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
print(my_house >= 18)

# my_house меньше, чем your_house
print(my_house < 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
print(np.logical_or(my_house > 18.5, my_house < 10))

# Оба my_house и your_house меньше 11
print(np.logical_and(my_house < 11, your_house < 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
dr = cars['drives_right']

# Использование dr для фильтрации cars: sel
sel = cars[dr]
```

```
# Вывод sel
print(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
sel = cars[cars['drives_right']]

# Вывод sel
print(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
car_maniac = cars[cars['cars_per_cap'] > 500]

# Вывод car_maniac
print(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)]
cpc = cars['cars_per_cap']
between = np.logical_and(cpc > 100, cpc < 500)
medium = cars[between]

# Вывод medium
print(medium)
```

```
    cars_per_cap country  drives_right
RU           200  Russia          True
```