Applying Functions to Data

This lesson will teach us how to apply user defined functions to individual items in a dataset.

WE'LL COVER THE FOLLOWING

During data analysis, we need to use our data to perform some calculations and generate some new data or output from it. Pandas makes it very easy to apply user-defined operations, *a.k.a functions*, in Python terminology, on individual data items, rows, and columns of a dataframe.

Functions on individual Items of a column

Pandas has an `apply` function which applies the provided function to the data. One of the reasons for the success of pandas is how fast the `apply` function performs. We will be using the California Housing Dataset. All of the data is in the *housing.csv* file.

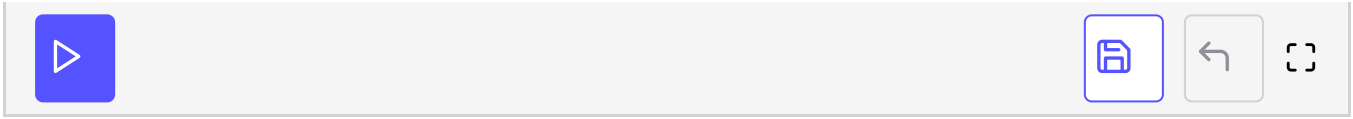
housing.csv

Example: Converting numbers

In the Dataset, the field `median_income` has values which are written in tens of thousands of dollars. During analysis, we might want to convert this to Dollars. Let's see how we can do that with the `apply` function.

```
# function that we will be applying to the data
def convert(value):
    return value * 10000

import pandas as pd
```

```
df = pd.read_csv('housing.csv')

print(df['median_income'].head())


# apply function on a column element by element
converted = df['median_income'].apply(convert)
print('\n\n converted \n',converted.head())

# change the original dataframe to new values
df['median_income'] = converted
```



We have defined the function that we want to apply in **lines 2-3**. We write our function so that it will receive each value of the column on which it is applied. We write it to operate on a single value. It will then be used on each of the values in the column.

In **line 11** we select the column `median_income` and use the `apply` function. We give the name of the function we defined above to it and store the results as `converted`. We replace the column in the original dataframe with `converted` in **line 15**.

Example: Converting numerical values to categories

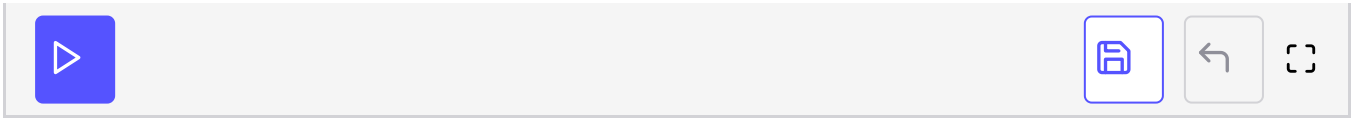
During analysis, sometimes we want to classify our data into separate classes based on some criteria. For instance, we might want to separate these housing blocks into three distinct categories based on the median income of the households i.e.

- High-incomes
- Moderate-incomes
- Low-incomes

We can do that by writing our own function that will check which range the values of the `median_income` column fall into.

```
# Function to convert a single numerical value to a category
def convert_categories(value):
    if value > 10:
        return 'high-incomes'
    elif value > 2 and value < 10:
        return 'moderate-incomes'
    else:
        return 'low-incomes'
```

```
# Read file
import pandas as pd
df = pd.read_csv('housing.csv')

# print original value
print('Original Values:')
print(df['median_income'].head())

# Apply Function on the column
categories = df['median_income'].apply(convert_categories)
print('Converted Values: ')
print(categories.head())

# make a new column in the dataframe
df['income_category'] = categories
```



We have defined our function `convert_categories` in **lines 2-8**. Remember the values in the dataset are in tens of thousands of dollars. We can say that if the value is greater than 10, it is a high-income housing block. If the values are between 2 and 10, it is a moderate-income housing block, and if the value is less than that, then it is a low-income housing block.

In **line 19** we select the column `median_income` and use the `apply` function. We give it the name of the function we defined above and store the results as `categories`. We add a new column in the original dataframe named `income_category` and save our `categories` in that column in **line 24**.

Now that we know how to operate on individual values in a column, we will see how we can use the whole column at once in the next lesson.