جزوه شماره ۷

NUMPY LIBRARY

ArcGIS Pro

محمدطاهر طاهرپور

دانشجوی ارشد مدیریت شهری

دانشگاه تهران

mttaherpoor
@mtaherpoor
09336144947

ناهید نعمتی کوتنائی (تیسا)

دکتری جغرافیا و برنامه‌ریزی شهری

مدرس دانشگاه

Dr.nemati.K
@Nemati_k
09112230798

تابستان ۱۴۰۲

فهرست مطالب:

Numpy Library ۳
نامپی Numpy ۳
چرا باید از نامپی Numpy استفاده کنیم؟ ۴
چطوری از نامپی Numpy استفاده کنیم؟ ۴
متدهای مهم آرایه و نامپی ۵
ویژگی ndim ۶
ویژگی shape ۶
ویژگی size ۶
حلقه for در آرایه ۶
ویژگی flat ۶
توابع zeros, ones, full ۶
تابع arange ۷
تابع linspace ۷
reshape ۸
np.random.randint() ۸
عملگرهای آرایه ۸
نمایه سازی و برش indexing and slicing ۹
توابع نامپی ۱۲
تابع np.argpartition() ۱۲
تابع np.allclose() ۱۶
تابع np.clip() ۱۷
تابع np.where() ۱۷
تابع np.pad() ۱۷
تابع np.put() ۱۸
خلاصه دستورهای numpy ۱۹
حل چند تا تمرین ۲۳
جواب تمرینها ۲۴

Numpy Library

تو این جزوه آموزش گام به گام کتابخونه numpy رو با هم یاد میگیریم و انتهای جزوه هم یه تعداد تمرین برای درک بهترش با هم کار میکنیم. مطالب این جزوه از روی مطالب آموزشی Udemy - Data Visualization with Python Masterclass - Python A-Z بخش مربوط به Fundamentals of Data Science و همینطور سایت نامپی https://numpy.org نوشته شده.

برای دریافت اطلاعات بیشتر هم میتونید به این دو تا لینک مراجعه کنید:

https://numpy.org/devdocs/user/

https://github.com/numpy

لینک دوم مربوط به پلتفرم محبوب و معروف گیت هاب هست که سرویس‌های Cloud-based داره. به دولوپرها این امکان رو میده که کدهای Open sourceخودشون رو اینجا با بقیه به اشتراک بگذارن و اشکالات کدهاشون رو رفع کنن و پروژه‌های نرم‌افزاریشون رو توسعه بدن. در واقع یه مرجع مهم برای یادگیری زبانهای برنامه نویسی و همینطور کنترل کار تیمی هست که بعدا حسابی در موردش با هم صحبت میکنیم.

نامپی Numpy

انتهای جزوه جلسه قبل (جزوه شماره ۶) در مورد کتابخونه‌های پایتون با هم صحبت کردیم و یه دسته‌بندی تجربی از کتابخونه‌ها و کاربردهاشون براتون آوردیم.

- برای کار با دیتا یا data در پایتون:
  - ✓ numpy as np
  - ✓ pandas as pd
  - ✓ matplotlib.pyplot as plt
- برای کار با دیتا جغرافیایی در پایتون:
  - ✓ geopandas as gpd
  - ✓ geomatplotlib
  - ✓ arcpy
- برای کار با داده‌های رستری
  - ✓ rasterio
- برای چک کردن داده های موجود در یک جدول
  - ✓ missingno as msno
- برای زیبایی بیشتر نمودارها
  - ✓ seaborn as sns
  - ✓ squarify
- برای یادگیری ماشین
  - ✓ scikit_learn

یکی از مهمترین کتابخونه‌های پایتون نامپی هست که دونستنش واسه علم داده، حیاتی هست.

در واقع نامپی:

- یه کتابخونه ریاضی هست و یه سری آرایه یا Arrays داره که میتونن چندبعدی باشن. مهمترین این آرایه‌ها، N-dimensional array هست.
- علاوه بر اون، نامپی، ابزارهای پایه واسه محاسبات علمی داره.
- شامل توابع پیچیده یا sophisticated function هست.
- ابزارهایی داره که زبانهای برنامه‌نویسی C/C++ و Fortran رو باهم ترکیب میکنه.
- شامل linear algebra یا جبر خطی، Fourier transform (یه تبدیگر هست که یک تابع رو به شکلی تبدیل میکنه که بتونه توالی‌های موجود در تابع اصلی رو تعریف کنه) و random number یا عدد تصادفی میشه.
- علاوه بر اون میتونه به عنوان نگهدارنده چندبعدی هم رفتار کنه.
- همچنین میتونه به سرعت با بقیه پایگاه‌های داده هم ادغام بشه.

اگه از جزوه جلسه ۶ یادت مونده باشه، گفتیم که ما انواع داده آرایه دیگه هم تو پایتون داریم مثل لیستها، ولی نامپی مزایای خیلی بیشتری نسبت به لیستها داره مثل:

- آرایه‌های چندبعدی همنوع
- سرعت بالاتر نسبت به لیست که اینجا سرعت اجرا خیلی اهمیت داره
- داخل آرایه میشه لیستها و حلقه‌های تودر تو هم ایجاد کرد.

نامپی یه برنامه آرایه محور هست. آرایه رو دستکاری میکنه و باگهایی که تو طول اجرای برنامه پیش می‌آد رو پاک میکنه.

چرا باید از نامپی Numpy استفاده کنیم؟

- نامپی یه کتابخونه پایتون منبع باز یا open source هست.
- کلی آرایه چند وجهی و ماتریس ساختار داده داره.
- باهاش میشه عملیات ریاضی انجام داد. در واقع یه اکستنشن از Numeric و Numarray هست و کلی تولیدکننده عدد داره.
- آرایه محور هست.
- از نامپی واسه Big Data و Machine learning استفاده میشه.
- میشه ازش واسه محاسبات فشرده داده‌ها به صورت آسون و سریع استفاده کرد.

پایتون محبوبترین زبان برای محاسبات آماری هست ولی پایتون OOP یا Object oriented Programing یعنی برنامه نویسی شی‌گرا هست و *در واقع پایتون به خاطر Data Science متولد نشده*. ولی نامپی که یه کتابخونه پایتون هست، قدرت محاسبات آماری پایتون رو از ۲۰ درصد به ۱۰۰ درصد افزایش میده.

چطوری از نامپی Numpy استفاده کنیم؟

به شکلی که تو جزوه شماره ۶ توضیح دادیم، وارد ArcGIS pro شو و نوت بوکش رو باز کن. باید نامپی رو شبیه شیوه‌ای که ماژولها رو وارد میکنیم با دستور import وارد کنیم.

```
In [1]: import numpy as np
```

نامپی با تگ np وارد میشه که نوشتن کد رو راحت‌تر کنه.

حالا یه آرایه چندوجهی به اسم N_Array تولید می‌کنیم که از **np.array()** برای ساخت اعضاش استفاده میشه.

```
In [1]: import numpy as np

In [2]: N_Array =np.array([[1,2,3],[4,5,6]])

In [3]: type(N_Array)
Out[3]: <class 'numpy.ndarray'>

In [4]: N_Array
Out[4]: array([[1, 2, 3],
               [4, 5, 6]])
```

از تابع **type()** واسه تشخیص نوع آرایه استفاده میشه و اگه اسم آرایه رو جدا تو یه سلول جدید تایپ کنیم کل آرایه رو با داده‌هاش بهمون خروجی میده. به خروجی که داده دقت کن؛ عدد اول خط اول و دوم به براکت چسبیده و بعد از کاما یه فاصله گذاشته و بعد عدد جدید رو به همراه کاما نوشته. اگه موقع تعریف آرایه، بین اعضایی که تعریف می‌کنی، فاصله بذاری، پایتون اونها رو نادیده میگیره.

حالا با یه آرایه دیگه همینکار رو تکرار می‌کنیم. اینبار به نتیجه نگاه کن نتیجه یه کمی متفاوت میشه. عدد اول خط اول و دوم از براکت فاصله میگیرن. چون موقعیت دوم رو پیدا کرده. در واقع هر عدد پشتش فاصله هست به جز آرایه اول یا هر عدد دو تا کرکتر رو اشغال میکنه.

```
In [5]: N_Array_1 = np.array([[2,4,6],[9,8,15]])

In [6]: type(N_Array_1)
Out[6]: <class 'numpy.ndarray'>

In [7]: N_Array_1
Out[7]: array([[ 2,  4,  6],
               [ 9,  8, 15]])
```

متدهای مهم آرایه و نامپی

اول نامپی رو به عنوان np وادر میکنیم. دو تا مجموعه داده به اسم Integers با اعداد صحیح و Floats با اعداد اعشاری درست میکنیم و ازشون خروجی میگیریم.

```
In [19]: import numpy as np

In [20]: Integers = np.array([[1,2,3, 4],[5,6,7, 13]])

In [21]: Integers
Out[21]: array([[ 1,  2,  3,  4],
                [ 5,  6,  7, 13]])

In [34]: Floats = np.array([0.0, 0.1, 0.12, 3.0, 0.5, 0.63])

In [36]: Floats
Out[36]: array([0.  , 0.1 , 0.12, 3.  , 0.5 , 0.63])
```

نکته:

- ✓ تو بخش مربوط به تعریف آرایه اگه فاصله بین اعضا بذاریم یا نذاریم، نامپی اهمیتی بهشون نمیده.
- ✓ وقتی عدد اعشاری تعریف میکنی، نامپی موقع خروجی ۰ بعد از اعشار رو نادیده میگیره. یعنی 0.0 رو یه صورت .0 خروجی میده.
- ✓ وقتی آرایه چندبعدی درست میکنی یه براکت اضافه هم دور کل مجموعه باید بذاری ([[],[]]). ولی واسه آرایه تک بعدی همون یه براکت دور اعضا کافیه.

در ادامه چند تا از ویژگیهای مهم نامپی و آرایه‌ها رو با هم یاد میگیریم.

ویژگی ndim

واسه مشخص کردن تعداد ابعاد آرایه استفاده میشه.

```
In [12]: Integers.ndim
Out[12]: 2

In [13]: Floats.ndim
Out[13]: 1
```

ویژگی shape

واسه مشخص کردن تعداد سطر و ستون آرایه‌ها استفاده میشه و خروجی رو به صورت Tuples برمیگردونه (یعنی اعدادی داخل پرانتز که با ویرگول از هم جدا شدن). اولین نتیجه میگه آرایه ۲ تا سطر و ۴ تا ستون داره. دومی میگه ۶ تا عضو داره. حتما , بعد از عدد باید باشه.

```
In [39]: Integers.shape
Out[39]: (2, 4)

In [40]: Floats.shape
Out[40]: (6,)
```

ویژگی size

این ویژگی تعداد اعضای آرایه رو برمیگردونه.

```
In [41]: Integers.size
Out[41]: 8

In [42]: Floats.size
Out[42]: 6
```

حلقه for در آرایه

باید به حلقه for بنویسیم که روی تک تک اعضا بررسی انجام بده و ازشون خروجی بگیره.

```
In [43]: for row in Integers:
             print(row)

         [1 2 3 4]
         [ 5  6  7 13]
```

ویژگی flat

واسه اینکه بتونیم آرایه چندبعدی رو تو یه بعد ببینیم استفاده میشه.

end یه پارامتر از تابع print() هست. به شکل عمومی این تابع نتیجه رو تو خطهای متفاوت نشون میده. ولی end کمک میکنه که نتیجه تو یه خط دیده شه. همین کد رو بدون end بنویسیم نتایج رو میتونیم با هم مقایسه کنیم.

```
In [44]: for i in Integers.flat:
             print(i, end="")

         123456713

In [45]: for i in Integers.flat:
             print(i)

         1
         2
         3
         4
         5
         6
         7
         13
```

توابع zeros, ones, full

نامپی یه سری تابع به اسم zeros, ones, full داره که برای ساخت آرایه شامل عدد ۰ یا ۱ یا مقادیر خاصی استفاده میشه.

```
In [47]: np.zeros(6)
Out[47]: array([0., 0., 0., 0., 0., 0.])

In [50]: np.ones((3,4))
Out[50]: array([[1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.]])
```

خروجیها همه عدد اعشاری هستن. میتونیم تعداد سطر و ستون رو هم برای خروجی مشخص کنیم. کد مقابل میگه که بهم تو ۳ تا سطر و ۴ تا ستون عدد ۱ رو به صورت اعشاری بده. یادت باشه که نامپی 0 بعد از اعشار رو نشون نمیده و 1.0 رو به صورت 1. خروجی میده.

اگه بخوایم اعداد رو به صـورت عدد صـحیح دریافت کنیم باید از dtype استفاده کنیم.

```
In [51]: np.ones((3,4), dtype = int)
Out[51]: array([[1, 1, 1, 1],
                [1, 1, 1, 1],
                [1, 1, 1, 1]])
```

از full واســه نوشــتن یه عدد خاص اســتفاده میشــه. کد مقابل تو ۳ ســطر و ۵ ســتون عدد ۱۳ رو به صـورت عدد صـحیح خروجی میده چون براش عدد رو به صورت عدد صحیح مشخص کردیم.

```
In [52]: np.full((3,5), 13)
Out[52]: array([[13, 13, 13, 13, 13],
                [13, 13, 13, 13, 13],
                [13, 13, 13, 13, 13]])
```

تابع arange

یادت باشـه a + range هسـت. دو تا rr نذار. این تابع هر آرگومانی که بهش بدی، از ۰ تا یکی مونده به اون عدد رو به صـورت آرایه بهت خروجی میده. میتونی دو تا آرگومان بهش بدی که از عدد اول تا یکی مونده به آخر رو به صـورت ،رایه خروجی بده. با سـه تا آرگومان هم میشـه آرایه درسـتش کرد که آرگومان سـوم گام رو تعیین میکنه که میتونه عدد منفی هم باشه یعنی از آخر به اول شمرده شه.

```
In [53]: np.arange(10)
Out[53]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [55]: np.arange(3,8)
Out[55]: array([3, 4, 5, 6, 7])

In [56]: np.arange(20,1,-3)
Out[56]: array([20, 17, 14, 11,  8,  5,  2])
```

کد اول یه آرگومان ۱۰ داره پس از ۰ تا ۱۰ خروجی گرفته میشه.

کد دوم عدد اول ۳ و عدد دوم ۸ هست پس از ۳ تا ۷ خروجی گرفته میشه.

کد سوم گام داره که منفی هست یعنی از ۲۰ باید سه تا سه تا برگردیم عقب یا یکی مونده به ۱ یا عدد ۲.

تابع linspace

بهش اولین و آخرین آرگومان رو میدیم. یه ســری عدد از عدد اول تا آخر که عدد آخر هم جزوش هســت رو خروجی میده.

```
In [57]: np.linspace(0,50)
Out[57]: array([ 0.        ,  1.02040816,  2.04081633,  3.06122449,  4.08163265,
                 5.10204082,  6.12244898,  7.14285714,  8.16326531,  9.18367347,
                10.20408163, 11.2244898 , 12.24489796, 13.26530612, 14.28571429,
                15.30612245, 16.32653061, 17.34693878, 18.36734694, 19.3877551 ,
                20.40816327, 21.42857143, 22.44897959, 23.46938776, 24.48979592,
                25.51020408, 26.53061224, 27.55102041, 28.57142857, 29.59183673,
                30.6122449 , 31.63265306, 32.65306122, 33.67346939, 34.69387755,
                35.71428571, 36.73469388, 37.75510204, 38.7755102 , 39.79591837,
                40.81632653, 41.83673469, 42.85714286, 43.87755102, 44.89795918,
                45.91836735, 46.93877551, 47.95918367, 48.97959184, 50.        ])
```

یه کلیدواژه اختیاری به اسم **num** هم داره که تعیین میکنه اعداد با چه فاصله‌ای از هم تو این محدوده خروجی گرفته شن.

```
In [63]: np.linspace(0,50,num=5)
Out[63]: array([ 0. , 12.5, 25. , 37.5, 50. ])
```

reshape

میتونیم یه آرایه از arange درست کنیم و بعد از reshape() استفاده کنیم که شکلش رو تغییر بدیم و آرایه تک بعدش رو به یه آرایه چندبعدی تبدیلش کنیم.

اینجا تو ۵ تا سطر و ۶ تا ستون اعدادی بین ۰ تا ۳۰ رو خروجی میده.

```
In [65]: np.arange(1,31).reshape(5,6)

Out[65]: 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, 26, 27, 28, 29, 30]])
```

اگه مجدد همین دستور رو بنویسیم و براش reshape اعداد جدید تعیین کنیم بهمون خطا میده. چون اعداد محدوده ۰ و ۳۱ تکراری هستن.

```
In [68]: np.arange(1,31).reshape(4,6)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
In [68]:
Line 1:     np.arange(1,31).reshape(4,6)

ValueError: cannot reshape array of size 30 into shape (4,6)
---------------------------------------------------------------------------
```

np.random.randint()

برای دریافت اعداد رندوم از یه محدوده استفاده میشه. کد مقابل بین عدد ۱ تا ۱۹ بهمون ۶ تا عدد خروجی میده که هر بار اجراش کنیم اعداد متفاوتی دریافت میکنیم.

```
In [70]: np.random.randint(1,20,6)

Out[70]: array([18, 18, 15,  1,  4,  9])
```

میتونیم یه متغیر براش تعریف کنیم و بعد بخوایم که مجموع اعضاش رو با **sum** بهمون برگردونه. میتونیم بزرگترین **max** و کوچیکترین عدد **min** یا میانگین اعداد **mean** و همینطور انحراف معیار اعداد **std** رو خروجی بگیریم.

```
In [78]: Rand_Array.sum()

Out[78]: 757

In [79]: Rand_Array.max()

Out[79]: 93

In [80]: Rand_Array.min()

Out[80]: 10

In [81]: Rand_Array.mean()

Out[81]: 50.46666666666667

In [82]: Rand_Array.std()

Out[82]: 28.097133108003902
```

عملگرهای آرایه

میخوایم با عملگرها تو آرایه‌ها کار کنیم.

دو تا متغیر تعریف کن و برای یکی np.arange() و برای دومی np.random.ranint() بنویس و ازشون خروجی بگیر.

```
In [85]: Numbers_1 = np.arange(1,10,2)
         Numbers_Rand = np.random.randint(1,20,5)

In [86]: Numbers_1

Out[86]: array([1, 3, 5, 7, 9])

In [87]: Numbers_Rand

Out[87]: array([ 9,  9, 17, 14,  6])
```

حالا روشون یه سری عملیات ریاضی مثل جمع و ضرب و ... انجام بده. مثل کدهای زیر:

```
In [88]: Numbers_1 + Numbers_Rand
Out[88]: array([10, 12, 22, 21, 15])

In [89]: Numbers_1 * Numbers_Rand
Out[89]: array([ 9, 27, 85, 98, 54])

In [90]: Numbers_1 / 3
Out[90]: array([0.33333333, 1.        , 1.66666667, 2.33333333, 3.        ])

In [91]: Numbers_Rand / 3
Out[91]: array([3.        , 3.        , 5.66666667, 4.66666667, 2.        ])
```

```
In [92]: Numbers_1 += 5

In [93]: Numbers_Rand += 3

In [94]: Numbers_1
Out[94]: array([ 6,  8, 10, 12, 14])

In [95]: Numbers_Rand
Out[95]: array([12, 12, 20, 17,  9])
```

با کمک **sqrt** میتونیم از اعداد جذر بگیری.

```
In [96]: np.sqrt(Numbers_1)
Out[96]: array([2.44948974, 2.82842712, 3.16227766, 3.46410162, 3.74165739])

In [97]: np.sqrt(Numbers_Rand)
Out[97]: array([3.46410162, 3.46410162, 4.47213595, 4.12310563, 3.        ])
```

تو کد بالا هر دو تا آرایه ۵ تا عضو دارن، واسه همین میشه روشون عملیات ریاضی انجام داد. ولی اگه مجموعه داده‌هایی درست کنی که اعضاشون با هم تناظر نداشته باشن خطا دریافت میکنی.

```
In [98]: S_1 = np.arange(1,10,2)
         S_2 = np.arange(1,11,3)

In [99]: S_1
Out[99]: array([1, 3, 5, 7, 9])

In [100]: S_2
Out[100]: array([ 1,  4,  7, 10])

In [101]: S_1 + S_2
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
In [101]:
Line 1:     S_1 + S_2

ValueError: operands could not be broadcast together with shapes (5,) (4,)
---------------------------------------------------------------------------
```

تو کد مقابل اولین آرایه ۵ تا عضو داره و دومی ۴ تا عضو واسه همین خطا میده.

نمایه سازی و برش indexing and slicing

طبق معمول numpy رو با عنوان np وارد میکنیم. یه آرایه با تابع np.random.randint() درست میکنیم که ۸ تا عضو از عدد ۱ تا ۳۰ بهمون بده و بعد ازش خروجی میگیریم.

واسه اینکه به اعضای آرایه دسترسی داشته باشیم از ایندکس استفاده میکنیم که داخل [] می‌آد.

مثلا واسه دسترسی به چهارمین عضو آرایه باید [3] رو جلوی اسم ارایه بنویسیم.

برای اینکه از سومین عضو تا هفتمین عضو آرایه خروجی بگیریم باید بنویسیم [2:7]. حواست باشه که اولین عدد نشونه همون ایندکس هست اینجا میشه ۲۸ ولی آخرین عدد ایندکسی هست که باید عدد با ایندکس قبل از اون رو خروجی بدی یعنی ۵.

[:3] یعنی از اولین عدد تا یکی مونده به چهارمین عدد با ایندکس ۳ رو خروجی بده. ایندکس ۳ عدد ۲۲ هست که تو خروجی نمی‌آد.

```
In [48]: import numpy as np

In [49]: N_Array_1 = np.random.randint(1,30,8)

In [50]: N_Array_1
Out[50]: array([26, 10, 28, 22, 29, 29,  5, 11])

In [51]: N_Array_1[3]
Out[51]: 22

In [52]: N_Array_1[2:7]
Out[52]: array([28, 22, 29, 29,  5])

In [53]: N_Array_1[:3]
Out[53]: array([26, 10, 28])
```

مجدد یه آرایه دیگه تو محدوده ۱ تا ۷ درست میکنیم و ازش خروجی می‌گیریم.

برای دسترسی به اعضاش باز هم باید ایندکس بدیم. اینجا گفته اعداد از اولین عدد تا یکی مونده به ایندکس ۳ یا چهارمین عدد رو تبدیل کن به عدد ۷.

یعنی سه تا عدد اول آرایه تبدیل به عدد ۷ میشه.

```
In [56]: N_Array_2 = np.arange(1,7)

In [57]: N_Array_2
Out[57]: array([1, 2, 3, 4, 5, 6])

In [59]: N_Array_2[:3] = 7

In [60]: N_Array_2
Out[60]: array([7, 7, 7, 4, 5, 6])
```

یه تابع تک بعدی دیگه به صورت رندم درست میکنیم و با متد reshape() تبدیلش میکنیم به یه آرایه چند بعدی که اینجا ۵ تا سطر و ۴ تا ستون بهش داده. بعد ازش خروجی میگیریم که نتیجه رو ببینیم.

```
In [61]: N_Array_3 = np.random.randint(1,100,20)

In [62]: N_Array_3
Out[62]: array([80, 55, 51, 46, 97,  1, 62, 46, 58, 22, 39, 75, 86, 93, 79, 89, 30,
                92, 99, 31])

In [63]: N_Array_3 = N_Array_3.reshape(5,4)

In [64]: N_Array_3
Out[64]: array([[80, 55, 51, 46],
                [97,  1, 62, 46],
                [58, 22, 39, 75],
                [86, 93, 79, 89],
                [30, 92, 99, 31]])
```

```
In [65]: N_Array_3[4]
Out[65]: array([30, 92, 99, 31])

In [66]: N_Array_3[3,0]
Out[66]: 86

In [67]: N_Array_3[:2]
Out[67]: array([[80, 55, 51, 46],
                [97,  1, 62, 46]])
```

واســه دســترســی به آرایه چندبعدی هم به همین شــیوه ایندکســینگ جلو میریم. [4] یعنی ســطر پنجم رو بهم خروجی بده. [3,0] یعنی از سطر چهارم عدد اول رو بهم خروجی بده.

واسه برش زدن آرایه هم از ایندکسینگ استفاده میشه. [:2] یعنی از اولین ســطر تا یکی مونده به ســومین ســطر (ســطر اول و دوم) رو خروجی بده.

```
In [69]: N_Array_3[:,2:4]
Out[69]: array([[51, 46],
                [62, 46],
                [39, 75],
                [79, 89],
                [99, 31]])

In [71]: N_Array_3[:,(2,3)]
Out[71]: array([[51, 46],
                [62, 46],
                [39, 75],
                [79, 89],
                [99, 31]])
```

کد مقابل به دو شــکل نوشــته شــده و نتایج مشــابهی خروجی داده. اولی گفته همه سطرها از ستون سوم تا ستون یکی مونده به پنجم (ستون سوم و چهارم) با همه سطرهاش رو خروجی بده.

میتونیم از tuples استفاده کنیم. یعنی به جای ۲:۴ تو پرانتز نویسیم (2,3) یعنی سطر سوم و چهارم.

```
In [72]: N_Array_4 = np.arange(1,7)

In [73]: N_Array_4
Out[73]: array([1, 2, 3, 4, 5, 6])

In [75]: N_Array_4_SC = N_Array_4.view()

In [76]: N_Array_4_SC
Out[76]: array([1, 2, 3, 4, 5, 6])

In [77]: N_Array_4[3]*=7

In [78]: N_Array_4
Out[78]: array([ 1,  2,  3, 28,  5,  6])
```

تابع view()

یه آرایه دیگه تو محدوده اعداد ۱ تا ۷ درســت میکنیم و ازش خروجی میگیریم. میخوایم اعدادش رو به یه آرایه دیگه هم بدیم. از تابع view() جلوی اســم آرایه قبلی اســتفاده میکنیم.

تو بخش بعدی خواسته عضو چهارم (ایندکس۳) رو در عدد ۷ ضرب کن و بهم خروجی بده. ۴ رو در ۷ ضرب میکنه و ۲۸ رو میده بیرون.

```
In [79]: N_Array_4_SC
Out[79]: array([ 1,  2,  3, 28,  5,  6])

In [80]: N_Array_4_SC[2] += 5

In [81]: N_Array_4_SC
Out[81]: array([ 1,  2,  8, 28,  5,  6])

In [82]: N_Array_4
Out[82]: array([ 1,  2,  8, 28,  5,  6])
```

اگه از آرایه‌ای که با تابع view() درســت کردیم رو مجدد خروجی بگیریم میبینیم که عدد ۲۸ رو جایگزین ۴ کرده. یعنی هر اتفاقی برای آرایه اصــلی بیفته برای این آرایه هم اعمال میشــه. برعکســش هم هســت یعنی اگه برای ارایه فرعی اتفاقی بیفته تو ارایه اصــلی هم همون تغییر اعمال میشــه. اینجا عدد ســوم آرایه فرعی رو با ۵ جمع زدیم و تو آرایه اصلی هم همین تغییر اعمال شد.

تابع copy()

میخوایم ببینیم وقتی از یه آرایه کپی میگیریم و اعضاش رو به یه آرایه دیگه میدیم چه اتفاقی میافته.

```
N_Array_4_DC = N_Array_4.copy()

N_Array_4_DC
array([ 1,  2,  8, 28,  5,  6])

N_Array_4
array([ 1,  2,  8, 28,  5,  6])
```

```
N_Array_4_DC[1] -=5

N_Array_4_DC
array([ 1, -3,  8, 28,  5,  6])

N_Array_4
array([ 1,  2,  8, 28,  5,  6])
```

اگه از تابع کپی شـــده بخوایم که عدد دومش رو از ۵ کم کنه و نتیجه رو خروجی بگیریم متوجه میشیم که تابع اصلی تغییری نمیکنه ولی اگ تابع اصلی رو تغییر بدیم این تابع کپی شده هم تغییر میکنه.

توابع نامپی

میخوایم چند تا از توابع کابردی نامپی رو با هم یاد بگیریم. این توابع لازمه یادگیری Data Science و بعد از اون Machine Learning هستن.

باز هم مثل همیشـــه تو نوت بوک numpy رو وارد کن. یه آرایه با **np.random.randint()** با ســـه تا آرگومان ۱ و ۵۰۰ و ۱۵ بنویس و ازش خروجی بگیر.

```
In [17]: Array_1 = np.random.randint(1,500,15)
         Array_1

Out[17]: array([296,  60,  14, 442, 212, 263, 474, 231, 320, 479, 363, 265, 432,
                437, 263])
```

از تابع **reshape()**. استفاده کن که تو ۳ تا سطر و ۵ تا ستون آرایه قبلی رو خروجی بده.

حالا که یه آرایه چندبعدی درســـت شـــد بیا از توابع نامپی استفاده کنیم.

```
In [25]: Array_2 = Array_1.reshape(3,5)
         Array_2

Out[25]: array([[296,  60,  14, 442, 212],
                [263, 474, 231, 320, 479],
                [363, 265, 432, 437, 263]])
```

تابع np.argpartition()

بر اســاس ســایت اصــلی numpy این تابع با توجه به الگوریتمی که با کمک کلیدواژه kind بهش داده میشــه یه آرایه متناسب به شاخصهایی که بهش داده شده بهمون برمیگردونه.

این تابع چند تا پارامتر داره که باید به ترتیب تو پرانتزش تعریف شه:

- a: اشاره به نام آرایه داره. اینجا اسم آرایه اول رو بهش میدیم Array_1
- kth: به ایندکس عنصر که میخواد تو این بخش شرکت کنه اشاره داره. همه اعضای کوچکتر از این عدد باید قبل این عدد قرار بگیرن و اعداد مساوی یا بزرگتر از این عدد باید بیان بعدش.

- axis: به محور اشاره داره که اعضا در امتدادش مرتب میشن. پیش فرضش ۱- هست. اگه وجود نداشت از آرایه مسطح استفاده میشه.
- kind: الگوریتم رو انتخاب میکنه که پیش فرض introselect هست.
- order: اگه برای آرایه a فیلد تعریف شده باشه، این آرگومان مشخص میکنه که کدوم فیلدها اول، دوم و ... بیان. یه تک فیلد به عنوان یه رشته یا string تعریف میشه و نیازی نیست که همه فیلدها رو به این شکل خاص و ویژه کنیم. ولی معنیش این نیست که از بقیه فیلدها استفاده نمیشه. واسه اینکه بگیم بعد از این فیلد خاص چه فیلدهایی بیان از dtype استفاده میکنیم.

کد زیر رو بنویس که بهتر متوجه شی که چه اتفاقی در خروجی می‌افته.

استفاده از تابع argpartition تو آرایه یک بعدی

آرایه Array_1 شامل مقادیر زیر هست.

```
In [17]: Array_1 = np.random.randint(1,500,15)
         Array_1

Out[17]: array([296,  60,  14, 442, 212, 263, 474, 231, 320, 479, 363, 265, 432,
                437, 263])
```

میخوایم یه کد برای آرایه تک بعدی Arraye_1 بنویسیم که ایندکس ۵ تا از بزرگترین اعداد این مجموعه رو بهمون بده. ۵ عدد بزرگ این مجموعه شامل: ۴۷۹و ۴۷۴و ۴۴۲و ۴۳۷ و ۴۳۲ هست که به ترتیب تو ارایه بالا ایندکشون میشه ۱۲و ۱۳و ۳ و ۹و ۶.

برای نوشتن این کد باید از تابع argpartition() استفاده کنیم.

```
In [19]: Index_Value = np.argpartition(Array_1, -5 )[-5:]
         Index_Value

Out[19]: array([12, 13,  3,  9,  6], dtype=int64)
```

تو کد بالا پایتون بهمون ایندکس بالاترین مقادیر عناصر رو از Array_1 داده. چون تو براکت عدد ۵ رو تعریف کردیم بهمون ایندکس ۵ تا عدد بالا رو میده.

حالا بیا با تابع sort اعداد رو مرتب کنیم که ببینم درست داده یا نه. اعدادی که از آرایه داده با ایندکسهای خروجی مطابقت داره.

```
In [23]: np.sort(Array_1[Index_Value])

Out[23]: array([432, 437, 442, 474, 479])
```

حالا اگه آرایه چندبعدی داشتیم چی؟

استفاده از تابع argpartition تو آرایه چندبعدی

ارایه Arraye_2 یه آرایه چندبعدی هست. کد بالا رو مجدد روی این آرایه مینویسیم که ببینیم چه نتیجه‌ای میده.

```
In [25]: Array_2 = Array_1.reshape(3,5)
         Array_2

Out[25]: array([[296,  60,  14, 442, 212],
                [263, 474, 231, 320, 479],
                [363, 265, 432, 437, 263]])
```

```
In [26]: Index_Value_2 = np.argpartition(Array_2, -5 )[-5:]
         Index_Value_2

Out[26]: array([[2, 1, 0, 3, 4],
                [2, 1, 0, 3, 4],
                [4, 1, 2, 3, 0]], dtype=int64)
```

باز هم براش sort() بنویس که نتیجه رو چک کنی.

میبینی که پایتون تو این شرایط بهمون خطای دسترسی میده.

```
In [27]: np.sort(Array_2[Index_Value_2])

         ---------------------------------------------------------------------------
         IndexError                                Traceback (most recent call last)
         In  [27]:
         Line 1:     np.sort(Array_2[Index_Value_2])

         IndexError: index 3 is out of bounds for axis 0 with size 3
         ---------------------------------------------------------------------------
```

دلیلش اینه که آرایه چندبعدی هست واسه همین تو تعریف تابع argpartition باید مقدار دسترسی رو براش تعریف کنیم. مقدار پیش فرض برای دسترسی ۱ هست که برای آرایه تک بعدی مناسب هست. عدد axis رو بده ۰. علاوه بر اون عدد k باید از تعداد سطرها کمتر باشه. واسه همین بهش ۳ میدیم.

```
In [28]: Index_Value_2 = np.argpartition(Array_2, -3, axis = 0)[-5:]
         Index_Value_2

Out[28]: array([[1, 0, 0, 1, 0],
                [0, 1, 1, 0, 1],
                [2, 2, 2, 2, 2]], dtype=int64)

In [29]: np.sort(Array_2[Index_Value_2])

Out[29]: array([[[231, 263, 320, 474, 479],
                 [ 14,  60, 212, 296, 442],
                 [ 14,  60, 212, 296, 442],
                 [231, 263, 320, 474, 479],
                 [ 14,  60, 212, 296, 442]],

                [[ 14,  60, 212, 296, 442],
                 [231, 263, 320, 474, 479],
                 [231, 263, 320, 474, 479],
                 [ 14,  60, 212, 296, 442],
                 [231, 263, 320, 474, 479]],

                [[263, 265, 363, 432, 437],
                 [263, 265, 363, 432, 437],
                 [263, 265, 363, 432, 437],
                 [263, 265, 363, 432, 437],
                 [263, 265, 363, 432, 437]]])
```

واسه اینکه بتونیم ایندکس بزرگترین اعداد رو داشته باشیم باید از متد ()ravel استفاده کنیم که نتیجه رو از این وضعیت در بیاریم. یعنی محور یا axis بهش ندیم و عدد ۳- رو هم به ۵- دربیاریم چون دیگه محدودیت اینکه باید عدد k از تعداد سطرهای کمتر باشه رو نداریم.

```
In [30]: Index_Value_3 = np.argpartition(Array_2.ravel(), -5)[-5:]
         Index_Value_3

Out[30]: array([12, 13,  3,  9,  6], dtype=int64)
```

اگه با Index_Value عادی مقایسه‌اش کنیم میبینیم که نتیجه یکسان میشه.

```
Out[30]: array([12, 13,  3,  9,  6], dtype=int64)

In [31]: Index_Value

Out[31]: array([12, 13,  3,  9,  6], dtype=int64)
```

حالا اگه بخوایم به مقادیر Array_2 نگاه کنیم بهمون خطا میده چون چندبعدی هســـت و Index_Value_3 به عنوان فیلتر اینجا عمل میکنه.

```
In [32]: Array_2[Index_Value_3]

         ---------------------------------------------------------------------------
         IndexError                                Traceback (most recent call last)
         In [32]:
         Line 1:     Array_2[Index_Value_3]

         IndexError: index 12 is out of bounds for axis 0 with size 3
         ---------------------------------------------------------------------------
```

واسه حلش باید از **unravel_index()** به همراه **shape** استفاده کنیم.

```
In [33]: Index_Value_3_UR = np.unravel_index(Index_Value_3, Array_2.shape)

In [34]: Array_2[Index_Value_3_UR]

Out[34]: array([432, 437, 442, 479, 474])
```

حالا بیا نتیجه دو تا آرایه رو با هم مقایســـه کنیم. می‌بینیم که نتایج یکسان میشه:

```
In [34]: Array_2[Index_Value_3_UR]
Out[34]: array([432, 437, 442, 479, 474])

In [35]: Array_1[Index_Value]
Out[35]: array([432, 437, 442, 479, 474])
```

تابع np.allclose()

واسـه تطبیق دادن دو تا آرایه با هم اسـتفاده میشـه که نتیجه مقدار بولینی هسـت. این تابع هم مثل تابع بالا چند تا پارامتر داره:

- a و b: اسم دو تا آرایه هستن که میخوایم با هم مقایسه‌شون کنیم.
- rtol: بهش پارامتر تحمل نســبی یا the relative tolerance parameter میگن که عددش برابر هســت با rtol=1e-05
- atoll: بهش پارامتر تحمل مطلق یا the absolute tolerance parameter میگن که عددش برابر هســت با atol=1e-08
- equl_nan: nan مخفف not a number هســت یعنی عدد نیســت. وقتی بخوایم nan ها رو تو دو تا آرایه مقایسه کنیم. اگه nan تو آرایه a برابر nan تو آرایه b باشه True برمیگرده در غیر اینصورت False.

کد زیر رو ببین. مقدار تحمل یا tolerance رو اینجا داده 0.1. نتیجه بهمون Shape error میده.

```
In [40]: np.allclose(Array_1,Array_2,0.1)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
In  [40]:
Line 1:     np.allclose(Array_1,Array_2,0.1)

File <__array_function__ internals>, in allclose:
Line 5:

File E:\ArcGIS Pro\bin\Python\envs\arcgispro-py3\lib\site-packages\numpy\core\numeric.py,
in allclose:
Line 2256:  res = all(isclose(a, b, rtol=rtol, atol=atol, equal_nan=equal_nan))

File <__array_function__ internals>, in isclose:
Line 5:

File E:\ArcGIS Pro\bin\Python\envs\arcgispro-py3\lib\site-packages\numpy\core\numeric.py,
in isclose:
Line 2365:  return within_tol(x, y, atol, rtol)

File E:\ArcGIS Pro\bin\Python\envs\arcgispro-py3\lib\site-packages\numpy\core\numeric.py,
in within_tol:
Line 2346:  return less_equal(abs(x-y), atol + rtol * abs(y))

ValueError: operands could not be broadcast together with shapes (15,) (3,5)
---------------------------------------------------------------------------
```

مثل قبل باید از ravel() واســـه آرایه چندبعدی استفاده کنیم که مشکل حل شه باید نتیجه True یا False باشه.

```
In [41]: np.allclose(Array_1,Array_2.ravel(),0.1)
Out[41]: True
```

میخوایم ببینیم که مقدار تحمل یا Tolerance کاربردش چیه. دو تا آرایه تک بعدی جدیدی به اسم Array_3 و Array_4 با ۵ تا مقدار متفاوت تعریف میکنیم. با تابع allclose() یکبار عدد تحمل رو 0.1 میدیم و یکبار 0.2 که ببینیم چه نتیجه‌ای بهمون میده.

```
In [43]: Array_3 = np.array([0.12,0.17,0.24,0.29,0.33])
         Array_4 = np.array([0.13,0.19,0.26,0.31,0.33])
         np.allclose(Array_3, Array_4, 0.1)

Out[43]: False

In [44]: np.allclose(Array_3, Array_4, 0.2)

Out[44]: True
```

اعداد دو تا آرایه رو با هم مقایسه کن، حداکثر به اندازه ۰/۲ با هم فاصله دارن یعنی حد تحمل 0.2 براشون True رو برمیگردونه ولی 0.1 مقدار False بهمون میده.

تابع np.clip()

مقادیر آرایه رو برش میزنه یا کمترشون میکنه. باز هم مثل دو تا تابع بالا چند تا پارامتر داره.

- a: اسم آرایه اصلی که میخوایم روش clip یا برش انجام بدیم.
- a_min, a_max: یه عدد به عنوان بزرگترین و کوچکترین عدد آرایه بهش میدیم و تو خروجی اگه عدد از مینیمم کمتر بود خود عدد مینیمم برمیگرده و اگه بیشتر بود خود عدد ماکزیمم
- out: آرایه خروجی باید تو این بشینه باید shape یا شکلش با آرایه اول یکی باشه. البته نوشتنش اختیاری هست.
- **kwargs: به بقیه کلیدواژه‌های پارامتر اشاره داره.

تو کد پایین عدد مینیمم رو داده ۱۰۰ و ماکزیمم رو داده ۲۵۰. یعنی اگه تو آرایه اصلی عددی زیر ۱۰۰ باشه خود ۱۰۰ نشون داده میشه و اگه عددی بالای ۲۵۰ باشه به جاش ۲۵۰ نشون داده میشه.

```
In [45]: np.clip(Array_1,100,250)

Out[45]: array([250, 100, 100, 250, 212, 250, 250, 231, 250, 250, 250, 250, 250,
                250, 250])

In [46]: Array_1

Out[46]: array([296,  60,  14, 442, 212, 263, 474, 231, 320, 479, 363, 265, 432,
                437, 263])
```

همین کار رو با آرایه چندبعدی انجام بدی هم نتیجه مشابه میشه.

```
In [47]: np.clip(Array_2,100,250)

Out[47]: array([[250, 100, 100, 250, 212],
                [250, 250, 231, 250, 250],
                [250, 250, 250, 250, 250]])
```

تابع np.where()

تو نامپی متد index وجود نداره و به جاش از np.where() استفاده میشه.

numpy.where(condition, [x, y,]/)

یه عنصر رو که از x یا y انتخاب شده بر اساس شرطی که بهش میدیم بهمون برمیگردونه (تمرین این تابع و توابع بعدی رو میذاریم تو بخش حل چند تا تمرین).

تابع np.pad()

واسه اضافه کردن پدینگ به آرایه استفاده میشه. پدینگ یعنی اضافه کردن یه مقدار به لبه‌های آرایه.

numpy.pad(array, pad_width, mode='constant', **kwargs)

آرایه‌ای که میده با آرایه اول رتبه برابر داره. pad_width هم مشـــخص میکنه که عددی که به لبه‌های هر محور آرایه اضافه شده چی هست.

بخش mode شامل توابع constant, edge, linear_ramp, reflect, symmetric و wrap هست که پیش فرض روی constant هست و وقتی انتخابش کنیم باید constant_values هم براش بنویسیم.

تابع np.put()

عناصر خاصی از یه آرایه رو با مقادیر داده شده جایگزین میکنه.

numpy.put(a, ind, v, mode='raise')

- a: آرایه هدف هست که میخوایم عناصرش رو تغییر بدیم.
- ind: ایندکسهای هدف که به صورت عدد صحیح هستن.
- v: مقداری هست که میخوایم در آرایه هدف با ایندکس مشخص قرارش بدیم. اگه v از ind کوچیکتر باشه به صورت تکراری پر میشه.
- mode: یه گزینه هست که رفتار ایندکسهای خارج زا محدوده رو تعریف میکنه و سه تا مقدار raise, wrap, clip قبول میکنه. raise پیش فرض هست و معنیش میشه اضافه کردن خطا. wrap به معنی پیچیدن دور چیزی و clip به معنی برش دادن به بازه هست.

تابع np.random.choice()

یه نمونه تصادفی از یه آرایه تک بعدی ایجاد میکنه.

numpy,random.choice(a, size=None, replace=True, p=None)

- a: یه آرایه تک بعدی اولیه هست.
- size: مقدار پیش فرضش None هست. ولی میتونه مقادیری به صورت tuples داشته باشه.
- replace: یه عدد بولینی قبول میکنه. پیش فرضـــش True هســـت. یعنی مقادیر a میتونن چندین بار انتخاب بشن.
- p: احتمال هر ورودی در a رو بررسـی میکنه. اگه این بخش رو تعریف نکنیم فرض رو بر این گذاشـتیم که همه ورودیهای a توزیع یکنواختی دارن.

تابع np.random.rand()

این تابع شـبیه تابع np.random.rand() هسـت. فرقشـون اینه که تابع rand اعداد تصـادفی اعشـاری بین ۰ تا ۱ تولید میکنه ولی randint یه سـری اعداد تصـادفی تو محدوده‌ای که براش تعیین میشـه خروجی میده. اینکه کدوم رو انتخاب کنیم بسته به نیازمون برای تولید اعداد تصادفی در کد پایتون هست.

np.random.rand(d0, d1, ..., dn)

داخل پرانتزش ابعاد آرایه رو میتونیم بدیم.

خلاصه دستورهای numpy

دستورها	توضیحات
Import **numpy** as **np**	وارد کردن numpy به نوت بوک ArcGIS Pro
np.array()	ساخت آرایه به همراه عضوهای آرایه
type()	تشخیص نوع آرایه
.ndim	مشخص کردن تعداد ابعاد آرایه
.shape	مشخص کردن تعداد سطر و ستون آرایه‌ها (خروجی رو به صورت Tuples برمیگردونه (یعنی اعدادی داخل پرانتز که با ویرگول از هم جدا شدن: (2,4) یا (6,))
.size	مشخص کردن تعداد اعضای آرایه
for / in	بررسی تک تک اعضا و خروجی گرفتن از همه اعضا
.flat	دیدن آرایه **چندبعدی** در **یک بعد**
np.zeros() **np.ones((3,5), dtype = int)** **np.full()**	تولید آرایه با اعضای ۰ تولید آرایه با اعضای ۱ تولید آرایه با اعضای خاص اگه بخوایم اعداد رو به صورت عدد صحیح دریافت کنیم باید از dtype استفاده کنیم.
print (i, end =" ")	end یه پارامتر از تابع ()print هست. به شکل عمومی، این تابع نتیجه رو تو خط‌های متفاوت نشون میده. ولی end کمک میکنه که نتیجه تو یه خط دیده شه.
np.arange()	این تابع هر آرگومانی که بهش بدی، از ۰ تا یکی مونده به اون عدد رو به صورت آرایه بهت خروجی میده. میتونی دو تا آرگومان بهش بدی که از عدد اول تا یکی مونده به آخر رو به صورت ،رایه خروجی بده. با سه تا آرگومان هم میشه آرایه درستش کرد که آرگومان سوم گام رو تعیین میکنه که میتونه عدد منفی هم باشه یعنی از آخر به اول شمرده شه.
np.linspace(0 , 5, num =5)	بهش اولین و آخرین آرگومان رو میدیم. یه سری عدد از عدد اول تا آخر که عدد آخر هم جزوش هست رو خروجی میده. یه کلیدواژه اختیاری به اسم num هم داره گام رو مشخص میکنه.
.reshape	تبدیل آرایه یک بعدی به آرایه چندبعدی
np.random.randint()	برای دریافت اعداد رندوم از یه محدوده
.sum() **.max()** **.min()** **.mean()** **.std()** **.sqr()**	مجموع اعداد بزرگترین عدد آرایه کوچکترین عدد آرایه میانگین اعداد انحراف معیار اعداد جذر گرفتن از اعداد

[]/ [3:7]/ [:3]/ [3,0]/ [:,2:4]/ [:,(2,3)]	واسه دسترسی به اعضای آرایه با شماره ایندکس عضوها
.view()	از روی تابع اصلی یه تابع فرعی با همون اعضا میسازیم که با انجام عملیات روی هر کدوم از توابع، روی تابع دیگه هم اعمال میشه.
.copy()	از روی تابع اصلی یه تابع فرعی با همون اعضا میسازیم که با انجام عملیات روی تابع اصلی، تابع فرعی هم تغییر میکنه ولی برعکسش صادق نیست.
np.argpartition(a, kth, axis=-1, kind='introselect', order=None)	این تابع با توجه به الگوریتمی که با کمک کلیدواژه kind بهش داده میشه یه آرایه متناسب به شاخصهایی که بهش داده شده بهمون برمیگردونه. مثلا ایندکس ۵ تا از بزرگترین اعضای مجموعه رو بده. چند تا پارامتر داره که باید به ترتیب تو پرانتزش تعریف شه: - **a:** اشاره به نام آرایه داره. اینجا اسم آرایه اول رو بهش میدیم Array_1 - **kth:** به ایندکس عنصر که میخواد تو این بخش شرکت کنه اشاره داره. همه اعضای کوچکتر از این عدد باید قبل این عدد قرار بگیرن و اعداد مساوی یا بزرگتر از این عدد باید بیان بعدش. (این عدد باید از تعداد سطرهای آرایه چندبعدی کمتر باشه). - **axis:** به محور اشاره داره که اعضا در امتدادش مرتب میشن. پیش فرضش ۱- هست. اگه وجود نداشت از آرایه مسطح استفاده میشه (برای آرایه چند بعدی عددش رو ۰ میدیم). - **kind:** الگوریتم رو انتخاب میکنه که پیش فرض introselect هست. - **order:** اگه برای آرایه a فیلد تعریف شده باشه، این آرگومان مشخص میکنه که کدوم فیلدها اول، دوم و ... بیان. یه تک فیلد به عنوان یه رشته یا string تعریف میشه و نیازی نیست که همه فیلدها رو به این شکل خاص و ویژه کنیم. ولی معنیش این نیست که از بقیه فیلدها استفاده نمیشه. واسه اینکه بگیم بعد از این فیلد خاص چه فیلدهایی بیان از dtype استفاده میکنیم.
np.sort()	واسه مرتب کردن اعضای آرایه از کوچکترین به بزرگترین
.raval() .unravel_index()	واسه داشتن ایندکس بزرگترین اعداد تو آرایه چندبعدی. در این صورت نیازی نیست axis بدیم. عدد k هم نیازی نیست از تعداد سطرها کمتر باشه. واسه رفع خطای axis و سایز از unrevel_index() استفاده میشه.
Np.allclose(a, b, rtol=1e-05, atol=1e-08, equal_nan=False)	واسه تطبیق دادن دو تا آرایه با هم استفاده میشه که نتیجه مقدار بولینی هست:

	✚ a و b: اسم دو تا آرایه هستن که میخوایم با هم مقایسه‌شون کنیم. ✚ rtol: بهش پارامتر تحمل نسبی یا the relative tolerance parameter میگن که عددش برابر هست با rtol=1e-05 ✚ atoll: بهش پارامتر تحمل مطلق یا the absolute tolerance parameter میگن که عددش برابر هست با atol=1e-08 ✚ equl_nan: nan مخفف not a number هست یعنی عدد نیست. وقتی بخوایم nan ها رو تو دو تا آرایه مقایسه کنیم. اگه nan تو آرایه a برابر nan تو آرایه b باشه True برمیگرده در غیر اینصورت False. منظور از عدد تحمل، اختلاف اعداد متناظر دو تا آرایه هست.
np.clip(a, a_min, a_max, out=None, **kwargs)	مقادیر آرایه رو برش میزنه یا کمترشون میکنه: ✚ a: اسم آرایه اصلی که میخوایم روش clip یا برش انجام بدیم. ✚ a_min, a_max: یه عدد به عنوان بزرگترین و کوچکترین عدد آرایه بهش میدیم و تو خروجی اگه عدد از مینیمم کمتر بود خود عدد مینیمم برمیگرده و اگه بیشتر بود خود عدد ماکزیمم ✚ out: آرایه خروجی باید تو این بشینه باید shape یا شکلش با آرایه اول یکی باشه. البته نوشتنش اختیاری هست. ✚ **kwargs: به بقیه کلیدواژه‌های پارامتر اشاره داره. عدد ماکزیمم و مینیمم نشون میده که اگه عددی زیر یا بالای این دو تا باشه، خود عدد ماکزیمم و مینیمم براشون خروجی گرفته شه.
np.where(condition, [x, y,]/)	یه عنصر رو که از x یا y انتخاب شده بر اساس شرطی که بهش میدیم بهمون برمیگردونه.
np.pad(array, pad_width, mode='constant', **kwargs)	واسه اضافه کردن پدینگ به آرایه استفاده میشه. پدینگ یعنی اضافه کردن یه مقدار به لبه‌های آرایه. ✚ array: آرایه‌ای که میده با آرایه اول رتبه برابر داره. ✚ pad_width: هم مشخص میکنه که عددی که به لبه‌های هر محور آرایه اضافه شده چی هست. ✚ بخش mode شامل توابع constant, edge, linear_ramp, reflect, symmetric و wrap هست که پیش فرض روی constant هست و وقتی انتخابش کنیم باید constant_values هم براش بنویسیم.
np.put(a, ind, v, mode='raise')	عناصر خاصی از یه آرایه رو با مقادیر داده شده جایگزین میکنه.

	a: آرایه هدف هست که میخوایم عناصرش رو تغییر بدیم. **ind:** ایندکسهای هدف که به صورت عدد صحیح هستن. v: مقداری هست که میخوایم در آرایه هدف با ایندکس مشخص قرارش بدیم. اگه v از ind کوچیکتر باشه به صورت تکراری پر میشه. **mode:** یه گزینه هست که رفتار ایندکسهای خارج زا محدوده رو تعریف میکنه و سه تا مقدار raise, wrap, clip قبول میکنه. raise پیش فرض هست و معنیش میشه اضافه کردن خطا. wrap به معنی پیچیدن دور چیزی و clip به معنی برش دادن به بازه هست.
np.random.choice(a, size=None, replace=True, p=None)	یه نمونه تصادفی از یه آرایه تک بعدی ایجاد میکنه. **a:** یه آرایه تک بعدی اولیه هست. **size:** مقدار پیش فرضش None هست. ولی میتونی مقادیری به صورت tuples داشته باشه. **replace:** یه عدد بولینی قبول میکنه. پیش فرضش True هست. یعنی مقادیر a میتونن چندین بار انتخاب بشن. **p:** احتمال هر ورودی در a رو بررسی میکنه. اگه این بخش رو تعریف نکنیم فرض رو بر این گذاشتیم که همه ورودیهای a توزیع یکنواختی دارن.
np.random.rand(d0, d1, ..., dn)	اعداد تصادفی اعشاری بین ۰ تا ۱ تولید میکنه.
.astype()	تبدیل اعداد به هم مثلا از اعشاری به عدد صحیح

حل چند تا تمرین

تمرین ۱: یه آرایه به اسم Array_R_1 با ۲۰ عضو به صورت رندم از ۱ تا ۱۵۰ ایجاد کن.

تمرین ۲: مجموع، مقدار ماکزیمم، مینیمم، میانگین و انحراف معیار آرایه تمرین قبل Array_R_1 رو محاسبه کن.

تمرین ۳: یه آرایه جدید به اسم Array_2 از ۱ تا ۵۹ با گام ۳ ایجاد کن. بعد این آرایه رو با آرایه قبلی Array_R_1 جمع کن.

تمرین ۴: از یازدهمین عضو آرایه اول Array_R_1 خروجی بگیر.

تمرین ۵: یه Index_Value رندم واسه آرایه اول Array_R_1 با ۱ عضو درست کن که هر بار که اجراش میکنی یه عنصر متفاوت دریافت کنی.

تمرین ۶: تمرین بالا رو با تابع np.where() انجام بده.

تمرین ۷: آرایه اول Array_R_1 رو به یه آرایه چندبعدی تو ۴ سطر و ۵ ستون تبدیل کن.

تمرین ۸: آرایه چندبعدی که تو تمرین قبل تولید کردی رو برش بزن طوری که از ستون دوم تا انتها رو بهمون خروجی بده.

تمرین ۹: تابع Array_R_2 که تو تمرین ۷ تولید شده رو با عدد ۰ در هر ۴ سمت آرایه چنددبعدی پدینگ کن.

تمرین ۱۰: یه تابع به اسم Array_R_3 با ۳۰ عضو به صورت رندم از ۱ تا ۱۰۰ ایجاد کن. بعد به یه تابع چندبعدی با ۶ تا سطر و ۵ تا ستون تبدیلش کن و بعد عناصر هر سطر رو از کم به زیاد مرتب کن.

تمرین ۱۱: به صورت رندم بعضی از عناصر تابع Array_R_3 رو به عدد ۱ تغییر بده.

تمرین ۱۲: یه آرایه یک بعدی با محدوده ۱ تا ۱۰۱ به اسم Array_5 تولید کن و به یه آرایه ۱۰ در ۱۰ تغییرش بده. بعد مجدد ازش بخواه که از سطر ۶ به بعد این آرایه رو برات برش بزنه.

تمرین ۱۳: تو تمرین قبلی Array_5 از سطر ۴ و ستون ۵ به بعد رو برش بزن.

تمرین ۱۴: تو تمرین قبلی Array_5 سطر سوم تا سطر پنجم و از ستون پنجم تا ستون ششم رو برش بزن.

تمرین ۱۵: تو تمرین قبلی Array_5 دهمین یا آخرین سطر آرایه رو خروجی بگیر.

تمرین ۱۶: دو تا آرایه تک بعدی به اسم Array_A و Array_B به صورت جداگونه بساز و برای هر کدوم ۱۵ تا عدد به صورت رندم از ۰ تا ۱ اختصاص بده. بعد اعداد هر کدوم رو در ۱۰ ضرب کن. اعداد خروجی رو به عدد صحیح تبدیل کن و بعد به یه آرایه با ۳ تا سطر و ۵ تا ستون تبدیلشون کن و ازشون خروجی بگیر.

تمرین ۱۷: دو تا آرایه سوال قبل رو با هم جمع، ضرب و از هم تقسیم کن.

جواب تمرینها

جواب تمرین ۱: باید با تابع np.random.rantint() بنویسمش که سه تا آرگومان داره. عدد اول و دوم محدوده اعداد رو مشخص میکنن و عدد سوم تعداد اعضای آرایه رو.

```
In [1]: import numpy as np

In [3]: Array_R_1 = np.random.randint(1,150,20)

In [4]: Array_R_1

Out[4]: array([135,  37,  54,  84, 103,  23,  75,   2,  12, 104, 126,   1,  39,
                8,  37,  95,  50,  84,  18,  40])
```

جواب تمرین ۲: باید از متدهای sum()/ max()/ min()/ mean()/std() استفاده کنیم.

```
In [5]: Array_R_1.sum()

Out[5]: 1127

In [6]: Array_R_1.max()

Out[6]: 135

In [7]: Array_R_1.min()

Out[7]: 1

In [8]: Array_R_1.mean()

Out[8]: 56.35

In [9]: Array_R_1.std()

Out[9]: 40.695546439383264
```

جواب تمرین ۳: باید با تابع np.arange() بنویســـمش. چون از ۱ تا ۵۹ رو خواســـته باید عدد ۶۰ رو به عنوان دومین آرگومان وارد کنیم چون همیشـــه یه عدد مونده به آخر رو بهمون میده نه آخرین عدد رو. عدد ســـوم هم در واقع گام هست. بعد با عملگر + دو تا آرایه رو جمع بزنیم:

```
In [10]: Array_2 = np.arange(1,60,3)

In [11]: Array_2

Out[11]: array([ 1,  4,  7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46,
49,
       52, 55, 58])

In [12]: Array_2 + Array_R_1

Out[12]: array([136,  41,  61,  94, 116,  39,  94,  24,  37, 132, 157,  35,  76,
                48,  80, 141,  99, 136,  73,  98])
```

جواب تمرین ۴: باید از [] به همراه شماره ایندکس استفاده کنیم. ایندکس اعضا از ۰ شروع میشه پس ایندکس عدد یازدهم میشه ۱۰.

```
In [14]: Array_R_1

Out[14]: array([135,  37,  54,  84, 103,  23,  75,   2,  12, 104, 126,   1,  39,
                8,  37,  95,  50,  84,  18,  40])

In [15]: Array_R_1[10]

Out[15]: 126
```

جواب تمرین ۵: باید تابع np.random.randint() رو داخل براکت [] جلوی اسم آرایه اول یا Array_R_1 بنویسیم. این آرایه ۲۰ تا عضو داره که اگه بخوایم بذاریمش تو پرانتز باید بنویسیم اعداد از محدوده ۱ تا ۲۱ که همین ایندکسها رو برداره. هربار که این کد رو اجرا کنی یه ایندکس شانسی از ۱ تا ۲۱ برمیداره و عدد معادلش رو از اعضای آرایه اول خروجی میده. میتونی با یه اسم جدید این آرایه رو تولید کنی اینجا اسمش رو گذاشتیم R_Element.

```
In [21]: R_Element = Array_R_1[np.random.randint(1,21,1)]

In [22]: R_Element

Out[22]: array([8])
```

جواب تمرین ۶: تابع np.where(condition, [x, y,]/) یه عنصر رو که از x یا y انتخاب شده بر اساس شرطی که بهش میدیم بهمون برمیگردونه. باید تو پرانتز where() بگیم که آرایه اول Array_R_1 با آرایه‌ای که تو تمرین قبل تعریف کردیم R_Element معادل هست. کلش رو هم به اسم Index_Value تعریف میکنیم.

```
In [23]: Index_Value = np.where(Array_R_1 == R_Element)

In [24]: Index_Value

Out[24]: (array([13], dtype=int64),)
```

جواب تمرین ۷: واسه تبدیل آرایه تک بعدی به آرایه چند بعدی باید از متد reshape() استفاده کنیم. این ارایه رو به اسم Array_R_2 ذخیره میکنیم.

```
In [25]: Array_R_2 = Array_R_1.reshape(4, 5)

In [26]: Array_R_2

Out[26]: array([[135,  37,  54,  84, 103],
                [ 23,  75,   2,  12, 104],
                [126,   1,  39,   8,  37],
                [ 95,  50,  84,  18,  40]])
```

جواب تمرین ۸: واسه برش زدن باید از [] استفاده کنیم. چون همه سطرها رو خواسته [:] شروع میکنیم. بعد از ستون دوم خواسته تا ستون آخر. ایندکس ستون دم ۱ هست و ایندکس ستون آخر ۴ پس یه کاما میذاریم و مینویسیم [:,1:4] ولی میشه یه شکل زیر هم نوشتش.

```
In [27]: Array_R_2[:,(1,4)]

Out[27]: array([[ 37, 103],
                [ 75, 104],
                [  1,  37],
                [ 50,  40]])
```

جواب تمرین ۹: باید از تابع np.pad(array, pad_width, mode='constant', **kwargs) استفاده کنیم. وقتی متد constant باشه یعنی یه عدد ثابت بخوایم بدیم باید خود عدد ثابت رو هم معرفی کنیم که اینجا constant_values رو ۰ داده. یکبار در هر ۴ سمت این عدد باید وارد شه. یعنی باید pad_width =1 باشه. اسم این آرایه رو Array_R_B میدیم.

```
In [29]: Array_R_B = np.pad(Array_R_2, pad_width = 1, mode = "constant", constant_values = 0)

In [31]: Array_R_B

Out[31]: array([[  0,   0,   0,   0,   0,   0,   0],
                [  0, 135,  37,  54,  84, 103,   0],
                [  0,  23,  75,   2,  12, 104,   0],
                [  0, 126,   1,  39,   8,  37,   0],
                [  0,  95,  50,  84,  18,  40,   0],
                [  0,   0,   0,   0,   0,   0,   0]])
```

جواب تمرین ۱۰: اول با تابع np.random.randint() یه آرایه یک بعدی به اســم Array_R_3 تعریف میکنیم. بعد با متد reshape() به آرایه چند بعدی تبدیلش میکنیم و بعد با متد sort() اعداد هر سطرش رو مرتب میکنیم.

```
In [32]: Array_R_3 = np.random.randint(1,100,30)

In [33]: Array_R_3 = Array_R_3.reshape(6,5)

In [34]: Array_R_3

Out[34]: array([[87, 77, 20, 27, 41],
                [12, 97, 72, 98, 63],
                [54, 12, 62, 77, 15],
                [74, 86, 13, 36, 51],
                [43, 87, 84, 55, 81],
                [83, 78, 85, 35, 11]])

In [35]: Array_R_3.sort()

In [36]: Array_R_3

Out[36]: array([[20, 27, 41, 77, 87],
                [12, 63, 72, 97, 98],
                [12, 15, 54, 62, 77],
                [13, 36, 51, 74, 86],
                [43, 55, 81, 84, 87],
                [11, 35, 78, 83, 85]])
```

جواب تمرین ۱۱: تمرین ازمون خواسته که یه سری عدد رو به صورت رندم انتخاب کنیم و تبدیلشون کنیم به عدد ۱.
چون انتخاب رو به صــورت رندم خواســته باید از تابع np.random.choice(a, size=None, replace=True, p=None) استفاده کنیم.
تو مرحله دوم چون خواسـته که این عدد رو بذاریم تو آرایه جدید باید از تابع np.put(a, ind, v, mode='raise') اسـتفاده شه.
اول تابع np.put() رو مینویسـیم که نیاز به یه آرایه اولیه داره. آرایه اولیه Array_R_3 هسـت. بعد مقدار ind رو میخواد که باید ایندکسـهای هدف رو بهش به صـورت عدد صـحیح بدیم. نمیدونیم کدوم ایندکس هسـت چون میخوایم رندم انتخاب شــه پس از تابع np.random.choice() به جای ind اســتفاده میکنیم. داخل پرانتز تابع np.random.choice() ازمون size رو میخواد که به صـورت range(5,6) براش تعریفش میکنم. بعد تو همین تابع باید replace تعریف شـه که پیش فرض True هســت یعنی مقادیر a میتونن چندین بار انتخاب شــن. ما بهش False میدیم که این اتفاق نیفته. پرانتزش رو میبندیم و برمیگردیم سـراغ تابع np.put() که الان ازمون v میخواد یعنی مقداری که میخوایم در آرایه هدف با ایندکس مشخص قرارش بدیم. بهش عدد ۱ رو میدیم یعنی به صورت رندم یه سری عدد رو با ۱ جایگزین کن.

```
In [36]: Array_R_3

Out[36]: array([[20, 27, 41, 77, 87],
                [12, 63, 72, 97, 98],
                [12, 15, 54, 62, 77],
                [13, 36, 51, 74, 86],
                [43, 55, 81, 84, 87],
                [11, 35, 78, 83, 85]])

In [37]: np.put(Array_R_3, np.random.choice(range(5,6), replace = False),1)

In [38]: Array_R_3

Out[38]: array([[20, 27, 41, 77, 87],
                [ 1, 63, 72, 97, 98],
                [12, 15, 54, 62, 77],
                [13, 36, 51, 74, 86],
                [43, 55, 81, 84, 87],
                [11, 35, 78, 83, 85]])
```

جواب تمرین ۱۲: با تابع np.arange() یه آرایه با دو آرگومان ۱ و ۱۰۱ ایجاد میکنیم و با متد reshape(). عدد سطر و ستون رو ۱۰ میدیم که چندبعدی شه. برای برش زدن باید از [] به همراه ایندکس سطر ششم که اینجا ۵ هست و دونقطه استفاده کنیم.

```
In [39]: Array_5 = np.arange(1,101).reshape(10,10)

In [40]: Array_5

Out[40]: 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,  26,  27,  28,  29,  30],
                [ 31,  32,  33,  34,  35,  36,  37,  38,  39,  40],
                [ 41,  42,  43,  44,  45,  46,  47,  48,  49,  50],
                [ 51,  52,  53,  54,  55,  56,  57,  58,  59,  60],
                [ 61,  62,  63,  64,  65,  66,  67,  68,  69,  70],
                [ 71,  72,  73,  74,  75,  76,  77,  78,  79,  80],
                [ 81,  82,  83,  84,  85,  86,  87,  88,  89,  90],
                [ 91,  92,  93,  94,  95,  96,  97,  98,  99, 100]])

In [41]: Array_5[5:]

Out[41]: array([[ 51,  52,  53,  54,  55,  56,  57,  58,  59,  60],
                [ 61,  62,  63,  64,  65,  66,  67,  68,  69,  70],
                [ 71,  72,  73,  74,  75,  76,  77,  78,  79,  80],
                [ 81,  82,  83,  84,  85,  86,  87,  88,  89,  90],
                [ 91,  92,  93,  94,  95,  96,  97,  98,  99, 100]])
```

جواب تمرین ۱۳: باید به صورت [] عدد سطر به همراه : بیاد و با کاما دنبال شه بعد عدد ستون با دونقطه بیاد.

```
In [42]: Array_5[4:, 5:]

Out[42]: array([[ 46,  47,  48,  49,  50],
                [ 56,  57,  58,  59,  60],
                [ 66,  67,  68,  69,  70],
                [ 76,  77,  78,  79,  80],
                [ 86,  87,  88,  89,  90],
                [ 96,  97,  98,  99, 100]])
```

جواب تمرین ۱۴: باز هم باید به صورت [] از سطر با ایندکس ۲ تا یکی مونده به سطر با ایندکس ۵ [2:5] یعنی سطر سوم تا پنجم رو خواسته و ستون با ایندکس ۴ تا یه ستون مونده به ایندکس ۷ [4:7] یعنی ستون پنجم تا ستون ششم رو خواسته.

```
In [43]: Array_5[2:5, 4:7]

Out[43]: array([[25, 26, 27],
                [35, 36, 37],
                [45, 46, 47]])
```

جواب تمرین ۱۵: باز هم باید از [] استفاده کینم. سوال در واقع خواسته که اعداد کل ستونهای ۱ تا ۱۰ فقط برای برای سطر ۱۰ برداشته شه.

```
In [44]: Array_5[:,9]
Out[44]: array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100])
```

جواب تمرین ۱۶: چون اعداد رندم از ۰ تا ۱ خواسته باید با تابع np.random.rand() بنویسیمشون. ۱۵ تا عدد برای هر کدوم خواسته که باید تو پرانتز تابع عدد ۱۵ رو وارد کنیم. بعد گفته هر عضو رو در ۱۰ ضرب کن. نیتجه تابع np.random.rand() اعداد اعضاری هست واسه همین با astype(int) به عدد صحیح تبدیلشون میکنیم و بعد با متد reshape. یه ۳ تا سطر و ۵ تا ستون تغییرشون میدیم.

```
In [45]: Array_A = ((np.random.rand(15)*10).astype(int)).reshape(3,5)

In [46]: Array_B = ((np.random.rand(15)*10).astype(int)).reshape(3,5)

In [47]: Array_A
Out[47]: array([[1, 9, 3, 1, 0],
                [9, 8, 9, 4, 4],
                [9, 2, 4, 1, 3]])

In [48]: Array_B
Out[48]: array([[2, 7, 8, 6, 9],
                [3, 6, 9, 0, 3],
                [5, 4, 6, 7, 3]])
```

جواب تمرین ۱۷: جمع و ضربشون ساده هست ولی در مورد تقسیم چون نتیجه تقسیم بر ۰ خطا میده باید نتیجه تقسیم رو بذاریم تو پرانتز و بعد با متد astype(int). نوعش رو به عدد صحیح تغییر بدیم.

```
In [49]: Array_A + Array_B
Out[49]: array([[ 3, 16, 11,  7,  9],
                [12, 14, 18,  4,  7],
                [14,  6, 10,  8,  6]])

In [50]: Array_A * Array_B
Out[50]: array([[ 2, 63, 24,  6,  0],
                [27, 48, 81,  0, 12],
                [45,  8, 24,  7,  9]])

In [51]: (Array_A * Array_B).astype(int)
Out[51]: array([[ 2, 63, 24,  6,  0],
                [27, 48, 81,  0, 12],
                [45,  8, 24,  7,  9]])
```

این جزوه رو حسابی تمرین کن. تو جزوه بعدی با هم در مورد یه کتابخونه پرکاربرد دیگه به اسم Pandas یاد میگیریم.