NumPy进阶80题完整版 | 附Notebook版本下载

原创 刘早起 早起Python 4天前

来自专辑

NumPy进阶修炼

本文是NumPy进阶修炼系列的第八篇

- **1 - NumPy进阶修炼 | 入门**
- **2 - NumPy进阶修炼 | 基础**
- **3 - NumPy进阶修炼 | 热身20题**
- **4 - NumPy进阶修炼 | 基础操作与运算**
- **5 - NumPy进阶修炼 | 矩阵操作20题**
- **6 - NumPy进阶修炼 | 41-60题**
- **7 - NumPy进阶修炼 | 最后二十问**
- **8 - NumPy进阶修炼80题完整版**

前言

大家好，NumPy进阶修改80题现在已经全部更新完毕，80道习题涵盖了NumPy中数组**创建、访问、筛选、修改、计算**等常用操作，如果不熟悉NumPy的读者可以刷一遍，因为里面的代码大多**拿走就能用**，所以如果你已经了解NumPy的基本操作，我更建议将这80题**当成速查手册使用**，随用随查！本文共分为两个部分：

- 完整版NumPy80题
- Notebook版下载方式

NumPy进阶修炼80题完整版

01 数据查看

题目：导入并查看NumPy版本

难度：⭐

答案

```
import numpy as np
print(np.__version__)
```

备注：你需要关注你的NumPy版本，部分方法会随着版本更新而变化

02 数据创建

题目：创建十个全为0的一维数组

难度：⭐

期望结果

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

答案

```
np.zeros(10)
```

03 数据创建

题目：创建10个全为0的一维数据并修改数据类型为整数

难度：⭐

答案

```
np.zeros(10,dtype = 'int')
```

04 数据创建

题目：创建20个0-100**固定步长**的数

难度：⭐

期望答案

```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80
       85, 90, 95])
```

答案

```
np.arange(0,100,5)
```

05 数据创建

题目：从list创建数组

难度：⭐

输入

```
List = [1,2,3,4,5,6,7,8,9]
```

答案

```
result = np.array(List)
```

06 数据创建

题目：创建一个三行三列全是1的矩阵

难度：⭐

答案

```
#方法1
np.ones((3,3))
#方法2
np.array([[ 1., 1., 1.],
       [ 1., 1., 1.],
       [ 1., 1., 1.]])
```

07 数据创建

题目：创建一个2行2列矩阵并且元素为布尔类型的True

难度：⭐⭐

期望结果

```
array([[ True, True],
       [ True, True]])
```

答案

```
np.full((2,2), True, dtype=bool)
```

08 数据创建

题目：创建等差数列

备注：从5开始，50结束，共10个数据

难度：⭐

答案

```
np.linspace(start=5,stop=50,num=10)
```

09 数据创建

题目：创建等差数列

备注1：从5开始，50结束，共10个数据，数据类型为int32

难度：⭐⭐

答案

```
np.arange(start = 5, stop = 55, step = 5,dtype = 'int32')
```

备注2:思考与上一题的不同

10 数据创建

题目：创建3x3矩阵

备注：矩阵元素均为0—10之间的随机数

难度：⭐

答案

```
np.random.randint(0,10,(3,3))
```

11 数据创建

题目：创建3x3矩阵

备注：矩阵元素均为服从标准正态分布的随机数

难度：⭐

答案

```
np.random.randn(3, 3)
```

12 数据重塑

题目：将第五题的result修改为3x3矩阵

难度：⭐

答案

```
result = result.reshape(3,3)
```

13 数据修改

题目：对上一题生成的result取转置

难度：⭐

答案

```
result.T
```

14 数据查看

题目：查看result的数据类型

难度：⭐

答案

```
result.dtype
#dtype('int64')
```

15 数据查看

题目：查看result的内存占用

难度：⭐

备注：直接查看或手动计算

答案

```
#方法一：直接查看
result.nbytes
#方法2
手动计算
result.itemsize * 9
#72
```

16 数据创建

题目：将result的数据类型修改为float

难度：⭐

答案

```
result = result.astype(float)
```

17 数据提取

题目：提取result第三行第三列的元素

难度：⭐

答案

```
result[2,2]
```

18 数据修改

题目：将result第三行第三列的元素放大十倍

难度：⭐

答案

```
result[2,2] = result[2,2] * 10
```

19 数据提取

题目：提取result中的所有偶数

难度：⭐⭐

期望输出

```
array([ 2., 4., 6., 8., 90.])
```

答案

```
result[result % 2 == 0]
```

20 数据修改

题目：将result中所有奇数修改为666

难度：⭐⭐

答案

```
result[result % 2 == 1] = 666
```

21 数据创建

题目：创建主对角线都是5的5x5矩阵

难度：⭐

答案

```
result = np.diag([5,5,5,5,5])
```

22 数据修改

题目：交换第一列与第二列

难度：⭐⭐

答案

```
a = result[:, [1,0,2,3,4]]
```

23 数据修改

题目：交换第一行与第二行

难度：⭐⭐

答案

```
result[result % 2 == 1] = 666
```

24 数据查看

题目：判断两个矩阵是否有任何元素不同

难度：⭐⭐

答案

```
print((a == b).all())
```

25 数据计算

题目：计算两个矩阵不同元素的个数

难度：⭐⭐

答案

```
len(np.argwhere(a != b))
```

26 数据查看

题目：找到两个矩阵不同元素的位置

难度：⭐⭐

答案

```
np.argwhere(a != b)
```

27 数据计算

题目：对a和b做矩阵乘法

难度：⭐⭐

答案

```
np.dot(a,b)
```

28 数据计算

题目：计算a和b对应元素相乘

难度：⭐⭐

答案

```
print(np.multiply(a,b))
print('========方法2========')
print(a * b) #方法2
```

29 数据计算

题目：计算行列式(使用21题生成的矩阵)

难度：⭐⭐

答案

```
np.linalg.det(result)
```

30 数据计算

题目：矩阵求逆(使用21题生成的矩阵)

难度：⭐⭐

答案

```
np.linalg.inv(result)
```

31 数据计算

题目：将22与23题生成的np.array对象修改为np.matrix对象

难度：⭐⭐

答案

```
a = np.matrix(a)
b = np.matrix(b)
```

32 数据计算

题目：计算上一题生成的两个np.matrix格式矩阵的对应元素乘积（对比异同）

难度：⭐⭐

答案

```
np.multiply(a,b)
```

33 数据计算

题目：对31题生成的两个np.matrix格式矩阵做矩阵乘法（对比异同）

难度：⭐⭐

答案

```
a * b
```

34 数据计算

题目：将ab两个矩阵按照行拼接

难度：⭐⭐

答案

```
np.hstack((a,b))
```

35 数据计算

题目：将ab两个矩阵按照列拼接

难度：⭐⭐

答案

```
np.vstack((a,b))
```

36 数据计算

题目：思考下面代码运行后new的结果

难度：⭐⭐⭐

答案

```
new = np.pad(result,pad_width = 1,constant_values=1)
```

37 数据查找

题目：找到new中大于1的元素的位置

难度：⭐⭐

答案

```
np.argwhere(new > 1)
```

38 数据修改

题目：将new中大于1的元素修改为8

难度：⭐⭐

答案

```
new[new > 1] = 8
```

39 数据计算

题目：对new矩阵按列求和

难度：⭐⭐

答案

```
np.sum(new, 0)
```

40 数据计算

题目：对new矩阵按行求和

难度：⭐⭐

答案

```
np.sum(new, 1)
```

41 数据创建

题目：生成6行6列的二维数组，值为1-100随机数

难度：⭐

答案

```
data = np.random.randint(1,100, [6,6])
```

42 数据查找

题目：找到每列的最大值

难度：⭐⭐

答案

```
np.amax(data, axis=0)
```

43 数据查找

题目：找到每行的最小值

难度：⭐⭐

答案

```
np.amin(data, axis=1)
```

44 数据计算

题目：计算data每个元素的出现次数

难度：⭐⭐

答案

```
np.unique(data,return_counts=True)
```

45 数据计算

题目：计算data每行元素大小排名

难度：⭐⭐

答案

```
data.argsort()
```

46 数据处理

题目：将data按行重复一次

难度：⭐⭐

答案

```
np.repeat(data, 2, axis=0)
```

47 数据处理

题目：去除data的重复行

难度：⭐⭐

答案

```
np.unique(data,axis = 0)
```

48 数据抽样

题目：从data的第一行中不放回抽3个元素

难度：⭐⭐

答案

```
np.random.choice(data[0:1][0], 3, replace=False)
```

49 数据计算

题目：计算data第二行中不含第三行的元素的元素

难度：⭐⭐

答案

```
a = data[1:2]
b = data[2:3]
index=np.isin(a,b)
array=a[~index]
array
```

50 数据计算

题目：判断data是否有空行

难度：⭐⭐

答案

```
(~data.any(axis=1)).any()
```

51 数据排序

题目：将data的每行升序排列

难度：⭐⭐

答案

```
data.sort(axis = 1)
```

52 数据转换

题目：将data的数据格式转换为float

难度：⭐⭐

答案

```
data1 = data.astype(float)
```

思考：为什么不能在data本身转换

53 数据修改

题目：将data1小于5的元素修改为nan

难度：⭐⭐

答案

```
data1[data1 < 5] = np.nan
```

54 数据处理

题目：删除data1含有空值的行

难度：⭐⭐

答案

```
data1 = data1[~np.isnan(data1).any(axis=1), :]
```

55 数据计算

题目：计算data1第一行出现频率最高的值

难度：⭐⭐⭐

答案

```
vals, counts = np.unique(data1[0,:], return_counts=True)
print(vals[np.argmax(counts)])
```

56 数据计算

题目：计算data1中与100最接近的元素

难度：⭐⭐⭐

答案

```
a = 100
data1.flat[np.abs(data1 - a).argmin()]
```

57 数据计算

题目：计算data1每一行的元素减去每一行的平均值

难度：⭐⭐

答案

```
data1 - data1.mean(axis=1, keepdims=True)
```

58 数据计算

题目：将data1归一化至区间[0,1]

难度：⭐⭐

答案

```
a = np.max(data1) - np.min(data1)
(data1 - np.min(data1)) / a
```

59 数据计算

题目：将data1标准化

难度：⭐⭐⭐

答案

```
mu = np.mean(data1, axis=0)
sigma = np.std(data1, axis=0)
```

```
(data1 - mu) / sigma
```

60 数据存储

题目：将data1存储至本地

难度：⭐

答案

```
np.savetxt('test.txt',data1)
```

61 数据查找

问：如何获得两个数组之间的相同元素

输入：

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
arr1 = np.random.randint(10,6,6)
arr2 = np.random.randint(10,6,6)
```

答案：

```
arr1 = np.random.randint(10,6,6)
arr2 = np.random.randint(10,6,6)
print("arr1: %s"%arr1)
print("arr2: %s"%arr2)
np.intersect1d(arr1,arr2)
```

```
print("arr1: %s"%arr1)
print("arr2: %s"%arr2)
np.intersect1d(arr1,arr2)
```

```
arr1: [9 4 1 4 9 9 8 5 3 1]
arr2: [1 5 8 7 9 2 6 9 1 2]

array([1, 5, 8, 9])
```

62 数据修改

问：如何从一个数组中删除另一个数组存在的元素

输入：

```
arr1 = np.random.randint(10,6,6)
arr2 = np.random.randint(10,6,6)
```

答案：

```
arr1 = np.random.randint(1,10,10)
arr2 = np.random.randint(1,10,10)
print("arr1: %s"%arr1)
print("arr2: %s"%arr2)
np.setdiff1d(arr1,arr2)
```

```
arr1 = np.random.randint(1,10,10)
arr2 = np.random.randint(1,10,10)
print("arr1: %s"%arr1)
print("arr2: %s"%arr2)
np.setdiff1d(arr1,arr2)
```

```
arr1: [3 3 2 6 6 1 4 6 2 2]
arr2: [3 3 2 6 8 8 5 2 4 7]

array([1])
```

63 数据修改

问：如何修改一个数组为只读模式

输入：

```
arr1 = np.random.randint(1,10,10)
```

答案：

```
arr1 = np.random.randint(1,10,10)
arr1.flags.writeable = False
```

```
arr1 = np.random.randint(1,10,10)
arr1.flags.writeable = False
```

```
#尝试修改会报错!
arr1[0] = 6
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-31-ddcf305e5efb> in <module>
      1 #尝试修改会报错!
----> 2 arr1[0] = 6

ValueError: assignment destination is read-only
```

64 数据转换

问：如何将list转为numpy数组

输入：

```
a = [1,2,3,4,5]
```

答案：

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

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

```
array([1, 2, 3, 4, 5])
```

65 数据转换

问：如何将pd.DataFrame转为numpy数组

输入：

```
df = pd.DataFrame({'A':[1,2,3],'B':[4,5,6],'C':[7,8,9]})
```

答案：

```
df.values
```

```
df = pd.DataFrame({'A':[1,2,3],'B':[4,5,6],'C':[7,8,9]})
print(df)
print(df.values)
```

```
   A  B  C
0  1  4  7
1  2  5  8
2  3  6  9
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

66 数据分析

问：如何使用numpy进行描述性统计分析

输入：

```
arr1 = np.random.randint(1,10,10)
arr2 = np.random.randint(1,10,10)
```

答案：

```
arr1 = np.random.randint(1,10,10)
arr2 = np.random.randint(1,10,10)

print("arr1的平均数为:%s" %np.mean(arr1))
print("arr1的中位数为:%s" %np.median(arr1))
print("arr1的方差为:%s" %np.var(arr1))
print("arr1的标准差为:%s" %np.std(arr1))
print("arr1,arr的相关性矩阵为:%s" %np.cov(arr1,arr2))
print("arr1,arr的协方差矩阵为:%s" %np.corrcoef(arr1,arr2))
```

```
arr1的平均数为:5.2
arr1的中位数为:5.0
arr1的方差为:6.56
arr1的标准差为:2.561249649731396
arr1,arr的相关性矩阵为:[[7.28888889 2.6        ]
 [2.6        7.12222222]]
arr1,arr的协方差矩阵为:[[1.         0.36085682]
 [0.36085682 1.        ]]
```

67 数据抽样

问：如何使用numpy进行概率抽样

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

输入：

```
arr = np.array([1,2,3,4,5])
np.random.choice(arr,10,p = [0.1,0.1,0.1,0.1,0.6])
```

答案：

```
arr = np.array([1,2,3,4,5])
np.random.choice(arr,10,p = [0.1,0.1,0.1,0.1,0.6])
```

```
array([2, 5, 5, 5, 5, 4, 1, 5, 5, 5])
```

68 数据创建

问：如何为数据创建副本

输入：

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

答案：

```
#对副本数据进行修改，不会影响到原始数据
arr = np.array([1,2,3,4,5])
arr1 = arr.copy()
```

69 数据切片

问：如何对数组进行切片

输入：

```
arr = np.arange(10)
```

备注：从索引2开始到索引8停止，间隔为2

答案：

```
arr = np.arange(10)
a = slice(2,8,2)
arr[a] #等价于arr[2:8:2]
```

70 字符串操作

问：如何使用NumPy操作字符串

输入：

```
str1 = ['I love']
str2 = [' Python']
```

答案：

```
#拼接字符串
str1 = ['I love']
str2 = [' Python']
print(np.char.add(str1,str2))

#大写首字母
str3 = np.char.add(str1,str2)
print(np.char.title(str3))
```

71 数据修改

问：如何对数据向上/下取整

输入：

```
arr = np.random.uniform(0,10,10)
```

答案：

```
arr = np.random.uniform(0,10,10)
print(arr)
###向上取整
print(np.ceil(arr))
###向下取整
print(np.floor(arr) )
```

```
arr = np.random.uniform(0,10,10)
print(arr)
###向上取整
print(np.ceil(arr))
###向下取整
print(np.floor(arr) )
```

```
[3.69853424 2.37607144 2.75296734 3.98645188 9.8325098  9.78061437
 1.05052842 6.93954312 3.74869061 0.89292687]
[ 4.  3.  3.  4. 10. 10.  2.  7.  4.  1.]
[3. 2. 2. 3. 9. 9. 1. 6. 3. 0.]
```

72 格式修改

问：如何取消默认科学计数显示数据

答案：

```
np.set_printoptions(suppress=True)
```

73 数据修改

问：如何使用NumPy对二维数组逆序

输入：

```
arr = np.random.randint(1,10,[3,3])
```

答案：

```
arr = np.random.randint(1,10,[3,3])
print(arr)
print('列逆序')
print(arr[:, -1::-1])
print('行逆序')
print(arr[-1::-1, :])
```

```
[[5 6 2]
 [1 6 9]
 [2 7 5]]
列逆序
[[2 6 5]
 [9 6 1]
 [5 7 2]]
行逆序
[[2 7 5]
 [1 6 9]
 [5 6 2]]
```

74 数据查找

问：如何使用NumPy根据位置查找元素

输入：

```
arr1 = np.random.randint(1,10,5)
arr2 = np.random.randint(1,20,10)
```

备注：在arr2中根据arr1中元素以位置查找

答案：

```
arr1 = np.random.randint(1,10,5)
arr2 = np.random.randint(1,20,10)
print(arr1)
print(arr2)
print(np.take(arr2,arr1))
```

```
arr1 = np.random.randint(1,10,5)
arr2 = np.random.randint(1,20,10)
print(arr1)
print(arr2)
print(np.take(arr2,arr1))
```

```
[4 9 3 7 9]
[ 1  3 14  5 13 15 16  6  1  6]
[13  6  5  6  6]
```

75 数据计算

问：如何使用numpy求余数

输入：

```
a = 10
b = 3
```

答案：

```
np.mod(a,b)
```

76 数据计算

问：如何使用NumPy进行矩阵SVD分解

输入：

```
A = np.random.randint(1,10,[3,3])
```

答案：

```
np.linalg.svd(A)
```

```
np.linalg.svd(A)
```

```
(array([[-0.53080534, -0.05856788, -0.84546762],
        [-0.50084903, -0.7830816 ,  0.36869155],
        [-0.68366361,  0.61915508,  0.38633023]]),
 array([15.54841031,  7.33950585,  1.54226802]),
 array([[-0.54854452, -0.50684335, -0.66498777],
        [-0.83142704,  0.24649305,  0.49796612],
        [-0.08847596,  0.82604539, -0.55661568]]))
```

77 数据筛选

问：如何使用NumPy多条件筛选数据

输入：

```
arr = np.random.randint(1,20,10)
```

答案：

```
arr = np.random.randint(1,20,10)
print(arr[(arr>1)&(arr<7)&(arr%2==0)])
```

78 数据修改

问：如何使用NumPy对数组分类

备注：将大于等于7，或小于3的元素标记为1，其余为0

输入：

```
arr = np.random.randint(1,20,10)
```

答案：

```
arr = np.random.randint(1,20,10)
print(arr)
print(np.piecewise(arr, [arr < 3, arr >= 7], [-1, 1]))
```

```
arr = np.random.randint(1,20,10)
print(arr)
print(np.piecewise(arr, [arr < 3, arr >= 7], [-1, 1]))
```

```
[ 8 19 16  1  4  7 12  1 16  1]
[ 1  1  1 -1  0  1  1 -1  1 -1]
```

79 数据修改

问：如何使用NumPy压缩矩阵

备注：从数组的形状中删除单维度条目，即把shape中为1的维度去掉

输入：

```
arr = np.random.randint(1,10,[3,1])
```

答案：

```
arr = np.random.randint(1,10,[3,1])
print(arr)
print(np.squeeze(arr))
```

```
arr = np.random.randint(1,10,[3,1])
print(arr)
print(np.squeeze(arr))
```

```
[[3]
 [8]
 [5]]
[3 8 5]
```

80 数据计算

问：如何使用numpy求解线性方程组

输入：

```
A = np.array([[1, 2, 3], [2, -1, 1], [3, 0, -1]])
b = np.array([9, 8, 3])
```

备注：求解Ax=b

答案：

```
A = np.array([[1, 2, 3], [2, -1, 1], [3, 0, -1]])
b = np.array([9, 8, 3])
x = np.linalg.solve(A, b)
print(x)
```

```
A = np.array([[1, 2, 3], [2, -1, 1], [3, 0, -1]])
b = np.array([9, 8, 3])
x = np.linalg.solve(A, b)
print(x)
```

```
[ 2. -1.  3.]
```

下载方式

为了让各位读者更方便的刷题，我已经将NumPy80题整理在Notebook中，共分为两个版本，一份**无答案版**可以用来刷题 👇

第一期｜热身20题:数据创建与访问

1.导入并查看NumPy版本

2.创建十个全为0的一维数组

3.创建10个全为0的一维数据并修改数据类型为整数

4.创建20个0-100固定步长的数

5.从list创建数组

NumPy进阶修炼80题 第二期 ｜基本矩阵操作与运算

In [ ]:

21.创建主对角线都是5的5x5矩阵

In [ ]:

22.交换第一列与第二列

In [ ]:

23.交换第一行与第二行

In [ ]:

24.判断两个矩阵是否有任何元素不同(使用22,23两题得到的矩阵)

In [ ]:

25.计算两个矩阵不同元素的个数(使用22,23两题得到的矩阵)

In [ ]:

26.找到两个矩阵不同元素的位置(使用22,23两题得到的矩阵)

In [ ]:

一份**有答案版本**用来参考学习 👇

第一期｜热身20题:数据创建与访问

1.导入并查看NumPy版本

In [2]:
```
import numpy as np
print(np.__version__)
```

```
1.15.4
```

2.创建十个全为0的一维数组

In [46]:
```
np.zeros(10)
```

Out[46]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

3.创建10个全为0的一维数据并修改数据类型为整数

In [9]:
```
np.zeros(10,dtype = 'int')
```

Out[9]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

4.创建20个0-100固定步长的数

In [10]:
```
np.arange(0,100,5)
```

Out[10]:
```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80,
       85, 90, 95])
```

54 删除data1含有nan的行

```
data1 = data1[~np.isnan(data1).any(axis=1), :]
```

```
data1
```

```
array([[33., 51., 56., 56., 56., 73.],
       [29., 44., 53., 63., 71., 73.],
       [25., 30., 44., 51., 87., 90.],
       [ 7., 29., 39., 50., 83., 89.],
       [ 8., 18., 33., 40., 64., 75.]])
```

55 找出data1第一行出现频率最高的值

```
vals, counts = np.unique(data1[0,:], return_counts=True)
print(vals[np.argmax(counts)])
```

```
56.0
```

56 找到data1中与100最接近的数字

```
a = 100
data1.flat[np.abs(data1 - a).argmin()]
```

```
90.0
```

57 data1每一行的元素减去每一行的平均值

```
data1 - data1.mean(axis=1, keepdims=True)
```

```
array([[-21.16666667,  -3.16666667,   1.83333333,   1.83333333,
          1.83333333,  18.83333333],
```

在后台回复**numpy**即可下载完整的NumPy80题，因为我每题仅给出一种解法，因此在刷题过程中也应该思考**是否有不一样/更高效的方法**，结合**Pandas120题**使用更是威力无穷！另外Python数据可视化同款专题正在准备中，敬请期待！

点分享

点点赞 点在看