آموزش زبان برنامه نویسی پایتون
بخش سوم

مرجع اصلی

http://w3schools.com

محمدجواد فدائی‌اسلام

۱۶ مهر ۱۳۹۹

باسمه تعالی

فهرست مطالب

۱- کتابخانه NumPy

- **مقدمه**

کتابخانه متن‌باز پایتون که نام آن مخفف Numerical Python است و در سال ۲۰۰۵ توسط تراویس اُلیفنت[۱] ایجاد شد. هدف این کتابخانه کار با آرایه‌ها در پایتون است. در این کتابخانه توابعی برای کار در فضای جبرخطی، تبدیل فوریه و ماتریس‌ها ایجاد شده است. در پایتون نوع داده list برای کار با آرایه‌ها وجود دارد اما کند است.

در این کتابخانه نوع داده‌ای برای آرایه ایجاد شده است که ۵۰ بار از لیست سریع‌تر است. این کتابخانه بر خلاف لیست یک فضای پیوسته برای ایجاد آرایه در نظر می‌گیرد. از این‌رو دسترسی و دستکاری عناصر آرایه در آن سریع‌تر است. این موضوع ارجاع محلی در علوم کامپیوتر نام دارد. این موضوع دلیل اصلی سرعت در NumPy است. البته این کتابخانه برای کار با پردازنده‌ها با معماری جدید بهینه‌سازی شده است. این کتابخانه با زبان پایتون نوشته شده است. اما در قسمت‌هایی که سرعت مدنظر بوده است با C++ کدنویسی شده است.

همان‌طور که در گذشته مطرح شد، می‌توان با دستور زیر این کتابخانه را رسم نمود

```
C:\Users\Your Name>pip install numpy
```

و با دستور زیر می‌توان به مجموعه کد وارد نمود:

```
import numpy
```

معمولا این کتابخانه را با نام جدید np به کد وارد می‌کنند:

```
import numpy as np
```

با دستور زیر می‌توان از ویرایش این کتابخانه اطلاع یافت.

```
print(np.__version__)
```

نوع داده آرایه در NumPy به نام ndarray مشخص می‌شود. در دستورات زیر نحوه به کارگیری آن نمایش داده شده است.

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

خروجی کد بالا به صورت زیر است:

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

---

۱ - Travis Oliphant

در کد بالا نوع داده لیست به صورت ndarray تعریف شده است. می‌توان tuple هم به عنوان ورودی (استفاده از پرانتز) باشد.

- **آرایه چند بعدی**

عدد اسکالر یک ارایه صفر بعدی است. در تکه کد زیر از یک آرایه صفر بعدی (عدد اسکالر) تا آرایه سه بعدی تعریف شده است. ndim نام صفتی است که بعد آرایه را یا یک عدد صحیح نشان می‌دهد.

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

در زیر نحوه ایجاد آرایه با تعداد بعد بیشتر نشان داده شده است.

```
import numpy as np
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('number of dimensions :', arr.ndim)
```

خروجی کد بالا به صورت زیر است:

```
[[[[[1 2 3 4]]]]]
number of dimensions : 5
```

در آرایه بالا داخلی‌ترین براکت بعد پنجم است که چهار عضو دارد. اما بعدهای دیگر یک داده دارند.

کد زیر دسترسی به عناصر آرایه یا به عبارتی اندیس‌گذاری آرایه را نشان می‌دهد:

```
import numpy as np
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print(arr[0, 1, 2])
```

خروجی دستور بالا ۶ است. از اندیس‌های منفی هم می‌توان استفاده نمود.

- **برش تکه‌ای از آرایه**

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

کد زیر یک آرایه دوبعدی استخراج می‌کند:

```
import numpy as np
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(arr[0:2, 1:4])
```

خروجی به شکل روبرو است:

```
[[2 3 4]
 [7 8 9]]
```

- **نوع داده در NumPy**

لیست داده‌های مجاز در NumPy به همراه کاراکترهای که نماینده آنها هستند در زیر آمده است.

- i - integer
- b - boolean
- u - unsigned integer
- f - float
- c - complex float
- m - timedelta
- M - datetime
- O - object
- S - string
- U - unicode string
- V - fixed chunk of memory for other type (void)

خصوصیت dtype برای نشان داده نوع داده است.

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

با آرگومان اختیاری dtype می‌توان نوع داده را به هنگام تعریف تعیین نمود.

```
import numpy as np
arr = np.array([1, 2, 3, 4], dtype='S')
print(arr)
print(arr.dtype)
```

با تابع astype می‌توان نوع داده موجود را تغییر داد.

```
arr = np.array([1.1, 2.1, 3.1])
newarr = arr.astype(int)
```

تفاوت copy و view

copy باعث ایجاد داده‌ای جدید می‌شود که در متغیر با نام جدید ذخیره می‌شود و به صورت مستقل عمل می‌کند. اما view یک نام جدید ایجاد می‌کند که تغییرات در مقادیر آن باعث می‌شود در داده‌های اصلی تغییر ایجاد شود.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42
print(arr)
print(x)
```

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42
print(arr)
print(x)
```

- **صفت shape**

این صفت بُعد ماتریس را نشان می‌دهد.

```
import numpy as np
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('shape of array :', arr.shape)
```

خروجی دستورات بالا به صورت زیر است:

```
[[[[[1 2 3 4]]]]]
shape of array : (1, 1, 1, 1, 4)
```

- **تغییر ابعاد ماتریس با reshape**

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

تعداد عناصر در چیدمان‌های متفاوت باید برابر باشد. در غیر این صورت با خطا مواجه می‌شود. اگر یکی از ابعاد را در چیدمان جدید ۱- قرار دهیم. تعیین آن را به پایتون سپرده‌ایم.

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

- **دسترسی به عناصر آرایه با شمارنده**

```
import numpy as np
arr = np.array([1, 2, 3])
for x in arr:
  print(x)
```

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

خروجی کد بالا به صورت زیر است:

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

در کد زیر درایه به درایه به عناصر دسترسی داریم

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

- **پیوستن دو آرایه**

```
import numpy as np
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])
arr = np.concatenate((arr1, arr2), axis=1)
print(arr)
```

- **تفکیک آرایه**

در مثال زیر،آرایه به ۳ قسمت تقسیم می‌شود.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
print(newarr[0])
print(newarr[1])
print(newarr[2])
```

- **جستجو با where**

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 0)
print(x)
```

خروجی کد بالا به شکل زیر است:

```
(array([1, 3, 5, 7]),)
```

- **مرتب‌سازی**

```
import numpy as np
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))
```

- **فیلتر**

```
import numpy as np
arr = np.array([41, 42, 43, 44])
x = [True, False, True, False]
newarr = arr[x]
print(newarr)
```

```
import numpy as np
arr = np.array([41, 42, 43, 44])
# Create an empty list
filter_arr = []
# go through each element in arr
for element in arr:
  # if the element is higher than 42, set the value to True, otherwise False:
  if element > 42:
    filter_arr.append(True)
  else:
    filter_arr.append(False)
newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```

- **عدد تصادفی**

تولید یک عدد تصادفی صحیح بین ۰ تا ۱۰۰

```
from numpy import random
x = random.randint(100)
```

تولید آرایه‌ای ۳ در ۵ از اعداد تصادفی صحیح بین صفر تا صد

```
from numpy import random
x = random.randint(100, size=(3, 5))
print(x)
```

تولید یک عدد تصادفی حقیقی بین ۰ و یک

```
from numpy import random
x = random.rand()
print(x)
```

تولید آرایه‌ای ۳ در ۵ از اعداد تصادفی حقیقی بین صفر تا یک

```
from numpy import random
x = random.rand(3, 5)
print(x)
```

- **انتخاب عدد**

```
from numpy import random
x = random.choice([3, 5, 7, 9])
print(x)
```

```
from numpy import random
x = random.choice([3, 5, 7, 9], size=(3, 5))
print(x)
```

۲- کتابخانه SciPy

کتابخانه محاسبات علمی که از NumPy استفاده می‌کند. مخفف Scientific Python است. و توسط توسعه‌دهنده NumPy ایجاد شده است. این کتابخانه باید توسط PIP نصب شود و برای استفاده به برنامه ضمیمه شود. مواردی که در این کتابخانه به کار گرفته شده است به قرار زیر است:

۱- اعداد ثابت و توابعی برای تبدیل کمیت‌ها به هم

۲- توابع بهینه‌سازی، یافتن ریشه‌های معادله، یافتن کمینه و بیشینه تابع

۳- نمایش داده‌های پراکنده

۴- کار با گراف و پیاده‌سازی الگوریتم‌های یافتن مولفه‌های همبندی[۲] دایکسترا، فلوید، بلمن فرد، پیمایش سطحی و عمقی درخت.

۵- مثلث‌بندی چندضلعی، یافتن پوسته محدب، محاسبه فاصله اقلیدسی، منهتن و همینگ

۶- تبادل اطلاعات در قالب نرم‌افزار متلب

۷- درون‌یابی

و ...

۳- تمرین

۱- عدد تصادفی با توزیع نرمال تولید کنید.

۲- خروجی کدهای روبرو چیست؟

```
import numpy as np
a = np.array([1.0, 2.0, 3.0])
# Example 1
b = 2.0
print(a * b)
# Example 2
c = [2.0, 3.0, 6.0]
print(a * c)
```

```
np.zeros((2,2), dtype=np.int16)
```

```
import numpy as np
A = np.array([[6, 1, 1],
              [4, -2, 5],
              [2, 8, 7]])
print("Rank of A:", np.linalg.matrix_rank(A))
print("\nTrace of A:", np.trace(A))
print("\nDeterminant of A:", np.linalg.det(A))
print("\nInverse of A:\n", np.linalg.inv(A))
print("\nMatrix A raised to power 3:\n", np.linalg.matrix_power(A, 3))
```

```
import numpy as np
# coefficients
a = np.array([[1, 2], [3, 4]])
# constants
b = np.array([8, 18])
print("Solution of linear equations:", np.linalg.solve(a, b))
```

```
import numpy as np
normal_array = np.random.normal(5, 0.5, 10)
```

۲ - connected components

```
print(normal_array)
### Min
print(np.min(normal_array))
### Max
print(np.max(normal_array))
### Mean
print(np.mean(normal_array))
### Median
print(np.median(normal_array))
### Std
print(np.std(normal_array))
```

```
## Linear algebra
### Dot product: product of two arrays
f = np.array([1,2])
g = np.array([4,5])
np.dot(f, g)
```

۳- بر روی گراف زیر الگوریتم دایکسترا، فلوید را اجرا نمایید.