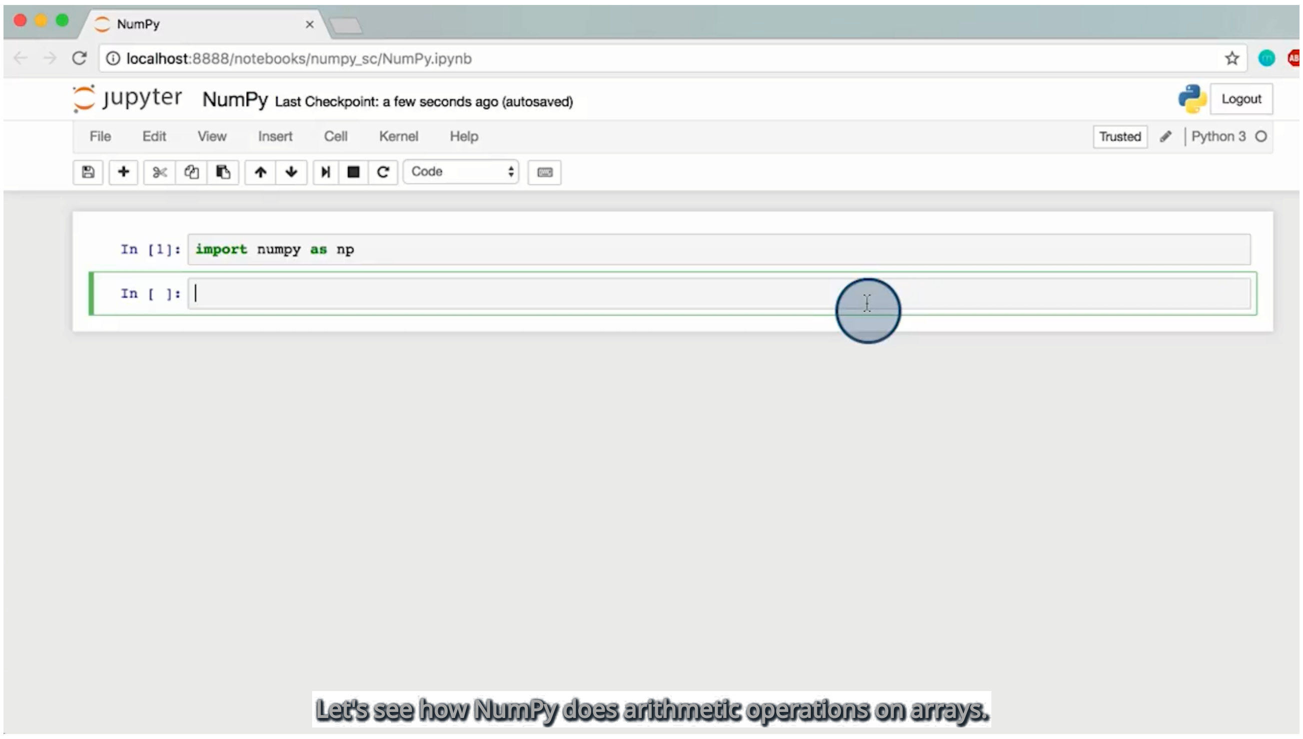
算术运算和广播


```
In [1]: import numpy as np
In [ ]:
```
Let's see how NumPy does arithmetic operations on arrays.

我们已经学到“ NumPy ”课程的最后一节课了。在最后一节课，我们将了解 NumPy 如何对 ndarray 进行算术运算。NumPy 允许对 ndarray 执行元素级运算以及矩阵运算。在这节课，我们将仅了解如何对 ndarray 进行元素级运算。为了进行元素级运算，NumPy 有时候会用到*广播*功能。广播一词用于描述 NumPy 如何对具有不同形状的 ndarray 进行元素级算术运算。例如，在标量和 ndarray 之间进行算术运算时，会隐式地用到广播。

我们先在 ndarray 之间进行元素级加减乘除运算。为此，我们可以在 NumPy 中使用 `np.add()` 等函数，或者使用 `+` 等算术符号，后者与数学方程式的写法更像。这两种形式都执行相同的运算，唯一的区别是如果采用函数方式，函数通常都具有各种选项，可以通过关键字调整这些选项。请注意，在进行元素级运算时，对其执行运算的 ndarray 必须具有相同的形状或者可以广播。我们将在这节课的稍后阶段详细讲解这方面的知识。我们先对秩为 1 的 ndarray 执行元素级算术运算：

```
y = np.array([5.5,6.5,7.5,8.5])

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

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

# We perfrom basic element-wise operations using arithmetic symbols and funct
ions
print('x + y = ', x + y)
print('add(x,y) = ', np.add(x,y))
print()
print('x - y = ', x - y)
print('subtract(x,y) = ', np.subtract(x,y))
print()
print('x * y = ', x * y)
print('multiply(x,y) = ', np.multiply(x,y))
print()
print('x / y = ', x / y)
print('divide(x,y) = ', np.divide(x,y))
```

```
x = [1 2 3 4]
```

```
y = [ 5.5 6.5 7.5 8.5]
```

```
x + y = [ 6.5 8.5 10.5 12.5]
add(x,y) = [ 6.5 8.5 10.5 12.5]
```

```
x - y = [-4.5 -4.5 -4.5 -4.5]
subtract(x,y) = [-4.5 -4.5 -4.5 -4.5]
```

```
x * y = [ 5.5 13. 22.5 34. ]
multiply(x,y) = [ 5.5 13. 22.5 34. ]
```

```
x / y = [ 0.18181818 0.30769231 0.4 0.47058824]
divide(x,y) = [ 0.18181818 0.30769231 0.4 0.47058824]
```

```
# We create two rank 2 ndarrays
X = np.array([1,2,3,4]).reshape(2,2)
Y = np.array([5.5,6.5,7.5,8.5]).reshape(2,2)

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

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

# We perform basic element-wise operations using arithmetic symbols and funct
ions
print('X + Y = \n', X + Y)
print()
print('add(X,Y) = \n', np.add(X,Y))
print()
print('X - Y = \n', X - Y)
print()
print('subtract(X,Y) = \n', np.subtract(X,Y))
print()
print('X * Y = \n', X * Y)
print()
print('multiply(X,Y) = \n', np.multiply(X,Y))
print()
print('X / Y = \n', X / Y)
print()
print('divide(X,Y) = \n', np.divide(X,Y))
```

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

```
Y =
[[ 5.5 6.5]
 [ 7.5 8.5]]
```

```
X + Y =
[[ 6.5 8.5]
 [ 10.5 12.5]]
```

> [10.5 12.5]]

> X - Y =
> [[-4.5 -4.5]
> [-4.5 -4.5]]

> subtract(X,Y) =
> [[-4.5 -4.5]
> [-4.5 -4.5]]

> X * Y =
> [[5.5 13.]
> [22.5 34.]]

> multiply(X,Y) =
> [[5.5 13.]
> [22.5 34.]]

> X / Y =
> [[0.18181818 0.30769231]
> [0.4 0.47058824]]

> divide(X,Y) =
> [[0.18181818 0.30769231]
> [0.4 0.47058824]]

我们还可以同时对 ndarray 的所有元素应用数学函数，例如 `sqrt(x)`。

```
# We print x
print()
print('x = ', x)

# We apply different mathematical functions to all elements of x
print()
print('EXP(x) =', np.exp(x))
print()
print('SQRT(x) =',np.sqrt(x))
print()
print('POW(x,2) =',np.power(x,2)) # We raise all elements to the power of 2
```

> x = [1 2 3 4]

> EXP(x) = [2.71828183 7.3890561 20.08553692 54.59815003]

> SQRT(x) = [1. 1.41421356 1.73205081 2.]

> POW(x,2) = [1 4 9 16]

NumPy 的另一个重要特性是具有大量不同的统计学函数。统计学函数为我们提供了关于 ndarray 中元素的统计学信息。我们来看一些示例：

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

print('Average of all elements in X:', X.mean())
print('Average of all elements in the columns of X:', X.mean(axis=0))
print('Average of all elements in the rows of X:', X.mean(axis=1))
print()
print('Sum of all elements in X:', X.sum())
print('Sum of all elements in the columns of X:', X.sum(axis=0))
print('Sum of all elements in the rows of X:', X.sum(axis=1))
print()
print('Standard Deviation of all elements in X:', X.std())
print('Standard Deviation of all elements in the columns of X:', X.std(axis=0
))
print('Standard Deviation of all elements in the rows of X:', X.std(axis=1))
print()
print('Median of all elements in X:', np.median(X))
print('Median of all elements in the columns of X:', np.median(X,axis=0))
print('Median of all elements in the rows of X:', np.median(X,axis=1))
print()
print('Maximum value of all elements in X:', X.max())
print('Maximum value of all elements in the columns of X:', X.max(axis=0))
print('Maximum value of all elements in the rows of X:', X.max(axis=1))
print()
print('Maximum value of all elements in X:', X.min())
print('Maximum value of all elements in the columns of X:', X.min(axis=0))
print('Maximum value of all elements in the rows of X:', X.min(axis=1))
```

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

```
Average of all elements in X: 2.5
Average of all elements in the columns of X: [ 2. 3.]
Average of all elements in the rows of X: [ 1.5 3.5]
```

```
Sum of all elements in X: 10
Sum of all elements in the columns of X: [4 6]
Sum of all elements in the rows of X: [3 7]
```

Standard Deviation of all elements in the rows of X: [0.5 0.5]

Median of all elements in X: 2.5
Median of all elements in the columns of X: [2. 3.]
Median of all elements in the rows of X: [1.5 3.5]

Maximum value of all elements in X: 4
Maximum value of all elements in the columns of X: [3 4]
Maximum value of all elements in the rows of X: [2 4]

Maximum value of all elements in X: 1
Maximum value of all elements in the columns of X: [1 2]
Maximum value of all elements in the rows of X: [1 3]

最后，我们来看看 NumPy 如何使 ndarray 中的所有元素与单个数字相加，而不使用复杂的循环。

```
# We create a 2 x 2 ndarray
X = np.array([[1,2], [3,4]])

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

print('3 * X = \n', 3 * X)
print()
print('3 + X = \n', 3 + X)
print()
print('X - 3 = \n', X - 3)
print()
print('X / 3 = \n', X / 3)
```

```
X =
[[1 2]
 [3 4]]
 [ 9 12]]
```

```
3 + X =
[[4 5]
 [6 7]]
```

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

```
X / 3 =
[[ 0.33333333 0.66666667]
 [ 1. 1.33333333]]
```

在上述示例中，NumPy 在后台对 ndarray 广播 `3` ，使它们具有相同的形状。这样我们仅使用一行代码，就可以使 `X` 的每个元素加 3。

Numpy 可以对两个形状不同的 ndarray 执行相同的操作，但是存在一些限制，如下所示。

```python
# We create a 3 x 3 ndarray
Y = np.array([[1,2,3],[4,5,6],[7,8,9]])

# We create a 3 x 1 ndarray
Z = np.array([1,2,3]).reshape(3,1)

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

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

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

print('x + Y = \n', x + Y)
print()
print('Z + Y = \n',Z + Y)
```

```
x = [1 2 3]
```

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

```
Z =
[[1]
 [2]
 [3]]
```

```
x + Y =
[[ 2 4 6]
 [ 5 7 9]
 [ 8 10 12]]
 [ 6 7 8]
 [10 11 12]]
```

和之前一样，NumPy 能够通过沿着大的 ndarray 对更小的 ndarray 进行广播，将 1 x 3 和 3 x 1 ndarray 加到 3 x 3 ndarray 上。通常，NumPy 能够这么操作的前提是，更小的 ndarray（例如我们的示例中的 1 x 3 ndarray）可以扩展成更大的 ndarray 的形状，并且生成的广播很清晰明确。

确保阅读 NumPy 文档，详细了解广播及其规则： Broadcasting

下一项