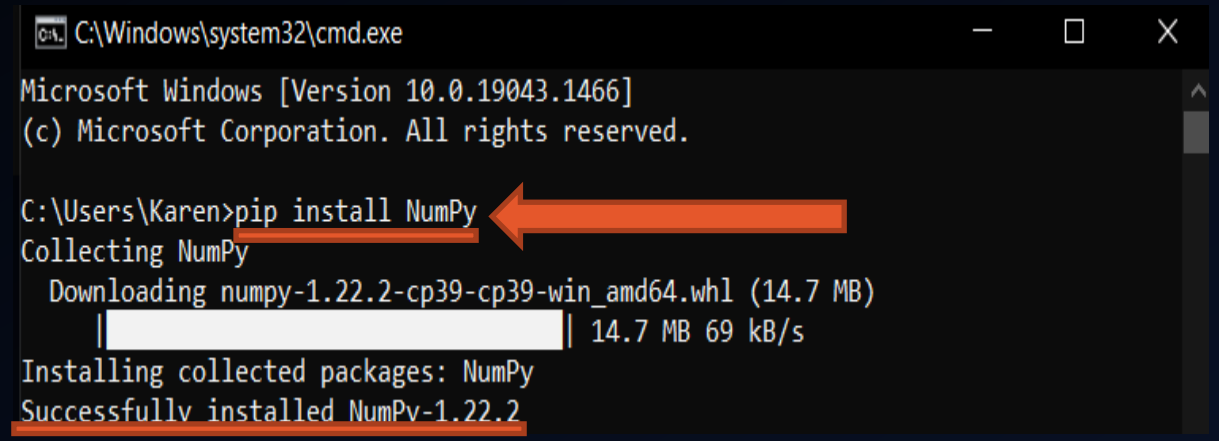
یاد آوری

در ابتدای کارگاه نحوه نصب این کتابخانه مورد بحث قرار گرفت

در منوی جست و جوی ویندوز تایپ کنید CMD و برنامه را اجرا کنید

یا با استفاده از کلید های ویندوز و R پنجره RUN را باز کرده و کلمه CMD را تایپ کنید
و با زدن کلید ok برنامه را باز کنید





در حالت کلی برای نصب هر کتابخانه ای فرمول زیر صادق است:

Pip install نام کتابخانه

به یاد داشته باشید که کتابخانه ها و ماژول ها باید در برنامه فراخوانی شوند

پس با دستور import numpy یا import numpy as np این کتابخانه را در برنامه خود فراخوانی می‌کنیم

حال توانایی ساختن آرایه های numpy را داریم تا بعد ها از آن در برنامه خود استفاده کنیم
این کار توسط تابع ()array صورت میگیرد

```
My_array = np.array([1, 2, 3, 4, 5])
```

تفاوتی در نوع آرگومان این تابع وجود ندارد این تابع چه لیست چه تاپل را به یک ndarray تبدیل می‌کند

با تابع ()type امتحان کنید

همانند لیست ها آرایه ها نیز از آرایه های چند بعدی پشتیبانی میکنند
آرایه صفر بعدی یا اسکالر

```
np.array(12)
```

آرایه یک بعدی یا بردار

```
np.array([12,25])
```

آرایه دو بعدی یا ماتریس

```
np.array([ [12,25] , [10,0] ])
```

آرایه سه بعدی یا تانسور مرتبه سه

```
np.array([  [ [1,3],[5,4] ] ,  [ [4,1],[6,5] ]  ])
```

و ...

رایج ترین نوع آرایه ها آرایه های یک بعدی و گاها دو بعدی است

لازم به ذکر است که کتابخانه Numpy دارای یک زیر ماژول به اسم numpy.mat می‌باشد که مختص عملیات ماتریس ها است
با دستور زیر هم میتوان ابعاد یک آرایه را به دست آورد

```
Array_name.ndim
```

از آنجایی که آرایه ها منظم هستند پس با ایندکس دهی مانند لیست ها میتوان به عناصر داخل آن دسترسی داشت

آرایه های یک بعدی

Array_name[index]

آرایه های دو بعدی

Array_name[row][column] یا array_name[row,column]

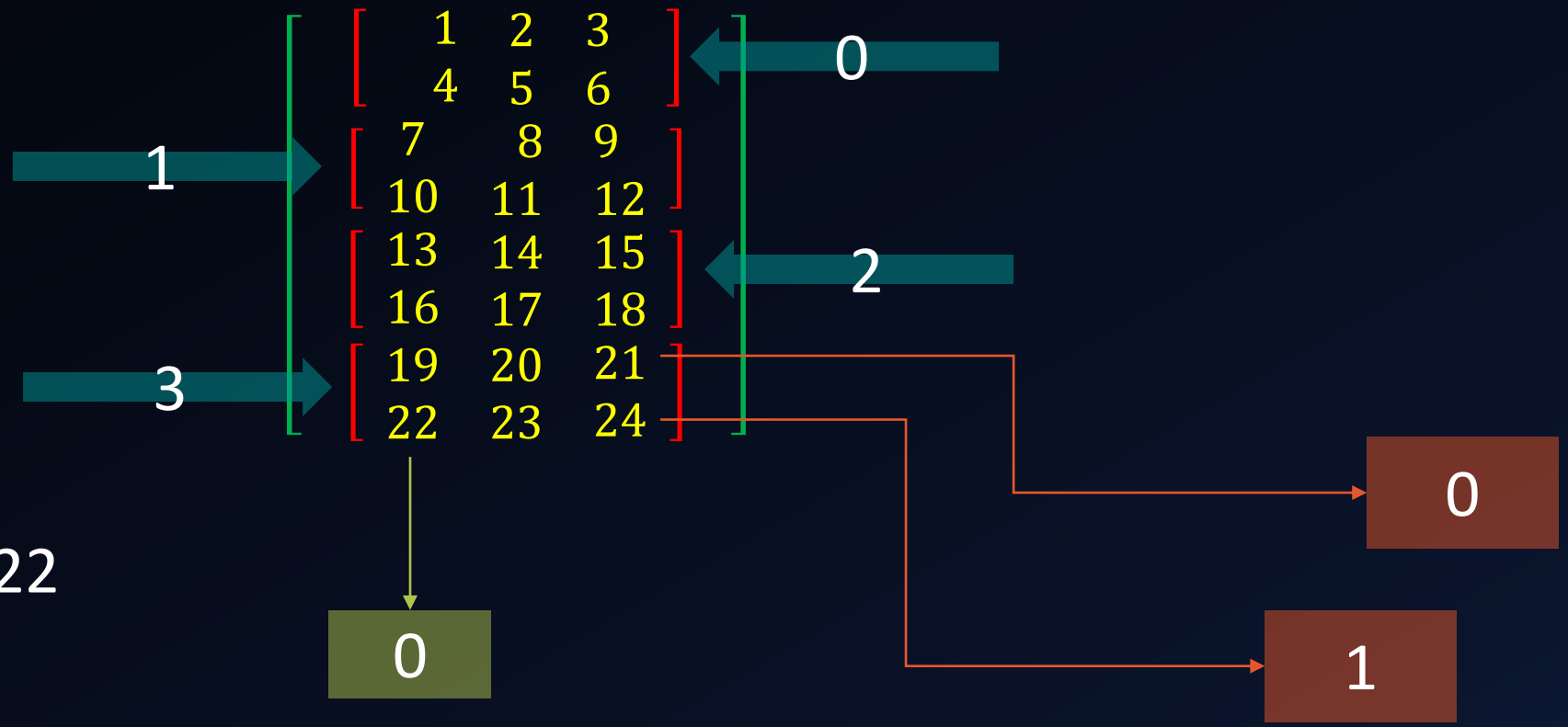
آرایه های سه بعدی

Array_name[row][column][index] یا array_name[row,column,index]

و به همین ترتیب ابعاد بیشتر ...

فرض کنید یک آرایه مرتبه سه به شرح زیر داریم

```
Arr = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]],[[13,14,15],[16,17,18]],[[19,20,21],[22,23,24]]])
```



Arr[3,1,0] ⟹ = 22

برش دادن آرایه های یک بعدی دقیقا مانند برش دادن لیست ها می‌باشد که قبلا بحث شد

My_list [x : Y : z]

ابتدای برش

انتهای برش

قدم های برش

اما برش دادن آرایه های دو بعدی کمی متفاوت است از این جهت که باید سطر مورد نظر را هم مشخص کرد

My_array[k : m : n , x : y : z]

برش دلخواه از سطر های انتخاب شده

تعیین سطر های دلخواه

مثال:

بر روی آرایه زیر به عنوان مثال تمرین کنید

```
arr = np.array([[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]])
```

شکل یک آرایه (shape)

از آنجا که اغلب در کار با ماتریس ها ابعاد آنها برای ما مهم هستند بنا بر این متدی به همین نام برای مشخص کردن ابعاد ماتریس ها وجود دارد

Array_name.shape

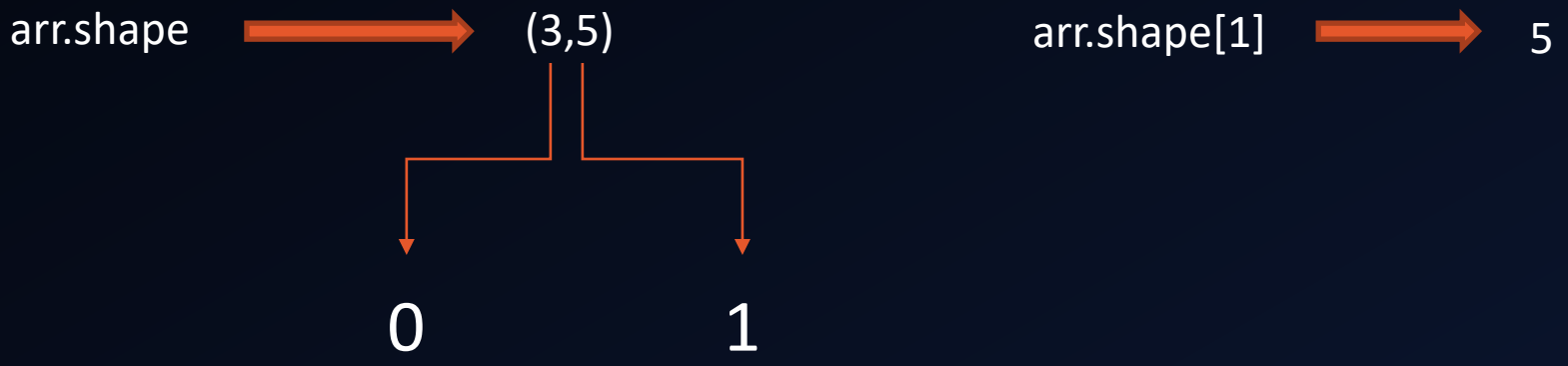
این دستور ابعاد ماتریس را در قالب یک تاپل بر میگرداند (به تبع قوانین تاپل ها اینجا نیز حاکم است)

برای مثال اگر یک ماتریس ۳ در ۵ داشته باشیم پاسخ دریافتی ما (3,5) خواهد بود

سوال
نتیجه کد زیر را پیشبینی کنید:

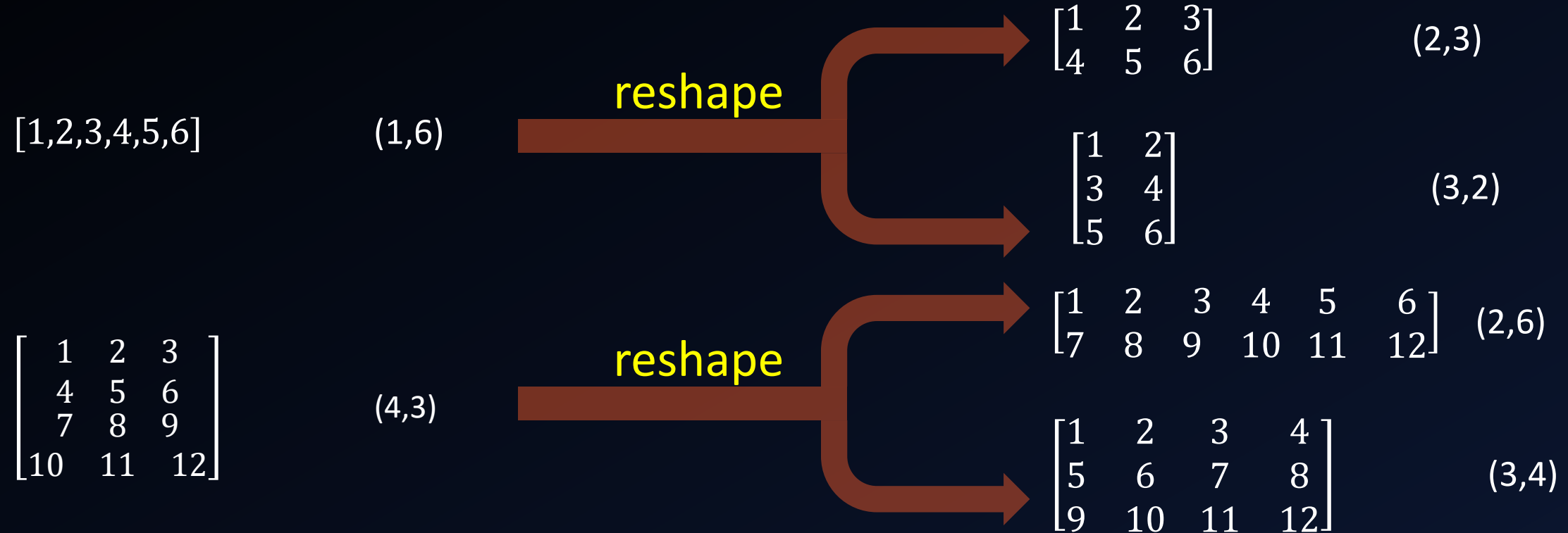
```
arr = np.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]])
print(arr.shape[1])
```

پاسخ:



تغییر شکل دادن آرایه ها (reshape)

با استفاده از تغییر شکل یک آرایه را می‌توان به آرایه ای با ابعاد مختلف تغییر داد



بدیهی است تغییراتی قابل قبول می‌باشد که ضرب سطر و ستون ها برابر با تعداد درایه های ماتریس باشد
برای مثال تغییر شکل دادن یک ماتریس (4,3) (مثال دوم) به یک ماتریس (3,5) غیر قابل قبول است

```
arr=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
arr=arr.reshape(3,4)
```

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

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

```
arr=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
arr=arr.reshape(2,3,2)
```

```
[[[ 1,  2],
  [ 3,  4],
  [ 5,  6]],

 [[ 7,  8],
  [ 9, 10],
  [11, 12]]]
```

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} \end{bmatrix}$$

گاهی نیاز می‌شود تا آرایه را به یک آرایه (1,n) تبدیل کنیم برای این کار آرایه (2,3,2) را در نظر بگیرید :

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} \end{bmatrix}$$

```
arr=arr.reshape(-1)
```

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

حال میتوانیم در این آرایه flattened (مسطح شده) عملیات مورد نظر را انجام داده
پس از اتمام کار آن را به حالت اول reshape کنیم

array_split

با این متد میتوان آرایه مورد نظر را به تعداد دلخواه تقسیم کرد

```
arr=np.array([1,2,3,4,5,6,7,8])
array_split(arr , 3)
```

```
[1 2 3] , [4 5 6] , [7 8]
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 \end{bmatrix}$$

```
arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])
newarr = np.array_split(arr, 3)
```

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\ \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ \begin{bmatrix} 9 & 10 \\ 11 & 12 \end{bmatrix} \end{bmatrix}$$

میتوان در صورت نیاز یک شرط را در یک آرایه جست و جو کرد و نتیجه را در یک تاپل دریافت کرد

```
np.where(condotion)
```

مثال:

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

در این مثال index های اعدادی از این آرایه که شرط را ارضا کنند برگردانده میشوند
یعنی اعداد زوج

خروجی این مثال به این صورت است

```
(array([1, 3, 5, 6], dtype=int64),)
```

یا این مثال که اعداد بزرگ تر یا مساوی ۵ را بر میگرداند

```
arr = np.array([1,4,5,8,9,6,2,5,7])

np.where(arr >= 5 )
```

```
(array([2, 3, 4, 5, 7, 8], dtype=int64),)
```