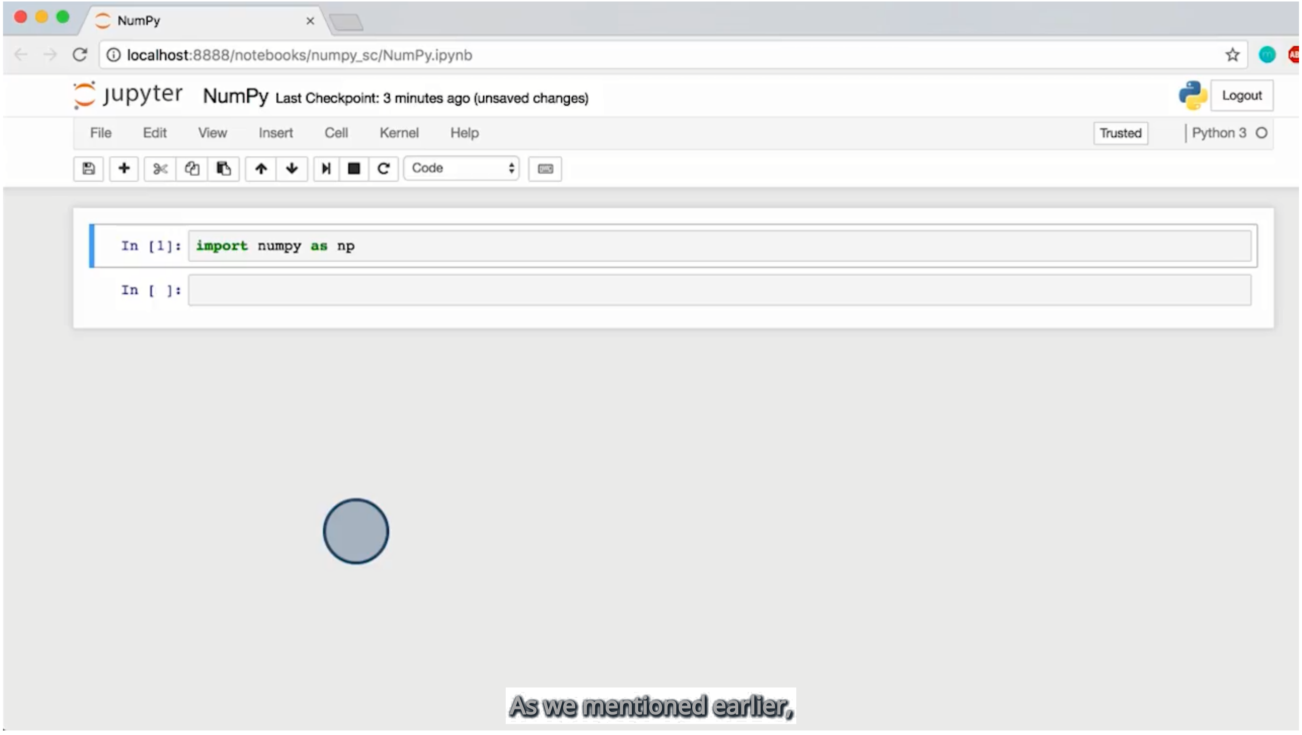
ndarray 切片



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

As we mentioned earlier,

00:00 / 06:23 1x CC

正如之前提到的，我们除了能够一次访问一个元素之外，NumPy 还提供了访问 ndarray 子集的方式，称之为*切片*。切片方式是在方括号里用冒号 `:` 分隔起始和结束索引。通常，你将遇到三种类型的切片：

```
1. ndarray[start:end]
2. ndarray[start:]
3. ndarray[:end]
```

第一种方法用于选择在 `start` 和 `end` 索引之间的元素。第二种方法用于选择从 `start` 索引开始直到*最后一个*索引的所有元素。第三种方法用于选择从*第一个*索引开始直到 `end` 索引的所有元素。请注意，在第一种方法和第三种方法中，结束索引*不包括*在内。此外注意，因为 ndarray 可以是多维数组，在进行切片时，通常需要为数组的每个维度指定一个切片。

现在我们将查看一些示例，了解如何使用上述方法从秩为 2 的 ndarray 中选择不同的子集。

```python
# We print X
print()
print('X = \n', X)
print()

# We select all the elements that are in the 2nd through 4th rows and in the 3rd to 5th columns
Z = X[1:4,2:5]

# We print Z
print('Z = \n', Z)

# We can select the same elements as above using method 2
W = X[1:,2:5]

# We print W
print()
print('W = \n', W)

# We select all the elements that are in the 1st through 3rd rows and in the 3rd to 5th columns
Y = X[:3,2:5]

# We print Y
print()
print('Y = \n', Y)

# We select all the elements in the 3rd row
v = X[2,:]

# We print v
print()
print('v = ', v)

# We select all the elements in the 3rd column
q = X[:,2]

# We print q
print()
print('q = ', q)

# We select all the elements in the 3rd column but return a rank 2 ndarray
R = X[:,2:3]

# We print R
print()
print('R = \n', R)
```

```
X =
[[ 0 1 2 3 4]
```

```
 [15 16 17 18 19]]
```

```
Z =
[[ 7 8 9]
 [12 13 14]
 [17 18 19]]
```

```
W =
[[ 7 8 9]
 [12 13 14]
 [17 18 19]]
```

```
Y =
[[ 2 3 4]
 [ 7 8 9]
 [12 13 14]]
```

```
v = [10 11 12 13 14]
```

```
q = [ 2 7 12 17]
```

```
R =
[[ 2]
 [ 7]
 [12]
 [17]]
```

注意，当我们选择第 3 列中的所有元素，即上述变量 `q` ，切片返回一个秩为 1 的 ndarray，而不是秩为 2 的 ndarray。但是，如果以稍微不同的方式切片 `X` ，即上述变量 `R` ，实际上可以获得秩为 2 的 ndarray。

请务必注意，如果对 ndarray 进行切片并将结果保存到新的变量中，就像之前一样，数据不会复制到新的变量中。初学者对于这一点经常比较困惑。因此，我们将深入讲解这方面的知识。

```
Z = X[1:4,2:5]
```

原始数组 `X` 的切片没有复制到变量 `Z` 中。 `X` 和 `Z` 现在只是*同*一个 ndarray 的两个不同名称。我们提到，切片只是创建了原始数组的一个*视图*。也就是说，如果对 `Z` 做出更改，也会更改 `X` 中的元素。我们来看一个示例：

```
# We create a 4 x 5 ndarray that contains integers from 0 to 19
X = np.arange(20).reshape(4, 5)

# We print X
print()
print('X = \n', X)
print()

# We select all the elements that are in the 2nd through 4th rows and in the 3rd to 4th columns
Z = X[1:4,2:5]

# We print Z
print()
print('Z = \n', Z)
print()

# We change the last element in Z to 555
Z[2,2] = 555

# We print X
print()
print('X = \n', X)
print()
```

```
X =
[[ 0 1 2 3 4]
 [ 5 6 7 8 9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

```
Z =
[[ 7 8 9]
 [12 13 14]
 [17 18 19]]
```

```
[ 5 6 7 8 9]
[ 10 11 12 13 14]
[ 15 16 17 18 555]]
```

可以从上述示例中清晰地看出，如果对 `Z` 做出更改， `X` 也会更改。

但是，如果我们想创建一个新的 ndarray，其中包含切片中的值的副本，需要使用 `np.copy()` 函数。 `np.copy(ndarray)` 函数会创建给定 `ndarray` 的一个副本。此函数还可以当做方法使用，就像之前使用 reshape 函数一样。我们来看看之前的相同示例，但是现在创建数组副本。我们将 `copy` 同时当做函数和方法。

```python
# We create a 4 x 5 ndarray that contains integers from 0 to 19
X = np.arange(20).reshape(4, 5)

# We print X
print()
print('X = \n', X)
print()

# create a copy of the slice using the np.copy() function
Z = np.copy(X[1:4,2:5])

#  create a copy of the slice using the copy as a method
W = X[1:4,2:5].copy()

# We change the last element in Z to 555
Z[2,2] = 555

# We change the last element in W to 444
W[2,2] = 444

# We print X
print()
print('X = \n', X)

# We print Z
print()
print('Z = \n', Z)

# We print W
print()
print('W = \n', W)
```

```
X =
[[ 0 1 2 3 4]
```

```
 [15 16 17 18 19]]
```

```
X =
[[ 0 1 2 3 4]
 [ 5 6 7 8 9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

```
Z =
[[ 7 8 9]
 [ 12 13 14]
 [ 17 18 555]]
```

```
W =
[[ 7 8 9]
 [ 12 13 14]
 [ 17 18 444]]
```

可以清晰地看出，通过使用 `copy` 命令，我们创建了完全相互独立的新 ndarray。

通常，我们会使用一个 ndarray 对另一个 ndarray 进行切片、选择或更改另一个 ndarray 的元素。我们来看一些示例：

```
# We create a rank 1 ndarray that will serve as indices to select elements f
rom X
indices = np.array([1,3])

# We print X
print()
print('X = \n', X)
print()

# We print indices
print('indices = ', indices)
print()

# We use the indices ndarray to select the 2nd and 4th row of X
Y = X[indices,:]

# We use the indices ndarray to select the 2nd and 4th column of X
Z = X[:, indices]

# We print Y
print()
print('Y = \n', Y)

# We print Z
print()
print('Z = \n', Z)
```

```
X =
[[ 0 1 2 3 4]
 [ 5 6 7 8 9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

```
indices = [1 3]
```

```
Y =
[[ 5 6 7 8 9]
 [15 16 17 18 19]]
```

```
Z =
[[ 1 3]
 [ 6 8]
```

NumPy 还提供了从 ndarray 中选择特定元素的内置函数。例如，`np.diag(ndarray, k=N)` 函数会以 `N` 定义的 对角线 提取元素。默认情况下，`k=0`，表示主对角线。`k > 0` 的值用于选择在主对角线之上的对角线中的元素，`k < 0` 的值用于选择在主对角线之下的对角线中的元素。我们来看一个示例：

```
# We create a 4 x 5 ndarray that contains integers from 0 to 19
X = np.arange(25).reshape(5, 5)

# We print X
print()
print('X = \n', X)
print()

# We print the elements in the main diagonal of X
print('z =', np.diag(X))
print()

# We print the elements above the main diagonal of X
print('y =', np.diag(X, k=1))
print()

# We print the elements below the main diagonal of X
print('w = ', np.diag(X, k=-1))
```

```
X =
[[ 0 1 2 3 4]
 [ 5 6 7 8 9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

```
z = [ 0 6 12 18 24]
```

```
y = [ 1 7 13 19]
```

```
w = [ 5 11 17 23]
```

素，如以下示例所示：

```
# Create 3 x 3 ndarray with repeated values
X = np.array([[1,2,3],[5,2,8],[1,2,3]])

# We print X
print()
print('X = \n', X)
print()

# We print the unique elements of X
print('The unique elements in X are:',np.unique(X))
```

```
X =
[[1 2 3]
 [5 2 8]
 [1 2 3]]
```

```
The unique elements in X are: [1 2 3 5 8]
```

下一项