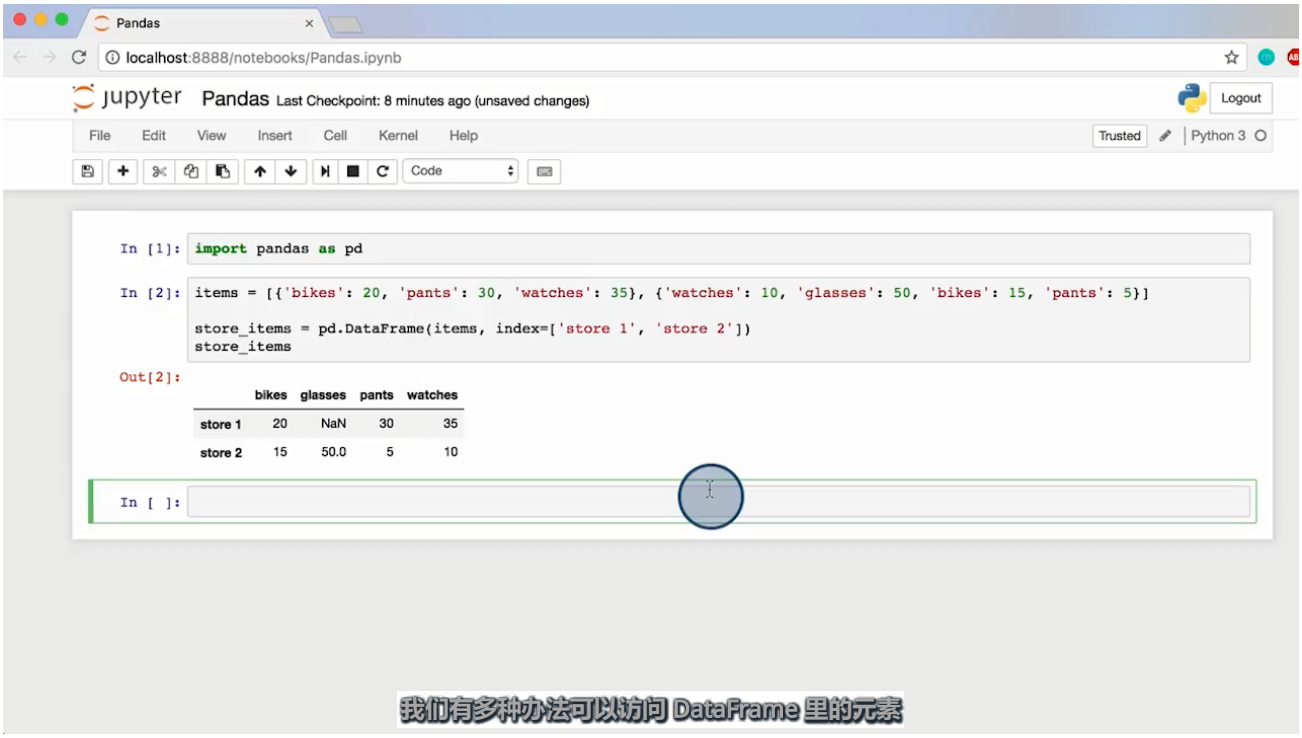
访问 Pandas DataFrame 中的元素



00:00 / 04:29 1x CC

我们可以通过多种不同的方式访问 Pandas DataFrame 中的元素。通常，我们可以使用行和列标签访问 DataFrame 的行、列或单个元素。我们将使用在上节课创建的同一 `store_items` DataFrame。我们来看一些示例：

```
# We print the store_items DataFrame
print(store_items)

# We access rows, columns and elements using labels
print()
print('How many bikes are in each store:\n', store_items[['bikes']
print()
print('How many bikes and pants are in each store:\n', store_items
print()
print('What items are in Store 1:\n', store_items.loc[['store 1']]
```

	bikes	glasses	pants	watches
store 1	20	NaN	30	35
store 2	15	50.0	5	10

How many bikes are in each store:

	bikes
store 1	20
store 2	15

How many bikes and pants are in each store:

	bikes	pants
store 1	20	30
store 2	15	5

What items are in Store 1:

	bikes	glasses	pants	watches
store 1	20	NaN	30	35

How many bikes are in Store 2: 15

数量时，我们首先使用列标签 **bikes**，然后使用行标签 **store 2**。如果先提供行标签，将出错。

我们还可以通过添加行或列修改 DataFrame。我们先了解如何向 DataFrame 中添加新的列。假设我们想添加每个商店的**衬衫**库存。为此，我们需要向 `store_items` DataFrame 添加一个新列，表示每个商店的衬衫库存。我们来编写代码：

```
# We add a new column named shirts to our store_items DataFrame i
# will put 15 shirts in store 1 and 2 shirts in store 2
store_items['shirts'] = [15,2]

# We display the modified DataFrame
store_items
```

	bikes	glasses	pants	watches	shirts
store 1	20	NaN	30	35	15
store 2	15	50.0	5	10	2

可以看出，当我们添加新的列时，新列添加到了 DataFrame 的末尾。

还可以使用算术运算符向 DataFrame 中的其他列之间添加新列。我们来看一个示例：

```
# We make a new column called suits by adding the number of shirt
store_items['suits'] = store_items['pants'] + store_items['shirts

# We display the modified DataFrame
store_items
```

	bikes	glasses	pants	watches	shirts	suits
store 1	20	NaN	30	35	15	45

store 2	15	50.0	5	10	2	7

假设现在你开了一家新店，需要将该商店的商品库存添加到 DataFrame 中。为此，我们可以向 `store_items` Dataframe 中添加一个新行。要向 DataFrame 中添加行，我们首先需要创建新的 Dataframe，然后将其附加到原始 DataFrame 上。我们来看看代码编写方式

```
# We create a dictionary from a list of Python dictionaries that w
new_items = [{'bikes': 20, 'pants': 30, 'watches': 35, 'glasses':

# We create new DataFrame with the new_items and provide and index
new_store = pd.DataFrame(new_items, index = ['store 3'])

# We display the items at the new store
new_store
```

	bikes	glasses	pants	watches
store 3	20	4	30	35

现在，我们使用 `.append()` 方法将此行添加到 `store_items` DataFrame 中。

```
# We append store 3 to our store_items DataFrame
store_items = store_items.append(new_store)

# We display the modified DataFrame
store_items
```

	bikes	glasses	pants	shirts	suits	watches
store 1	20	NaN	30	15.0	45.0	35
store 2	15	50.0	5	2.0	7.0	10

store 3	*20*	*4.0*	*30*	*NaN*	*NaN*	*35*

注意，将新行附加到 DataFrame 后，列按照字母顺序排序了。

我们还可以仅使用特定列的特定行中的数据向 DataFrame 添加新的列。例如，假设你想在商店 2 和 3 中上一批**新手表**，并且**新手表**的数量与这些商店原有手表的库存一样。我们来看看如何编写代码

```
# We add a new column using data from particular rows in the watcl
store_items['new watches'] = store_items['watches'][1:]

# We display the modified DataFrame
store_items
```

	bikes	*glasses*	*pants*	*shirts*	*suits*	*watches*	*wa*
store 1	*20*	*NaN*	*30*	*15.0*	*45.0*	*35*	
store 2	*15*	*50.0*	*5*	*2.0*	*7.0*	*10*	
store 3	*20*	*4.0*	*30*	*NaN*	*NaN*	*35*	

我们还可以将新列插入 DataFrames 的任何位置。`dataframe.insert(loc,label,data)` 方法使我们能够将新列（具有给定列`标签`和给定`数据`）插入 `dataframe` 的 `loc` 位置。我们将名称为 **shoes** 的新列插入 **suits** 列前面。因为 **suits** 的数字索引值为 4，我们将此值作为 `loc`。我们来看看代码编写方式：

```
# We insert a new column with label shoes right before the column
store_items.insert(4, 'shoes', [8,5,0])
```

	bikes	glasses	pants	shirts	shoes	suits	watc
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	NaN	0	NaN	35

就像我们可以添加行和列一样，我们也可以删除它们。要删除 DataFrame 中的行和列，我们将使用 `.pop()` 和 `.drop()` 方法。`.pop()` 方法仅允许我们删除列，而 `.drop()` 方法可以同时用于删除行和列，只需使用关键字 `axis` 即可。我们来看一些示例：

```
# We remove the new watches column
store_items.pop('new watches')

# we display the modified DataFrame
store_items
```

	bikes	glasses	pants	shirts	shoes	suits	watc
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	NaN	0	NaN	35

```
store_items = store_items.drop(['watches', 'shoes'], axis = 1)

# we display the modified DataFrame
store_items
```

	bikes	glasses	pants	shirts	suits
store 1	20	NaN	30	15.0	45.0
store 2	15	50.0	5	2.0	7.0
store 3	20	4.0	30	NaN	NaN

```
# We remove the store 2 and store 1 rows
store_items = store_items.drop(['store 2', 'store 1'], axis = 0)

# we display the modified DataFrame
store_items
```

	bikes	glasses	pants	shirts	suits
store 3	20	4.0	30	NaN	NaN

有时候，我们可能需要更改行和列标签。我们使用 `.rename()` 方法将 **bikes** 列标签改为 **hats**

```
# We change the column label bikes to hats
store_items = store_items.rename(columns = {'bikes': 'hats'})

# we display the modified DataFrame
store_items
```

store 3	*20*	*4.0*	*30*	*NaN*	*NaN*

现在再次使用 `.rename()` 方法更改行标签。

```
# We change the row label from store 3 to last store
store_items = store_items.rename(index = {'store 3': 'last store'})

# we display the modified DataFrame
store_items
```

	hats	*glasses*	*pants*	*shirts*	*suits*
last store	*20*	*4.0*	*30*	*NaN*	*NaN*

你还可以将索引改为 DataFrame 中的某个列。

```
# We change the row index to be the data in the pants column
store_items = store_items.set_index('pants')

# we display the modified DataFrame
store_items
```

pants	*hats*	*glasses*	*shirts*	*suits*
30	*20*	*4.0*	*NaN*	*NaN*