Series

Learn about the pandas Series object for 1-D data.

Chapter Goals

- Learn about the pandas Series object and its basic utilities
- Write code to create several Series objects

A. 1-D data

Similar to NumPy, pandas frequently deals with 1-D and 2-D data. However, we use two separate objects to deal with 1-D and 2-D data in pandas. For 1-D data, we use the `pandas.Series` objects, which we'll refer to simply as a Series.

A Series is created through the `pd.Series` constructor, which takes in no required arguments but does have a variety of keyword arguments.

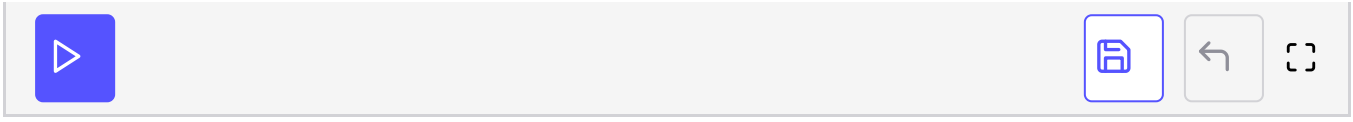
The first keyword argument is `data`, which specifies the elements of the Series. If `data` is not set, `pd.Series` returns an empty Series. Since the `data` keyword argument is almost always used, we treat it like a regular first argument (i.e. skip the `data=` prefix).

Similar to the `np.array` constructor, `pd.Series` also takes in the `dtype` keyword argument for manual casting.

The code below shows how to create pandas Series objects using `pd.Series`.

```
series = pd.Series()
# Newline to separate series print statements
print('{}\n'.format(series))

series = pd.Series(5)
print('{}\n'.format(series))

series = pd.Series([1, 2, 3])
print('{}\n'.format(series))

series = pd.Series([1, 2.2]) # upcasting
print('{}\n'.format(series))
```

```
arr = np.array([1, 2])
series = pd.Series(arr, dtype=np.float32)
print('{}\n'.format(series))

series = pd.Series([[1, 2], [3, 4]])
print('{}\n'.format(series))
```



In our examples, we initialized each Series with its values by setting the first argument using a scalar, list, or NumPy array. Note that `pd.Series` upcasts values in the same way as `np.array`. Furthermore, since Series objects are 1-D, the `ser` variable represents a Series with lists as elements, rather than a 2-D matrix.

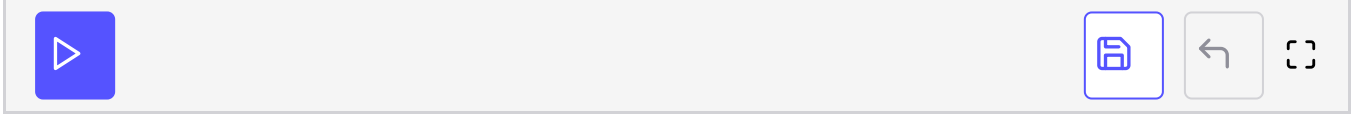
B. Index

In the previous examples, you may have noticed the zero-indexed integers to the left of the elements in each Series. These integers are collectively referred to as the *index* of a Series, and each individual index element is referred to as a *label*.

The default index is integers from 0 to n - 1, where n is the number of elements in the Series. However, we can specify a custom index via the `index` keyword argument of `pd.Series`.

The code below shows how to use the `index` keyword argument with `pd.Series`.

```
series = pd.Series([1, 2, 3], index=['a', 'b', 'c'])
print('{}\n'.format(series))

series = pd.Series([1, 2, 3], index=['a', 8, 0.3])
print('{}\n'.format(series))
```



The `index` keyword argument needs to be a list or array with the same length as the `data` argument for `pd.Series`. The values in the `index` list can be any hashable type (e.g. integer, float, string).

C. Dictionary input

Another way to set the index of a Series is by using a Python dictionary for the `data` argument. The keys of the dictionary represent the index of the Series, while each individual key is the label for its corresponding value.

The code below shows how to use `pd.Series` with a Python dictionary as the first argument. In our example, we set `'a'`, `'b'`, and `'c'` as the Series index, with corresponding values `1`, `2`, and `3`.

```
series = pd.Series({'a':1, 'b':2, 'c':3})
print('{}\n'.format(series))

series = pd.Series({'b':2, 'a':1, 'c':3})
print('{}\n'.format(series))
```

Time to Code!

The coding exercise for this chapter involves creating various pandas Series objects.

The first Series we create will contain basic floating point numbers. The list we use to initialize the Series is `[1, 3, 5.2]`.

Set `s1` equal to `pd.Series` with the specified list as the only argument.

```
# CODE HERE
```

The second Series we create comes from performing elemental multiplication on `s1` using a separate list of floating point numbers.

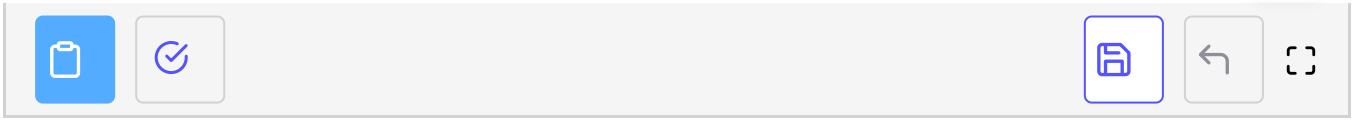
Set `s2` equal to `s1` multiplied by `[0.1, 0.2, 0.3]`.

```
# CODE HERE
```

We'll create another Series, this time with integers. The list we use to initialize this Series is `[1, 3, 8, np.nan]`. This Series will also have row labels, which will be `['a', 'b', 'c', 'd']`.

Set `s3` equal to `pd.Series` with the specified list of integers as the first argument and the list of labels as the `index` keyword argument.

```
# CODE HERE
```



The final Series we create will be initialized from a Python dictionary. The dictionary will have key-value pairs `'a':0`, `'b':1`, and `'c':2`.

Set `s4` equal to `pd.Series` with a dictionary of the specified key-value pairs as the only argument.

```
# CODE HERE
```