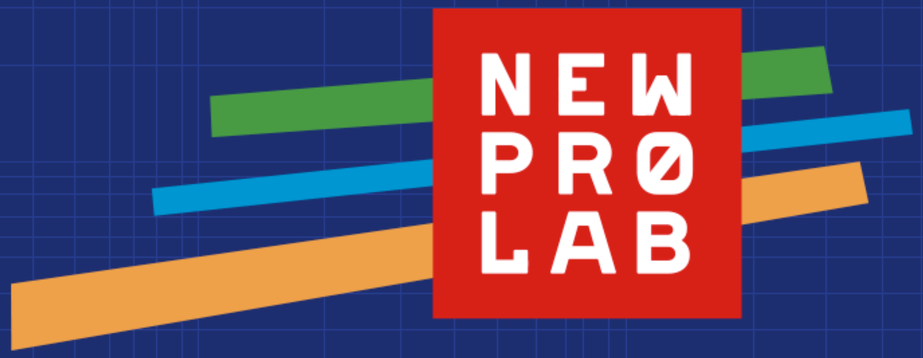


Практикум по Python: инструменты и подходы

Николай Марков
@enchantner

NEWPROLAB.COM

ЧТО ЕЩЕ ЗА PYTHON?

- Язык общего назначения
- Интерпретируемый
- С динамической (“утиной”) типизацией
- Автоматическое управление памятью и Garbage Collector
- Эталонная реализация - CPython (также существуют Pypy, Jython, IronPython)
- Все - объект

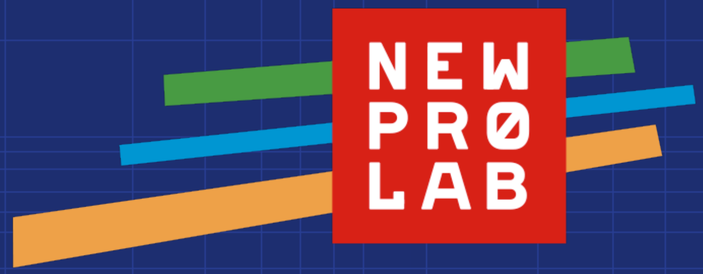
ПОЛЕЗНЫЕ ВЕЩИ

https://www.python.org/ # “ванильный” дистрибутив
http://www.lfd.uci.edu/~gohlke/pythonlibs/ # предсобрано для Windows
https://www.anaconda.com/distribution/ # data science дистрибутив

- easy_install (setuptools) - старый менеджер пакетов (практически не используется)
- pip - стандартный менеджер пакетов
- virtualenv - установить конкретные версии
- virtualenvwrapper - отличная обертка для пакетов локально
- virtualenv http://docs.python-guide.org/en/latest/dev/virtualenvs/

СОВЕТЫ ПО НАСТРОЙКЕ ОКРУЖЕНИЯ

- virtualenv+pip - предпочтительный вариант, особенно если хотите свежие версии пакетов
- как вариант - окружения Anaconda и менеджер conda
- альтернатива - использовать системный менеджер пакетов
  - в Mac OS - Homebrew (*~$ brew install python3*)
  - в Ubuntu/Debian - Apt (*~$ sudo apt install python3 python3-dev python3-venv*)
  - В CentOS/Fedora - Yum/Dnf
- Начни свой проект с *~$ pip install -U pip wheel setuptools*
- А вот этого не стоит делать никогда, за редким исключением: ~$ sudo pip install …



Jupyter Notebook, ipyparallel

IPython shell

- Подстановка по Tab
- История команд
- Интеграция с системным shell
- Встроенный механизм показа справки ("?")
- %magic
- Редактирование многострочных кусков кода

http://ipython.org/

ipyparallel

- Как, говорите? GIL?
- 0MQ + Kernels
- Поддержка платформ наподобие EC2
- mpi4py
- %px
- Task DAG

http://ipython.org/ipython-doc/dev/parallel/

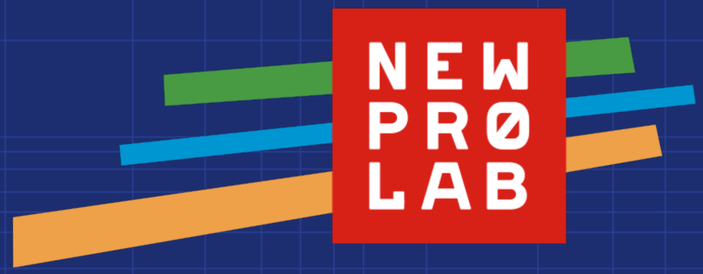
Jupyter Notebook

- Идея подсмотрена у Mathematica и RStudio
- Независимые ячейки с кодом
- Текстовые данные, Markdown и LaTeX
- Простой текстовый формат .ipynb
- Inline-графика, в том числе трехмерная и интерактивная
- Поддержка «ядер»
- https://github.com/jupyterlab/jupyterlab

@ellisonbg @fperez_org

Упражнение

1. Запустить Jupyter notebook
2. Создать новый файл с ядром Python 3
3. Вывести справку по магическим командам
4. Найти команду измерения времени выполнения выражения и померить время генерации списка из 100000 чисел



Генераторы и корутины

Дэвид Бизли

http://dabeaz.com/generators-uk/index.html

http://dabeaz.com/coroutines/index.html

Генераторные выражения

```
# в чем разница?
a = [i for i in xrange(10)]

a = (i for i in xrange(10))
b = (n for n in iterator if "foo" in n)

d = {n: i for i, n in iterator_by_pairs}
```

https://xakep.ru/2014/10/06/generatora-iteratory-python/

Yield

```
def gen_squares(n):
    for i in xrange(n):
        yield i * i


mygen = gen_squares(100)
for i in mygen:  # фактически вызов next(mygen)
    print(i)
```

Значения в генераторе имеют свойство заканчиваться!

Почему это круто?

- В памяти одновременно находится только один объект - идеально для задач стриминга
- Можно строить «цепочки» (помните конвейеры в Bash?)
- Можно передавать значения из одной сопрограммы в другую - «кооперативная многозадачность»

Асинхронность и параллельность

- Параллельность - это выполнение двух фрагментов кода одновременно.
- Асинхронность - это выполнение кода НЕ последовательно.
- Асинхронность может быть реализована с помощью параллельности, а может
- с помощью ручного переключения контекста в самом коде, с сохранением последнего состояния. Ничего не напоминает?
- Когда куски кода сами решают, когда передавать управление друг другу, и не зависят от внешнего системного планировщика, то это называется "кооперативной многозадачностью", а эти куски кода - корутинами или сопрограммами.
- Недостаток - долгоиграющая процедура НЕ под контролем event loop'а вешает вообще ВСЕ

Событийно-ориентированное программирование

- Две основные составляющие асинхронного кода - это event loop (цикл отлова событий) и корутины
- Пока корутина ждет внешнее событие - контекст переключается на другую
- Помимо переключения контекста корутины могут отправлять друг другу сообщения
- К сожалению, в современной реализации асинхронности в Python обычные и асинхронные функции не являются взаимозаменяемыми
- Альтернативные реализации для старых версий - Gevent, Eventlet и Tornado. И еще несколько.

Asyncio

```
import asyncio

async def hello(name):
    return "Hello, {}!".format(name)

async def call_vasya():
    greeting = await hello("Vasya")
    return greeting

loop = asyncio.get_event_loop()
print(loop.run_until_complete(call_vasya()))
```

https://xakep.ru/2017/01/11/python-3-asyncio/

Asyncio

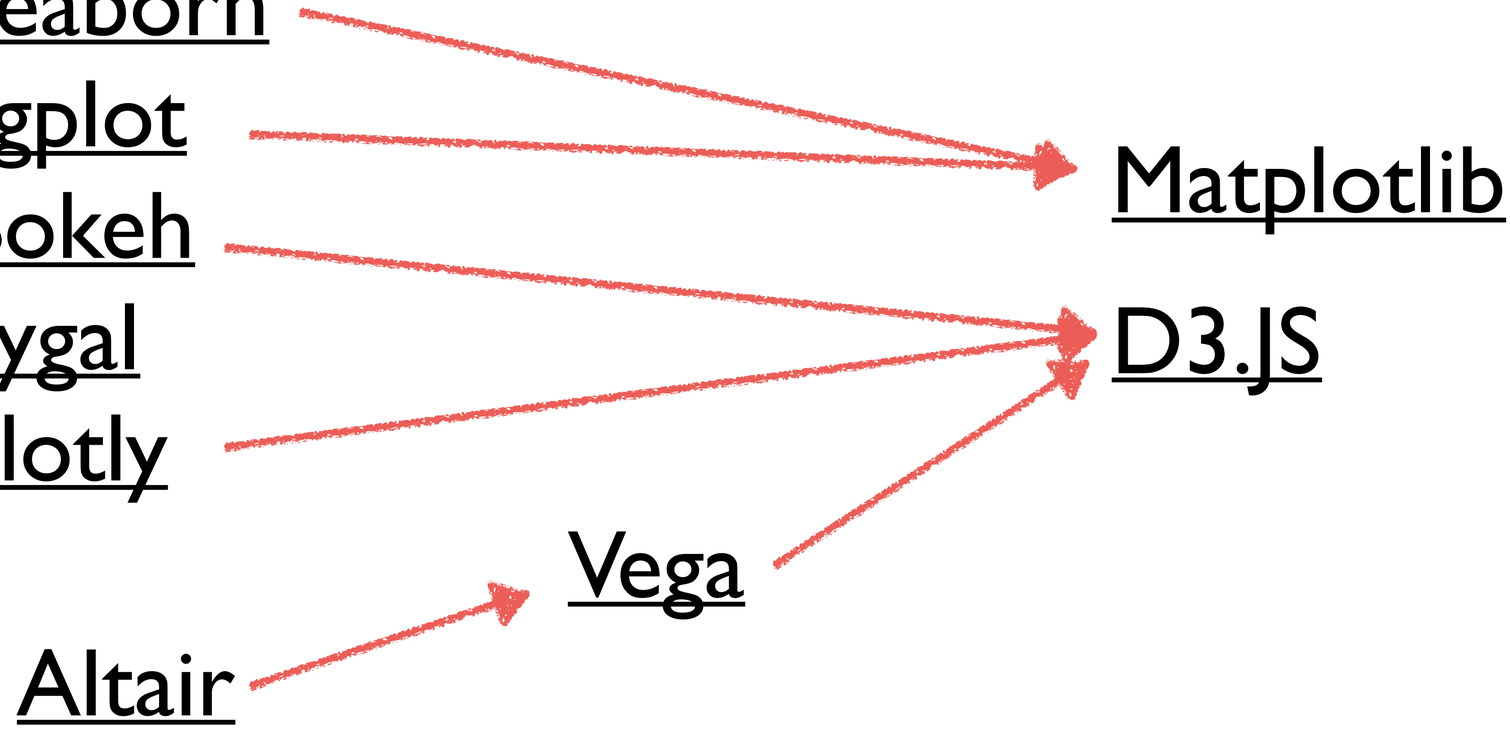
asyncio.Queue() # асинхронная очередь
asyncio.sleep(10) # асинхронный "сон"
asyncio.create_subprocess_exec() # асинхронный subprocess
asyncio.Lock() # асинхронный мьютекс
asyncio.ensure_future() # ручное добавление корутины в event loop
asyncio.gather() # дождаться окончания работы списка корутин

https://xakep.ru/2017/01/11/python-3-asyncio/

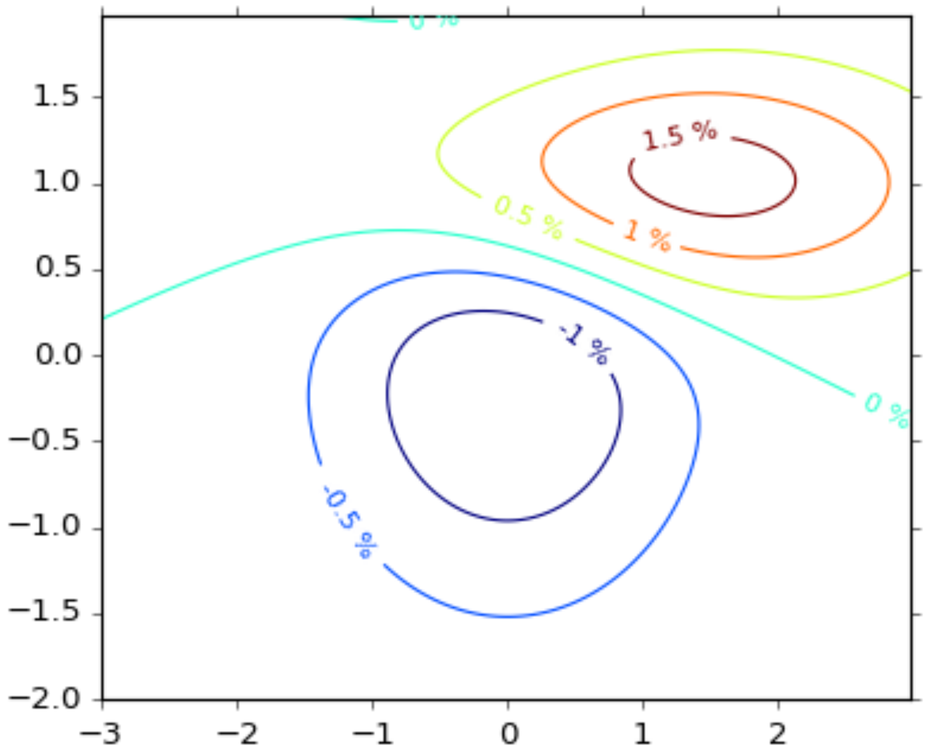
Смотрим на данные

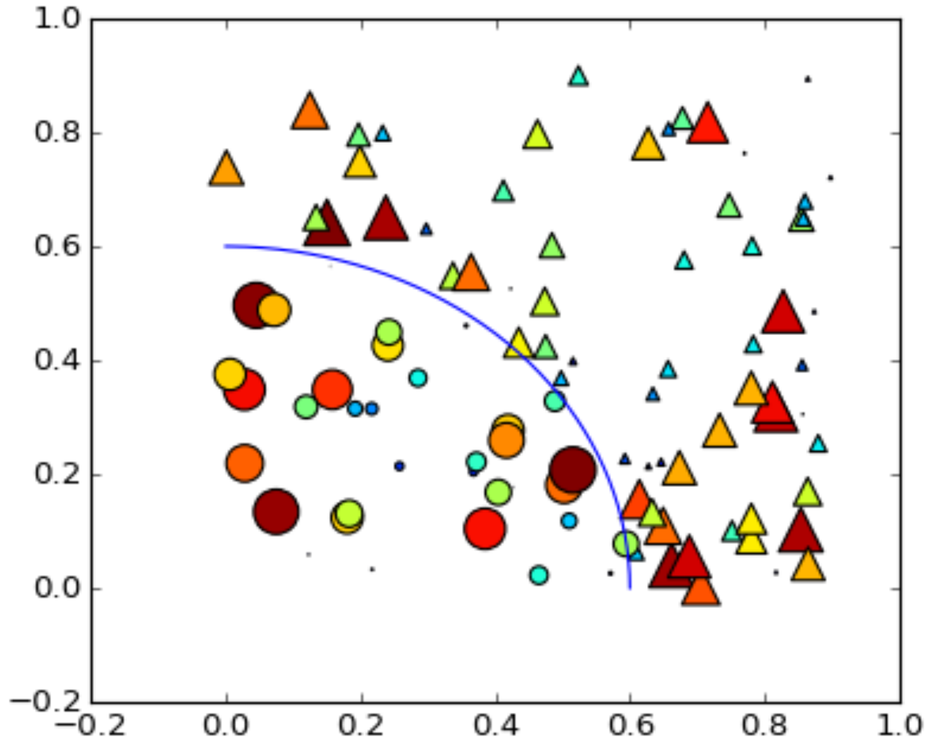
Визуализация

- Seaborn
- ggplot
- Bokeh
- pygal
- Plotly



Просто изображения

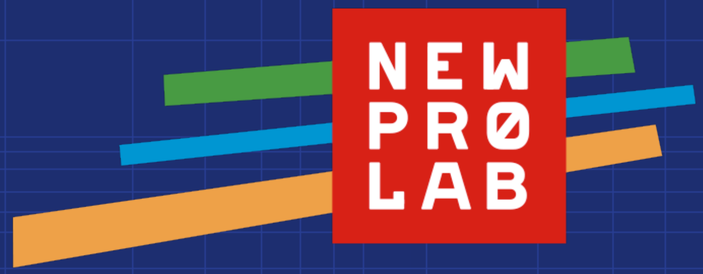




Как оно работает

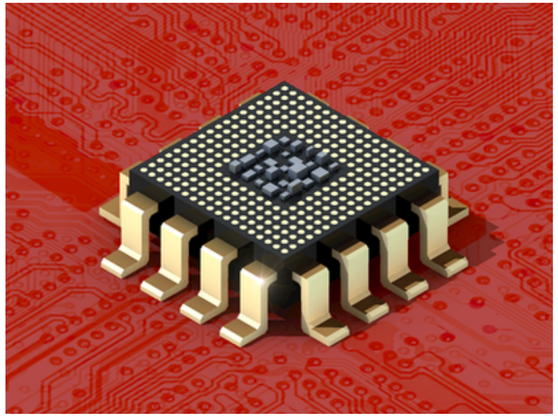
```
%matplotlib inline  # интеграция в Jupyter


# основной объект
from matplotlib import pyplot as plt
```

https://matplotlib.org/gallery/index.html



Numpy, BLAS, LAPACK

Фортран и процессоры

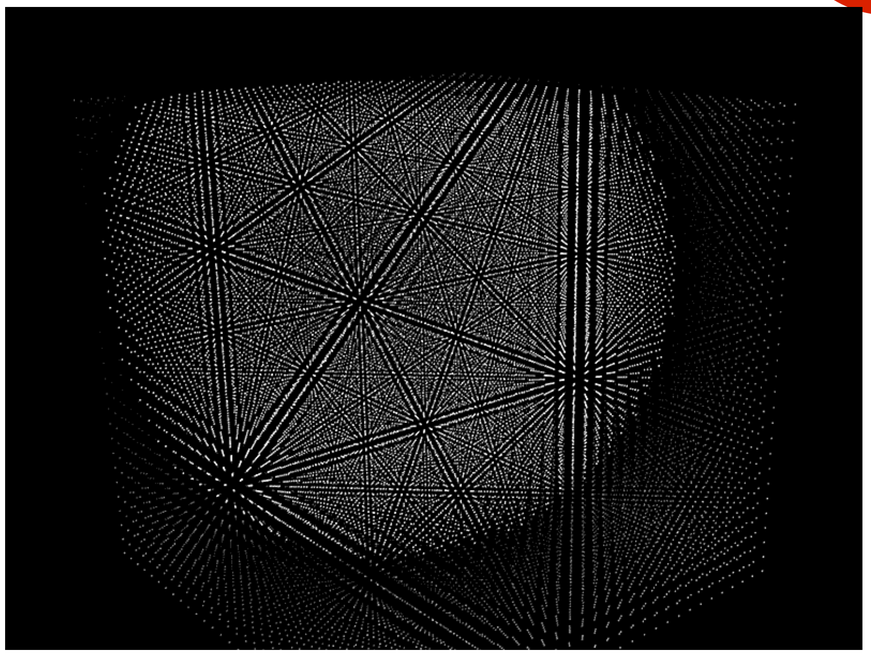


SSE, SSE2, SSE3 …
Векторные операции

Суровые дяди-ученые с 80 годов пишут библиотеку, которая реализует основные операции линейной алгебры
BLAS, LAPACK, CUDA

Матричные данные

Matlab (Octave), R, Julia, Wolfram, …
Numpy



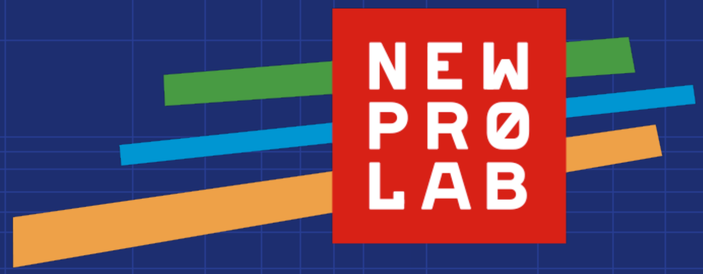
http://www.labri.fr/perso/nrougier/teaching/numpy.100/

Простой синтаксис

```
np.zeros(10)  # вектор из 10 нулевых элементов

#  матрица 3x3 с числами от 0 до 8
np.arange(9).reshape(3,3)
np.eye(3)  # единичная матрица

# трехмерная матрица 3x3x3 со случайными
# значениями
np.random.random((3,3,3))
```

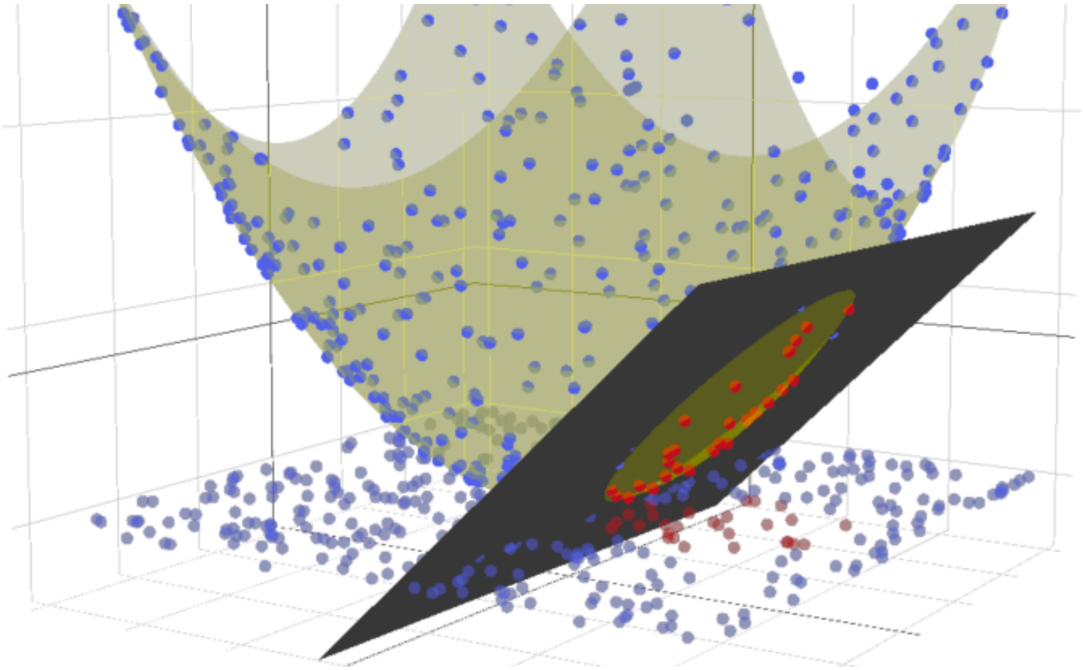
Простой синтаксис

```
# матрица 8x8 - шахматная доска
Z = np.zeros((8, 8), dtype=int)
Z[1::2, ::2] = 1
Z[::2,1::2] = 1

# она же
np.tile(np.array([[0, 1], [1, 0]]), (4, 4))
```



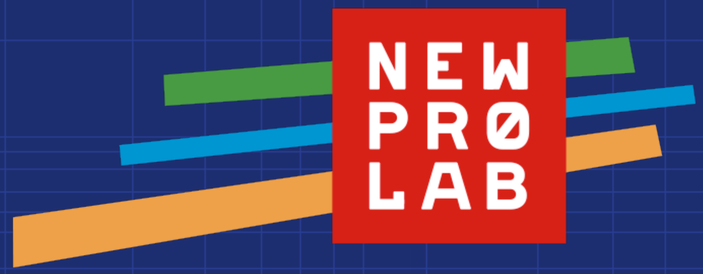
Scikit-learn

Самый лучший



http://scikit-learn.org/stable/

http://www.scipy-lectures.org/advanced/scikit-learn/



Pandas

Все любят Excel

Name	Thread pitch (mm)	Minor diameter tolerance	Nominal diameter (mm)	Head shape	Price for 50 screws	Available at factory outlet?	Number in stock	Flat or Phillips head?
M4	0.7	4g	4	Pan	$10.08	Yes	276	Flat
M5	0.8	4g	5	Round	$13.89	Yes	183	Both
M6	1	5g	6	Button	$10.42	Yes	1043	Flat
M8	1.25	5g	8	Pan	$11.98	No	298	Phillips
M10	1.5	6g	10	Round	$16.74	Yes	488	Phillips
M12	1.75	7g	12	Pan	$18.26	No	998	Flat
M14	2	7g	14	Round	$21.19	No	235	Phillips
M16	2	8g	16	Button	$23.57	Yes	292	Both
M18	2.1	8g	18	Button	$25.87	No	664	Both
M20	2.4	8g	20	Pan	$29.09	Yes	486	Both
M24	2.55	9g	24	Round	$33.01	Yes	982	Phillips
M28	2.7	10g	28	Button	$35.66	No	1067	Phillips
M36	3.2	12g	36	Pan	$41.32	No	434	Both
M50	4.5	15g	50	Pan	$44.72	No	740	Flat

DataFrame

- Схожий интерфейс с R (для олдскульных аналитиков)
- Бесшовная интеграция с Numpy и Matplotlib
- Легкое добавление колонок («фич»)
- Удобные механизмы для заполнения пробелов в данных

http://pandas.pydata.org/

Основные вещи

```
# чтение CSV
df = pd.read_csv("test_data.csv")
df.columns  # названия колонок
# просмотр верхних 15 строчек
df.head(15)
# применение функции к колонке
df["column"].apply(lambda c: c / 100.0)
# средняя зарплата по городу
df.groupby("city").mean("salary")
```

Еще основные вещи

```
# выбор нескольких колонок
df[["Column 1", "Column 2"]]
# создать новую колонку
df["Column"] = a  # массив подходящего размера
# переименование колонки
df.rename(
    columns={"Oldname": "Newname"},
    inplace=True
)
```

DataFrame’ы и генераторы

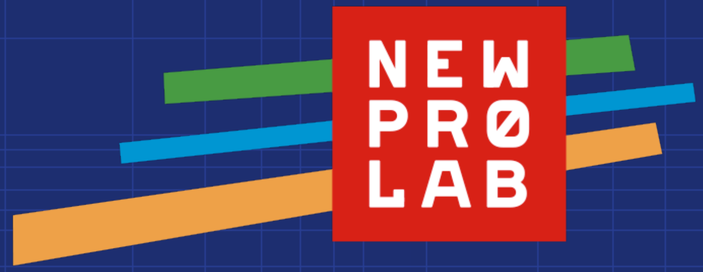
```
# создаем dataframe из генератора
df = pd.DataFrame.from_records((r for r in
some_gen))

# «кусочное» чтение - chunk by chunk
reader = pd.read_csv('tmp.csv', sep='|', chunksize=100)
for chunk in reader:
    do_something(chunk)
```

Упражнение

1. Прочитать датасет по ссылке в Pandas
2. Преобразовать колонку с именами населенных пунктов (Location) к нижнему регистру и добавить еще одну колонку, где будут числа - количество букв «а» в имени пункта
3. Посчитать сумму значений из колонки на предыдущем шаге для каждого Type

единичка

http://bit.ly/1RErAg6



Анализ естественного языка

Токенизация

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday
morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
```

Gensim Word2Vec

```
>>> model = Word2Vec(sentences, size=100, window=5, min_count=5, workers=4)
>>> model.wv['computer']  # numpy вектор
array([-0.00449447, -0.00310097,  0.02421786, ...], dtype=float32)
>>> model.wv.most_similar(positive=['woman', 'king'], negative=['man'])
[('queen', 0.50882536), ...]
>>> model.wv.doesnt_match("breakfast cereal dinner lunch".split())
'cereal'
```

https://radimrehurek.com/gensim/models/word2vec.html

А еще

- Dask - out-of-core версия Pandas/Numpy
- PyMC3 - Markov Chain Monte Carlo
- pymorphy2 + BigARTM - анализ текстов
- Tensorflow + MXNet + Keras + Theano + PyTorch - нейронные сети и deep learning