**Using vectorized functions**

When performance is paramount, you should avoid using .apply() and .map() because those constructs perform Python for-loops over the data stored in a pandas Series or DataFrame. By using vectorized functions instead, you can loop over the data at the same speed as compiled code (C, Fortran, etc.)! NumPy, SciPy and pandas come with a variety of vectorized functions (called Universal Functions or UFuncs in NumPy).

You can even write your own vectorized functions, but for now we will focus on the ones distributed by NumPy and pandas.

In this exercise you're going to import the zscore method from scipy.stats and use it to compute the deviation in voter turnout in Pennsylvania from the mean in fractions of the standard deviation. In statistics, the z-score is the number of standard deviations by which an observation is *above* the mean - so if it is negative, it means the observation is *below* the mean.

Instead of using .apply() as you did in the earlier exercises, the zscore UFunc will take a pandas Series as input and return a NumPy array. You will then assign the values of the NumPy array to a new column in the DataFrame. You will be working with the election DataFrame - it has been pre-loaded for you.

**INSTRUCTIONS**

* Import zscore from scipy.stats.
* Call zscore with election['turnout'] as input .
* Print the output of type(turnout\_zscore). This has been done for you.
* Assign turnout\_zscore to a new column in election as 'turnout\_zscore'.
* Print the output of election.head().

# Import zscore from scipy.stats

from scipy.stats import zscore

# Call zscore with election['turnout'] as input: turnout\_zscore

turnout\_zscore = zscore(election['turnout'])

# Print the type of turnout\_zscore

print(type(turnout\_zscore))

# Assign turnout\_zscore to a new column: election['turnout\_zscore']

election['turnout\_zscore'] = turnout\_zscore

# Print the output of election.head()

print(election.head())