# NumPy

import pandas as pd  
  
baseball = pd.read\_csv('datasets/MLB(baseball).csv')  
print(baseball.head())

Name Team Position Height Weight Age PosCategory  
0 Adam\_Donachie BAL Catcher 74 180 22.99 Catcher  
1 Paul\_Bako BAL Catcher 74 215 34.69 Catcher  
2 Ramon\_Hernandez BAL Catcher 72 210 30.78 Catcher  
3 Kevin\_Millar BAL First\_Baseman 72 210 35.43 Infielder  
4 Chris\_Gomez BAL First\_Baseman 73 188 35.71 Infielder

### Инструкции

* Импортируйте пакет numpy как np, чтобы вы могли обращаться к numpy с помощью np.
* Используйте np.array() для создания массива numpy из baseball. Назовите этот массив np\_baseball.
* Выведите тип np\_baseball, чтобы проверить, что все верно.

# Импортируем пакет numpy как np  
import numpy as np  
  
# Создаем массив numpy из baseball: np\_baseball  
  
  
# Выводим тип np\_baseball

<class 'numpy.ndarray'>  
[['Adam\_Donachie' 'BAL' 'Catcher' ... 180 22.99 'Catcher']  
 ['Paul\_Bako' 'BAL' 'Catcher' ... 215 34.69 'Catcher']  
 ['Ramon\_Hernandez' 'BAL' 'Catcher' ... 210 30.78 'Catcher']  
 ...  
 ['Chris\_Narveson' 'STL' 'Relief\_Pitcher' ... 205 25.19 'Pitcher']  
 ['Randy\_Keisler' 'STL' 'Relief\_Pitcher' ... 190 31.01 'Pitcher']  
 ['Josh\_Kinney' 'STL' 'Relief\_Pitcher' ... 195 27.92 'Pitcher']]

## Рост игроков в бейсболе

Вы являетесь большим фанатом бейсбола. Вы решаете позвонить в MLB (Major League Baseball) и спросить о дополнительной статистике по росту основных игроков. Они передают данные о более чем тысяче игроков, которые хранятся в виде обычного списка Python: height\_in. Рост выражен в дюймах. Можете ли вы создать из этого массив numpy и перевести значения в метры?

Переменная height\_in уже доступна, а пакет numpy загружен, так что вы можете начать сразу.

### Инструкции

* Создайте массив numpy из height\_in. Назовите этот новый массив np\_height\_in.
* Выведите np\_height\_in.
* Умножьте np\_height\_in на 0.0254, чтобы преобразовать все значения роста из дюймов в метры. Сохраните новые значения в новом массиве np\_height\_m.
* Выведите np\_height\_m и проверьте, имеет ли вывод смысл.

# Предполагая, что 'Height' находится в индексе 3 в вашем массиве  
  
  
# Преобразуем np\_height\_in в метры: np\_height\_m  
  
  
# Выводим np\_height\_m

[74 74 72 ... 75 75 73]  
[1.8796 1.8796 1.8288 ... 1.905 1.905 1.8541999999999998]

## ИМТ бейсбольных игроков

MLB также предлагает вам проанализировать их данные о весе. Опять же, оба доступны как обычные списки Python: height\_in (рост в дюймах) и weight\_lb (вес в фунтах).

Теперь можно рассчитать Индекс массы тела (ИМТ) каждого игрока в бейсболе. Python-код для преобразования height\_in в массив numpy с правильными единицами уже доступен в рабочем пространстве. Следуйте инструкциям шаг за шагом и завершите задание! height\_in и weight\_lb доступны как обычные списки.

### Инструкции

* Создайте массив numpy из списка weight\_lb с правильными единицами. Умножьте на 0.453592, чтобы перейти из фунтов в килограммы.
* Сохраните полученный массив numpy как np\_weight\_kg.
* Используйте np\_height\_m и np\_weight\_kg, чтобы рассчитать Индекс массы тела (ИМТ) каждого игрока
* Используйте следующее уравнение: **bmi = np\_weight\_kg / np\_height\_m**
* Сохраните полученный массив numpy как bmi.
* Выведите bmi.

# Создаем массив из weight\_lb с метрическими единицами: np\_weight\_kg  
  
np\_weight\_kg = np.array(weight\_lb) \* 0.453592  
  
# Вычисляем ИМТ: bmi  
  
  
# Выводим ИМТ

[23.11037638875862 27.604060686572797 28.48080464679448 ...  
 25.62295933480756 23.74810865177286 25.726863613607133]

## Легкие игроки в бейсболе

Для подмножеств обычных списков Python и массивов numpy можно использовать квадратные скобки:

x = [4 , 9 , 6, 3, 1]  
  
x[1]  
  
y = np.array(x)  
y[1]

Для numpy можно также использовать логические массивы:

high = y > 5  
y[high]

Код, который вычисляет ИМТ всех игроков в бейсболе, уже включен. Следуйте инструкциям и обнаружьте интересные факты из данных! height\_in и weight\_lb доступны как обычные списки.

### Инструкции

* Создайте логический массив numpy: элемент массива должен быть True, если ИМТ соответствующего игрока в бейсболе ниже 21.
* Для этого можно использовать оператор <. Назовите массив light.
* Выведите массив light.
* Выведите массив numpy с ИМТ всех игроков в бейсболе, у которых ИМТ ниже 21. Используйте light внутри квадратных скобок для выбора элементов массива bmi.

# Создаем массив light  
  
  
# Выводим light  
  
  
# Выводим ИМТ всех игроков в бейсболе, у которых ИМТ ниже 21

[False False False ... False False False]

array([20.542556790007662, 20.542556790007662, 20.69282047151352,  
 20.69282047151352, 20.343431890567484, 20.343431890567484,  
 20.69282047151352, 20.158834718074228, 19.498447103560874,  
 20.69282047151352, 20.92052190452328], dtype=object)

## Выделение подмножеств в массивах NumPy

Вы видели это своими глазами: списки Python и массивы numpy иногда ведут себя по-разному. К счастью, в этом мире все еще есть некоторые уверенности. Например, выделение подмножеств (использование квадратной скобочной нотации для списков или массивов) работает точно так же. Чтобы убедиться в этом, попробуйте следующие строки кода в

IPython Shell:  
  
x = ["a", "b", "c"]  
x[1]  
  
np\_x = np.array(x)  
np\_x[1]

### Инструкции

* Выделите подмассив np\_weight\_lb, выведя элемент с индексом 50.
* Выведите подмассив np\_height\_in, который содержит элементы с индекса 100 до и включительно индекса 110.

# Сохраняем списки веса и роста как массивы numpy  
np\_weight\_lb = np\_baseball[:, 4]  
np\_height\_in = np\_baseball[:, 3]  
  
# Выводим вес с индексом 50  
  
  
# Выводим подмассив np\_height\_in: от индекса 100 до включительно индекса 110

200  
[73 74 72 73 69 72 73 75 75 73 72]

## Ваш первый двумерный массив NumPy

Перед тем как работать с фактическими данными MLB, давайте попробуем создать двумерный массив numpy из небольшого списка списков.

В этом упражнении baseball\_ex - это список списков. Основной список содержит 4 элемента. Каждый из этих элементов - это список, содержащий рост и вес 4 игроков baseball\_ex в данном порядке. baseball\_ex уже задан для вас в скрипте.

### Инструкции

* Используйте np.array(), чтобы создать двумерный массив numpy из baseball. Назовите его np\_baseball\_ex.
* Выведите тип np\_baseball\_ex.
* Выведите атрибут формы np\_baseball\_ex. Используйте np\_baseball\_ex.shape.

# Создаем baseball, список списков  
baseball\_ex = [[180, 78.4],  
 [215, 102.7],  
 [210, 98.5],  
 [188, 75.2]]  
  
# Создаем двумерный массив numpy из baseball: np\_baseball  
  
  
# Выводим тип np\_baseball  
  
  
# Выводим форму np\_baseball

<class 'numpy.ndarray'>  
(4, 2)

## Данные о бейсболистах в виде 2D массива

Вы еще раз взглянули на данные MLB и поняли, что будет более логично переструктурировать всю эту информацию в виде двумерного массива numpy. Этот массив должен иметь 1015 строк, соответствующих 1015 бейсболистам, о которых у вас есть информация, и 2 столбца (для роста и веса).

MLB снова была очень полезной и передала вам данные в другой структуре - в виде обычного списка списков Python. В этом списке списков каждый подсписок представляет собой рост и вес одного бейсболиста. Имя этого вложенного списка - baseball.

Можете ли вы сохранить данные в виде двумерного массива, чтобы использовать дополнительные функциональные возможности numpy? baseball доступен как обычный список списков.

### Инструкции

* Используйте np.array(), чтобы создать двумерный массив numpy из baseball. Назовите его np\_baseball.
* Выведите атрибут формы np\_baseball.

# Создаем двумерный массив numpy из baseball: np\_baseball  
  
  
# Выводим форму np\_baseball

(1015, 7)

## Выделение подмножеств 2D массивов NumPy

Если ваш двумерный массив numpy имеет регулярную структуру, то есть каждая строка и столбец имеют фиксированное количество значений, сложные способы выделения подмножеств становятся очень простыми. Взгляните на приведенный ниже код, где элементы "a" и "c" извлекаются из списка списков.

# Обычный список списков  
x = [["a", "b"], ["c", "d"]]  
[x[0][0], x[1][0]]  
  
# numpy  
import numpy as np  
np\_x = np.array(x)  
np\_x[:, 0]

Для обычных списков Python это настоящая боль. Однако для двумерных массивов numpy это довольно интуитивно! Индексы перед запятой относятся к строкам, а после запятой - к столбцам. : используется для среза; в этом примере он говорит Python включить все строки.

Код, который преобразует предварительно загруженный список baseball в двумерный массив numpy, уже находится в скрипте. Первый столбец содержит рост игроков в дюймах, а второй столбец - вес игроков в фунтах. Добавьте несколько строк, чтобы сделать правильные выборки. Помните, что в Python первый элемент имеет индекс 0! baseball доступен как обычный список списков.

#### Инструкции

* Создайте np\_baseball\_he\_we, содержащий столбцы с ростом и весом из baseball.
* Выведите 50-ую строку np\_baseball\_he\_we.
* Создайте новую переменную np\_weight\_lb, содержащую весь второй столбец np\_baseball.
* Выберите рост (первый столбец) 124-ого игрока в np\_baseball и выведите его.

# Создаем np\_baseball\_he\_we, содержащий столбцы 'Height' и 'Weight'  
np\_baseball\_he\_we =   
  
# Выводим 50-ую строку np\_baseball  
  
  
# Выбираем весь второй столбец np\_baseball: np\_weight\_lb  
  
  
# Выводим рост 124-ого игрока

[70 195]  
75

## 2D Арифметика

Помните, как вы рассчитывали Индекс Массы Тела для всех игроков в бейсболе? numpy мог выполнять все вычисления покомпонентно (то есть поэлементно). Для двумерных массивов numpy это не отличается! Вы можете комбинировать матрицы с одиночными числами, векторами и другими матрицами.

Выполните код ниже в оболочке IPython и проверьте, понимаете ли вы:

import numpy as np  
np\_mat = np.array([[1, 2],  
 [3, 4],  
 [5, 6]])  
np\_mat \* 2  
np\_mat + np.array([10, 10])  
np\_mat + np\_mat

np\_baseball\_he\_we\_ye закодирован для вас; это снова двумерный массив numpy с 3 столбцами, представляющими рост (в дюймах), вес (в фунтах) и возраст (в годах).

### Инструкции

* У вас удалось получить данные об изменениях роста, веса и возраста всех игроков в бейсболе. Эти данные доступны как двумерный массив numpy, updated.
* Сложите np\_baseball\_he\_we\_ye и updated и выведите результат.
* Вы хотите преобразовать единицы измерения роста и веса в метрическую систему (метры и килограммы соответственно). В качестве первого шага создайте массив numpy из трех значений: 0.0254, 0.453592 и 1. Назовите этот массив conversion.
* Умножьте np\_baseball\_he\_we\_ye на conversion и выведите результат.

# Создаем np\_baseball\_he\_we\_ye (3 столбца)  
np\_baseball\_he\_we\_ye =   
  
# Создаем массив numpy: conversion  
  
  
# Выводим произведение np\_baseball\_he\_we\_ye и conversion  
  
  
# Применяем конверсию к столбцам Рост и Вес

[[1.8796 81.64656 22.99]  
 [1.8796 97.52228 34.69]  
 [1.8288 95.25431999999999 30.78]  
 ...  
 [1.905 92.98636 25.19]  
 [1.905 86.18248 31.01]  
 [1.8541999999999998 88.45044 27.92]]

## Среднее против медианы

Теперь вы знаете, как использовать функции numpy, чтобы получить лучшее представление о ваших данных. Это, по сути, сводится к импорту numpy и вызову нескольких простых функций для массивов numpy:

import numpy as np  
x = [1, 4, 8, 10, 12]  
np.mean(x)  
np.median(x)

Данные о бейсболе доступны как двумерный массив numpy с 3 столбцами (рост, вес, возраст) и 1015 строками. Имя этого массива numpy - np\_baseball\_he\_we\_ye. После переструктурирования данных, однако, вы замечаете, что некоторые значения роста аномально высокие. Следуйте инструкциям и определите, какая сводная статистика лучше всего подходит, если у вас есть так называемые выбросы (outliers). np\_baseball доступен.

### Инструкции

* Создайте массив numpy np\_baseball\_he\_we\_ye, который равен первому столбцу np\_baseball.
* Выведите среднее значение (mean) np\_baseball\_he\_we\_ye.
* Выведите медиану (median) np\_baseball\_he\_we\_ye.

# Создаем np\_baseball\_he\_we\_ye из np\_baseball  
np\_baseball\_he\_we\_ye =   
  
# Выводим среднее значение np\_baseball\_he\_we\_ye  
  
  
# Выводим медиану np\_baseball\_he\_we\_ye

101.24892610837442  
74.0

## Изучение данных о бейсболе

Поскольку среднее значение (mean) и медиана (median) находятся далеко друг от друга, вы решаете пожаловаться в MLB. Они находят ошибку и отправляют вам исправленные данные. Они снова доступны как двумерный массив NumPy np\_baseball\_he\_we\_ye с тремя столбцами.

Сценарий Python в редакторе уже включает код для вывода информативных сообщений с различными сводными статистиками. Можете ли вы завершить задачу? np\_baseball доступен.

### Инструкции

* Код для вывода среднего значения роста уже включен. Завершите код для медианы роста. Замените None на правильный код.
* Используйте np.std() для первого столбца np\_baseball для вычисления stddev. Замените None на правильный код.
* Тяжелее ли обычно более высокие игроки? Используйте np.corrcoef() для сохранения корреляции между первым и вторым столбцом np\_baseball в corr. Замените None на правильный код.

# Выводим средний рост (первый столбец)  
avg = np.mean(np\_baseball\_he\_we\_ye[:,0])  
print("Average: " + str(avg))  
  
# Выводим медиану роста. Замените 'None'  
med = np.median(np\_baseball\_he\_we\_ye[:,0])  
print("Median: " + str(med))  
  
# Выводим стандартное отклонение по росту. Замените 'None'  
stddev = np.std(np\_baseball\_he\_we\_ye[:,0])  
print("Standard Deviation: " + str(stddev))  
  
# Выводим корреляцию между первым и вторым столбцом. Замените 'None'  
# corr = np.corrcoef(np\_baseball\_he\_we\_ye[:,0], np\_baseball\_he\_we\_ye[:,1])  
# print("Correlation: " + str(corr))  
  
# Убеждаемся, что рост и вес имеют тип float  
heights = np\_baseball\_he\_we\_ye[:, 0].astype(float)  
weights = np\_baseball\_he\_we\_ye[:, 1].astype(float)  
  
# Рассчитываем корреляцию между ростом и весом  
corr = np.corrcoef(heights, weights)  
print("Correlation:", corr)

Average: 73.6896551724138  
Median: 74.0  
Standard Deviation: 2.3127918810465395  
Correlation: [[1. 0.53153932]  
 [0.53153932 1. ]]

## Смешиваем все вместе

В последних нескольких упражнениях вы узнали все, что нужно знать о росте и весе игроков в бейсболе. Теперь пришло время погрузиться в другой вид спорта: футбол.

Вы обратились в FIFA за данными, и они передали вам два списка. Списки выглядят следующим образом:

positions = ['GK', 'M', 'A', 'D', ...]  
heights = [191, 184, 185, 180, ...]

Каждый элемент в списках соответствует игроку. Первый список, positions, содержит строки, представляющие позицию каждого игрока. Возможные позиции: 'GK' (вратарь), 'M' (полузащитник), 'A' (нападающий) и 'D' (защитник). Второй список, heights, содержит целые числа, представляющие рост игрока в сантиметрах. Первый игрок в списках - вратарь и довольно высокий (191 см).

Вы довольно уверены, что медиана роста вратарей выше, чем у других игроков на футбольном поле. Некоторые из ваших друзей в это не верят, поэтому вы решили показать им это, используя данные, которые вы получили от FIFA, и свои недавно приобретенные навыки программирования на Python. heights и positions доступны как списки.

### Инструкции

* Преобразуйте heights и positions, которые являются обычными списками, в массивы numpy. Назовите их np\_heights и np\_positions.
* Извлеките все росты вратарей. Здесь можно использовать маленький трюк: используйте np\_positions == 'GK' как индекс для np\_heights. Присвойте результат переменной gk\_heights.
* Извлеките все росты остальных игроков. В этот раз используйте np\_positions != 'GK' как индекс для np\_heights. Присвойте результат переменной other\_heights.
* Выведите медиану роста вратарей с помощью np.median(). Замените None на правильный код.
* Сделайте то же самое для остальных игроков. Выведите их медиану роста. Замените None на правильный код.

# Читаем CSV-файл с другой кодировкой  
football = pd.read\_csv('datasets/FIFA(Football).csv', encoding='ISO-8859-1')  
  
# Удаляем ведущие и завершающие пробелы из всех строковых значений в DataFrame football  
football = football.applymap(lambda x: x.strip() if isinstance(x, str) else x)  
  
# Создаем массив numpy из football: np\_football  
np\_football =   
  
# Преобразуем позиции и рост в массивы numpy: np\_positions, np\_heights  
np\_positions =   
np\_heights =   
  
# Рост вратарей: gk\_heights  
gk\_heights =   
  
# Рост остальных игроков: other\_heights  
other\_heights =   
  
# Выводим медиану роста вратарей. Замените 'None'  
print("Медиана роста вратарей: " + str(np.median(gk\_heights)))  
  
# Выводим медиану роста других игроков. Замените 'None'  
print("Медиана роста других игроков: " + str(np.median(other\_heights)))

Медиана роста вратарей: 188.0  
Медиана роста других игроков: 181.0

/var/folders/xr/k\_0c\_95x3d75308lz6wxm2\_80000gn/T/ipykernel\_21192/1461017646.py:5: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.  
 football = football.applymap(lambda x: x.strip() if isinstance(x, str) else x)