Thus in two lines, we've been able to find the average heart rate and temperature measured among all subjects in all visits each year. This syntax is actually a shortcut to the `GroupBy` functionality, which we will discuss in "Aggregation and Grouping" on page 158. While this is a toy example, many real-world datasets have similar hierarchical structure.

Panel Data

Pandas has a few other fundamental data structures that we have not yet discussed, namely the `pd.Panel` and `pd.Panel4D` objects. These can be thought of, respectively, as three-dimensional and four-dimensional generalizations of the (one-dimensional) `Series` and (two-dimensional) `DataFrame` structures. Once you are familiar with indexing and manipulation of data in a `Series` and `DataFrame`, `Panel` and `Panel4D` are relatively straightforward to use. In particular, the `ix`, `loc`, and `iloc` indexers discussed in "Data Indexing and Selection" on page 107 extend readily to these higher-dimensional structures.

We won't cover these panel structures further in this text, as I've found in the majority of cases that multi-indexing is a more useful and conceptually simpler representation for higher-dimensional data. Additionally, panel data is fundamentally a dense data representation, while multi-indexing is fundamentally a sparse data representation. As the number of dimensions increases, the dense representation can become very inefficient for the majority of real-world datasets. For the occasional specialized application, however, these structures can be useful. If you'd like to read more about the `Panel` and `Panel4D` structures, see the references listed in "Further Resources" on page 215.

Combining Datasets: Concat and Append

Some of the most interesting studies of data come from combining different data sources. These operations can involve anything from very straightforward concatenation of two different datasets, to more complicated database-style joins and merges that correctly handle any overlaps between the datasets. `Series` and `DataFrames` are built with this type of operation in mind, and Pandas includes functions and methods that make this sort of data wrangling fast and straightforward.

Here we'll take a look at simple concatenation of `Series` and `DataFrames` with the `pd.concat` function; later we'll dive into more sophisticated in-memory merges and joins implemented in Pandas.

We begin with the standard imports:

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

For convenience, we'll define this function, which creates a `DataFrame` of a particular form that will be useful below:

```
In[2]: def make_df(cols, ind):
           """Quickly make a DataFrame"""
           data = {c: [str(c) + str(i) for i in ind]
                   for c in cols}
           return pd.DataFrame(data, ind)

       # example DataFrame
       make_df('ABC', range(3))
```

```
Out[2]:    A   B   C
        0  A0  B0  C0
        1  A1  B1  C1
        2  A2  B2  C2
```

Recall: Concatenation of NumPy Arrays

Concatenation of `Series` and `DataFrame` objects is very similar to concatenation of NumPy arrays, which can be done via the `np.concatenate` function as discussed in "The Basics of NumPy Arrays" on page 42. Recall that with it, you can combine the contents of two or more arrays into a single array:

```
In[4]: x = [1, 2, 3]
       y = [4, 5, 6]
       z = [7, 8, 9]
       np.concatenate([x, y, z])
```

```
Out[4]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

The first argument is a list or tuple of arrays to concatenate. Additionally, it takes an `axis` keyword that allows you to specify the axis along which the result will be concatenated:

```
In[5]: x = [[1, 2],
            [3, 4]]
       np.concatenate([x, x], axis=1)
```

```
Out[5]: array([[1, 2, 1, 2],
               [3, 4, 3, 4]])
```

Simple Concatenation with pd.concat

Pandas has a function, `pd.concat()`, which has a similar syntax to `np.concatenate` but contains a number of options that we'll discuss momentarily:

```
# Signature in Pandas v0.18
pd.concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False,
          keys=None, levels=None, names=None, verify_integrity=False,
          copy=True)
```