# Логика, Управление Потоком и Фильтрация

## Сравнение массивов

С помощью библиотеки NumPy можно сравнивать массивы при помощи операторов сравнения.

На этот раз есть два массива NumPy: my\_house и your\_house. Они содержат площади для кухни, гостиной, спальни и ванной комнаты в том же порядке, поэтому их можно сравнивать.

### Инструкции

С помощью операторов сравнения создайте булевые массивы, которые отвечают на следующие вопросы:

* Какие площади в my\_house больше или равны 18?
* Также можно сравнивать два массива NumPy поэлементно. Какие площади в my\_house меньше, чем в your\_house?
* Убедитесь, что обе команды обернуты в print(), чтобы можно было проверить вывод!

# Создание массивов  
import numpy as np  
my\_house = np.array([18.0, 20.0, 10.75, 9.50])  
your\_house = np.array([14.0, 24.0, 14.25, 9.0])  
  
# my\_house больше или равно 18  
  
# my\_house меньше, чем your\_house

[ True True False False]  
[False True True False]

## Булевы операторы с NumPy

Ранее операторы типа < и >= работали с массивами NumPy "из коробки". К сожалению, это не относится к булевым операторам and, or и not.

Для использования этих операторов с NumPy вам понадобятся np.logical\_and(), np.logical\_or() и np.logical\_not(). Вот пример на массивах my\_house и your\_house из предыдущего примера, чтобы вы поняли:

np.logical\_and(my\_house > 13,   
 your\_house < 15)

### Инструкции

* Создайте булевы массивы, которые отвечают на следующие вопросы:
* Какие площади в my\_house больше 18.5 или меньше 10?
* Какие площади меньше 11 в обоих my\_house и your\_house? Убедитесь, что обе команды обернуты в print(), чтобы можно было проверить вывод.

# my\_house больше 18.5 или меньше 10  
  
# Оба my\_house и your\_house меньше 11

[False True False True]  
[False False False True]

# Фильтрация pandas DataFrames

## Правостороннее движение (1)

Помните набор данных cars, содержащий количество автомобилей на 1000 человек (cars\_per\_cap) и информацию о том, едут ли люди с правосторонним движением (drives\_right) в разных странах (country)? В скрипте уже есть код, который импортирует эти данные в Python в формате CSV как DataFrame.

В видео вы видели пошаговый подход к фильтрации наблюдений из DataFrame на основе булевых массивов. Давайте начнем с простого и попробуем найти все наблюдения в cars, где drives\_right равно True.

drives\_right - это булевой столбец, поэтому вам нужно извлечь его как Series, а затем использовать этот булевой Series для выбора наблюдений из cars.

### Инструкции

* Извлеките столбец drives\_right как Pandas Series и сохраните его как dr.
* Используйте dr, булевой Series, для выбора подмножества DataFrame cars. Сохраните результат выбора в sel.
* Выведите sel и утвердите, что drives\_right равен True для всех наблюдений.

# Импорт данных об автомобилях  
import pandas as pd  
cars = pd.read\_csv('datasets/cars.csv', index\_col=0)  
  
# Извлечение столбца drives\_right как Series: dr  
  
# Использование dr для фильтрации cars: sel  
  
# Вывод sel

cars\_per\_cap country drives\_right  
US 809 United States True  
RU 200 Russia True  
MOR 70 Morocco True  
EG 45 Egypt True

## Правостороннее движение (2)

Код в предыдущем примере работал хорошо, но на самом деле вы лишний раз создали новую переменную dr. Вы можете достичь того же результата без этой промежуточной переменной. Поместите код, который вычисляет dr, прямо в квадратные скобки, которые выбирают наблюдения из cars.

### Инструкции

Преобразуйте код в однострочный, который вычисляет переменную sel, как и раньше.

# Преобразование кода в однострочный  
  
# Вывод sel

cars\_per\_cap country drives\_right  
US 809 United States True  
RU 200 Russia True  
MOR 70 Morocco True  
EG 45 Egypt True

## Автомобилей на человека (1)

Давайте продолжим работу с данными cars. На этот раз вы хотите выяснить, в каких странах высокий показатель автомобилей на человека. Другими словами, в каких странах у многих людей есть автомобиль, или может быть даже несколько автомобилей.

Аналогично предыдущему примеру, вам нужно создать булевой Series, который вы затем можете использовать для фильтрации DataFrame cars, чтобы выбрать определенные наблюдения. Если вы хотите сделать это в одной строке, это вполне нормально!

### Инструкции

* Выберите столбец cars\_per\_cap из cars как Pandas Series и сохраните его как cpc.
* Используйте cpc в сочетании с оператором сравнения и 500. Вы хотите получить булевой Series, который будет True, если у соответствующей страны cars\_per\_cap больше 500, и False в противном случае.
* Сохраните этот булевой Series как many\_cars.
* Используйте many\_cars для фильтрации cars, подобно тому, что вы делали ранее. Сохраните результат как car\_maniac.
* Выведите car\_maniac, чтобы убедиться, что все верно.

# Создание car\_maniac: наблюдения, у которых cars\_per\_cap больше 500  
  
# Вывод car\_maniac

cars\_per\_cap country drives\_right  
US 809 United States True  
AUS 731 Australia False  
JAP 588 Japan False

## Автомобилей на человека (2)

Помните о np.logical\_and(), np.logical\_or() и np.logical\_not(), вариантах операторов and, or и not в NumPy? Вы также можете использовать их с Pandas Series для более сложных операций фильтрации.

Возьмем этот пример, который выбирает наблюдения, у которых cars\_per\_cap находится между 10 и 80. Попробуйте эти строки кода пошагово, чтобы понять, что происходит.

Take this example that selects the observations that have a cars\_per\_cap between 10 and 80. Try out these lines of code step by step to see what's happening.

cpc = cars['cars\_per\_cap']  
between = np.logical\_and(cpc > 10, cpc < 80)  
medium = cars[between]

### Инструкции

* Используйте предоставленный образец кода, чтобы создать DataFrame medium, который включает все наблюдения автомобилей с cars\_per\_cap между 100 и 500.
* Выведите medium.

# Создание medium: наблюдения с cars\_per\_cap между 100 и 500  
# medium = cars[np.logical\_and(cars['cars\_per\_cap'] > 100, cars['cars\_per\_cap'] > 500)]  
  
# Вывод medium

cars\_per\_cap country drives\_right  
RU 200 Russia True