Accessing and Deleting Elements in Pandas Series

Now let's look at how we can access or modify elements in a Pandas Series. One great advantage of Pandas Series is that it allows us to access data in many different ways. Elements can be accessed using index labels or numerical indices inside square brackets, [ ], similar to how we access elements in NumPy ndarrays. Since we can use numerical indices, we can use both positive and negative integers to access data from the beginning or from the end of the Series, respectively. Since we can access elements in various ways, in order to remove any ambiguity to whether we are referring to an index label or numerical index, Pandas Series have two attributes, .loc and .iloc to explicitly state what we mean. The attribute .loc stands for *location* and it is used to explicitly state that we are using a labeled index. Similarly, the attribute .iloc stands for *integer location* and it is used to explicitly state that we are using a numerical index. Let's see some examples:

Example 1. Access elements using index labels

```
# We access elements in Groceries using index labels:


# We use a single index label

print('How many eggs do we need to buy:', groceries['eggs'])

print()


# we can access multiple index labels

print('Do we need milk and bread:\n', groceries[['milk', 'bread']])

print()


# we use loc to access multiple index labels

print('How many eggs and apples do we need to buy:\n', groceries.loc[['eggs', 'apples']])

print()


# We access elements in Groceries using numerical indices:


# we use multiple numerical indices

print('How many eggs and apples do we need to buy:\n',  groceries[[0, 1]])

print()
```

```
# We use a negative numerical index

print('Do we need bread:\n', groceries[[-1]])

print()


# We use a single numerical index

print('How many eggs do we need to buy:', groceries[0])

print()

# we use iloc to access multiple numerical indices

print('Do we need milk and bread:\n', groceries.iloc[[2, 3]])
```

```
How many eggs do we need to buy: 30

Do we need milk and bread:
milk       Yes
bread      No
dtype: object

How many eggs and apples do we need to buy:
eggs       30
apples      6
dtype: object

How many eggs and apples do we need to buy:
eggs       30
apples      6
dtype: object

Do we need bread:
bread      No
dtype: object

How many eggs do we need to buy: 30

Do we need milk and bread:
milk       Yes
bread      No
dtype: object
```

Pandas Series are also mutable like NumPy ndarrays, which means we can change the elements of a Pandas Series after it has been created. For example, let's change the number of eggs we need to buy from our grocery list

Example 2. Mutate elements using index labels

```
# We display the original grocery list

print('Original Grocery List:\n', groceries)


# We change the number of eggs to 2

groceries['eggs'] = 2


# We display the changed grocery list

print()

print('Modified Grocery List:\n', groceries)
```

```
Original Grocery List:
eggs      30
apples     6
milk     Yes
bread     No
dtype: object
```

```
Modified Grocery List:
eggs       2
apples     6
milk     Yes
bread     No
dtype: object
```

We can also delete items from a Pandas Series by using the .drop() method. The Series.drop(label) method removes the given label from the given Series. We should note that the Series.drop(label) method drops elements from the Series out-of-place, meaning that it doesn't change the original Series being modified. Let's see how this works:

Example 3. Delete elements out-of-place using drop()

```
# We display the original grocery list

print('Original Grocery List:\n', groceries)


# We remove apples from our grocery list. The drop function removes elements out of place

print()

print('We remove apples (out of place):\n', groceries.drop('apples'))
```

```
# When we remove elements out of place the original Series remains intact. To see this

# we display our grocery list again

print()

print('Grocery List after removing apples out of place:\n', groceries)
```

```
Original Grocery List:
eggs       30
apples      6
milk      Yes
bread      No
dtype: object

We remove apples (out of place):
eggs       30
milk      Yes
bread      No
dtype: object

Grocery List after removing apples out of place:
eggs       30
apples      6
milk      Yes
bread      No
dtype: object
```

We can delete items from a Pandas Series in place by setting the keyword inplace to True in the .drop() method. Let's see an example:

Example 4. Delete elements in-place using drop()

```
# We display the original grocery list

print('Original Grocery List:\n', groceries)


# We remove apples from our grocery list in place by setting the inplace keyword to True

groceries.drop('apples', inplace = True)


# When we remove elements in place the original Series its modified. To see this

# we display our grocery list again

print()
print('Grocery List after removing apples in place:\n', groceries)
```

```
Original Grocery List:
eggs        30
apples       6
milk       Yes
bread       No
dtype: object

Grocery List after removing apples in place:
eggs        30
milk       Yes
bread       No
dtype: object
```

Additional Reading - Pandas Series Documentation

Refer to the list of available functions in the following two sections:

- Reindexing / selection / label manipulation
- Indexing, and iteration