# Циклические структуры данных

## Цикл по словарю

В Python 3 для перебора словаря используется метод items():

world = { "afghanistan":30.55,   
 "albania":2.77,  
 "algeria":39.21 }  
  
for key, value in world.items() :  
 print(key + " -- " + str(value))

Помните словарь europe, который содержал названия некоторых европейских стран в качестве ключей и их столицы как соответствующие значения? Давайте напишем цикл для его итерации!

### Инструкции

Напишите цикл for, который проходит через каждую пару ключ:значение в europe. На каждой итерации должно выводиться столица x - это y, где x - ключ, а y - значение пары.

# Определение словаря  
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin',  
 'norway':'oslo', 'italy':'rome', 'poland':'warsaw', 'austria':'vienna' }  
  
# Итерация по Europe

Столица spain - это madrid  
Столица france - это paris  
Столица germany - это berlin  
Столица norway - это oslo  
Столица italy - это rome  
Столица poland - это warsaw  
Столица austria - это vienna

## Цикл по массиву NumPy

Если у вас есть одномерный массив NumPy, перебор всех элементов может быть таким простым:

for x in my\_array :  
 ...

Если у вас есть двумерный массив NumPy, это более сложно. 2D-массив состоит из нескольких 1D-массивов. Чтобы явно перебрать все отдельные элементы многомерного массива, вам понадобится следующий синтаксис:

for x in np.nditer(my\_array) :  
 ...

Два массива NumPy, которые вы, возможно, узнаете из вводного курса, доступны в вашей сессии Python: np\_height, массив NumPy, содержащий рост игроков Major League Baseball, и np\_baseball, 2D-массив NumPy, который содержит и рост (первый столбец), и вес (второй столбец) этих игроков.

### Инструкции

* Импортируйте пакет numpy под локальным псевдонимом np.
* Напишите цикл for, который перебирает все элементы в np\_height и выводит "x дюймов" для каждого элемента, где x - значение в массиве.
* Напишите цикл for, который посещает каждый элемент массива np\_baseball и выводит его.

import pandas as pd  
import numpy as np  
baseball = pd.read\_csv('datasets/MLB(baseball).csv')  
  
np\_height = np.array(baseball['Height'])  
np\_baseball = np.array(baseball[['Height', 'Weight']])  
  
# Цикл по np\_height  
  
# Цикл по np\_baseball

74 дюймов  
74 дюймов  
72 дюймов  
72 дюймов  
…

74  
74  
72  
72  
73  
69  
69  
71  
76  
71  
73  
73  
74

……  
180  
215  
210  
210  
188  
176  
209  
200  
231  
180  
188  
180  
185  
160  
180  
185

## Цикл по DataFrame (1)

Итерация по объекту Pandas DataFrame обычно выполняется с помощью метода iterrows(). Используя его в цикле for, каждое наблюдение перебирается, и на каждой итерации метка строки и собственно содержимое этой строки становятся доступными:

for lab, row in brics.iterrows() :  
 ...

В этом и последующих упражнениях вы будете работать с DataFrame под названием cars. Он содержит информацию о количестве автомобилей на человека и о том, едут ли люди направо или налево, для семи стран мира.

### Инструкции

Напишите цикл for, который перебирает строки cars и на каждой итерации выполняет два вызова print(): один для вывода метки строки и второй для вывода всех содержимого строки.

cars = pd.read\_csv('datasets/cars.csv', index\_col = 0)  
  
# Итерация по строкам автомобилей

US  
cars\_per\_cap 809  
country United States  
drives\_right True  
Name: US, dtype: object  
AUS  
cars\_per\_cap 731  
country Australia  
drives\_right False  
Name: AUS, dtype: object  
JAP  
cars\_per\_cap 588  
country Japan  
drives\_right False  
Name: JAP, dtype: object  
IN  
cars\_per\_cap 18  
country India  
drives\_right False  
Name: IN, dtype: object  
RU  
cars\_per\_cap 200  
country Russia  
drives\_right True  
Name: RU, dtype: object  
MOR  
cars\_per\_cap 70  
country Morocco  
drives\_right True  
Name: MOR, dtype: object  
EG  
cars\_per\_cap 45  
country Egypt  
drives\_right True  
Name: EG, dtype: object

## Цикл по DataFrame (2)

Данные строки, которые генерируются при каждом выполнении iterrows(), представляют собой объект Pandas Series. Этот формат не очень удобен для вывода. К счастью, вы можете легко выбирать переменные из объекта Pandas Series, используя квадратные скобки:

for lab, row in brics.iterrows() :  
 print(row['country'])

### Инструкции

* Используя итераторы lab и row, измените код в цикле так, чтобы при первой итерации выводилось US: 809, при второй AUS: 731 и так далее.
* Вывод должен иметь форму "country: cars\_per\_cap". Убедитесь, что выводится именно эта строка (с правильным пробелом).
* Вы можете использовать функцию str(), чтобы преобразовать целочисленные данные в строку и вывести их вместе с меткой страны.

# Адаптировать цикл  
  
for lab, row in cars.iterrows() :  
 # Напечатать (lab)

US: 809  
AUS: 731  
JAP: 588  
IN: 18  
RU: 200  
MOR: 70  
EG: 45

## Добавление столбца (1)

Добавить length (длину) названий стран из DataFrame brics в новый столбец:

for lab, row in brics.iterrows() :  
 brics.loc[lab, "name\_length"] = len(row["country"])

Вы можете сделать нечто подобное с DataFrame cars.

### Инструкции

* Используйте цикл for, чтобы добавить новый столбец с именем COUNTRY, который содержит верхний регистр названий стран из столбца "country". Для этого можно использовать метод upper() для строк.
* Чтобы убедиться, что ваш код работает, выведите cars.
* Не делайте отступ для этого кода, чтобы он не был частью цикла for.

# Код цикла, добавляющего столбец COUNTRY  
  
# Вывод cars

cars\_per\_cap country drives\_right COUNTRY  
US 809 United States True UNITED STATES  
AUS 731 Australia False AUSTRALIA  
JAP 588 Japan False JAPAN  
IN 18 India False INDIA  
RU 200 Russia True RUSSIA  
MOR 70 Morocco True MOROCCO  
EG 45 Egypt True EGYPT

## Добавление столбца (2)

Использование iterrows() для итерации по каждому наблюдению объекта Pandas DataFrame легко понять, но не очень эффективно. На каждой итерации создается новый объект Pandas Series.

Если вы хотите добавить столбец в DataFrame, вызывая функцию для другого столбца, метод iterrows() в сочетании с циклом for не является предпочтительным способом. Вместо этого вам следует использовать apply().

Сравните версию с iterrows() с версией с apply(), чтобы получить тот же результат в DataFrame brics:

for lab, row in brics.iterrows() :  
 brics.loc[lab, "name\_length"] = len(row["country"])  
  
brics["name\_length"] = brics["country"].apply(len)

Мы можем сделать нечто подобное для вызова метода upper() для каждого имени в столбце country. Однако upper() - это метод, поэтому нам понадобится немного другой подход:

## Инструкции

* Замените цикл for однострочным кодом, использующим .apply(str.upper). Вызов должен дать тот же результат: в cars должен быть добавлен столбец COUNTRY, содержащий верхний регистр названий стран.
* Как обычно, выведите cars, чтобы увидеть результат вашего труда.

# Использование .apply(str.upper)  
for lab, row in cars.iterrows() :

cars\_per\_cap country drives\_right COUNTRY  
US 809 United States True UNITED STATES  
AUS 731 Australia False AUSTRALIA  
JAP 588 Japan False JAPAN  
IN 18 India False INDIA  
RU 200 Russia True RUSSIA  
MOR 70 Morocco True MOROCCO  
EG 45 Egypt True EGYPT