Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики

«Наглядный вероятностно-статистический анализ данных»

Практическое занятие 1

«Библиотеки данных для анализа. Начало работы с Python. Чтение и обработка Excel файлов средствами Python»

Пройдакова Екатерина Вадимовна,
доцент кафедры ТВиАД ИИТММ

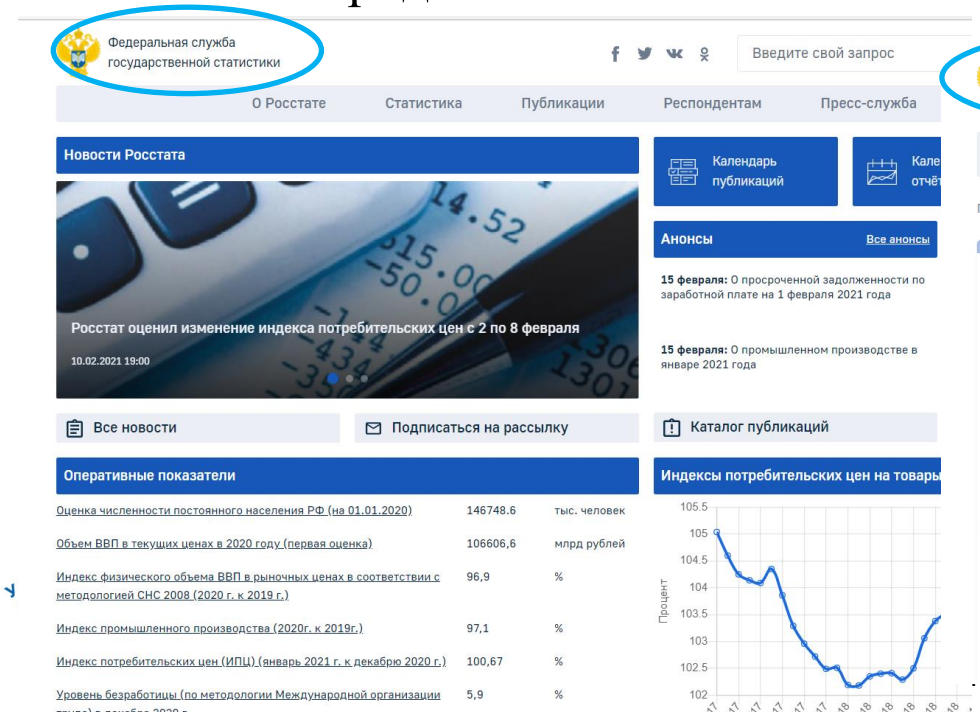
Содержание

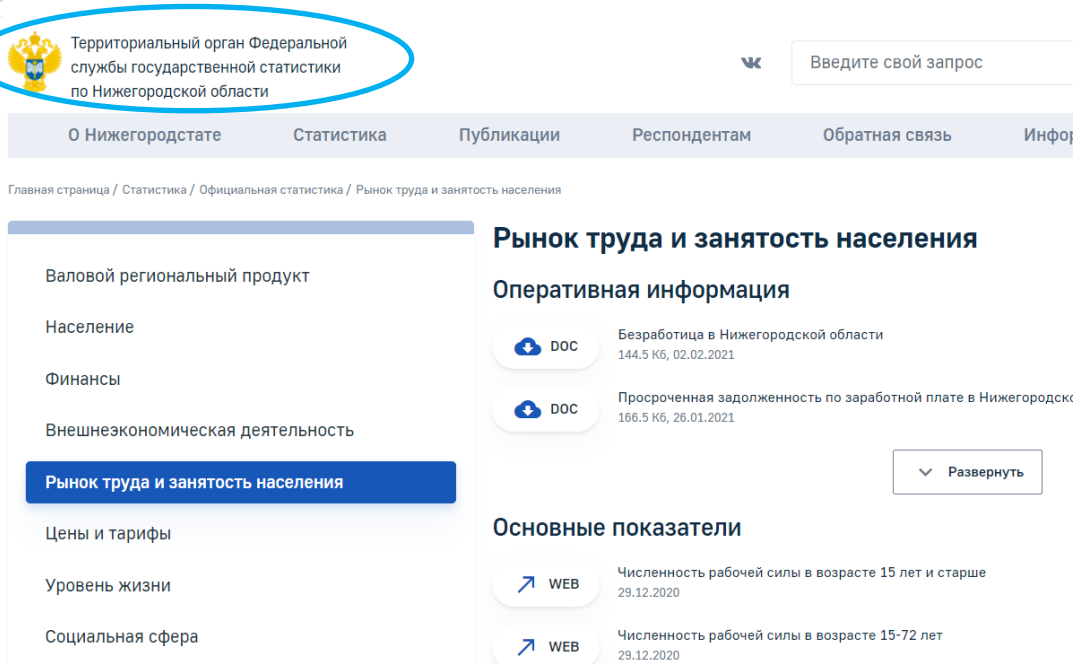
- Открытые библиотеки данных
- Python: начало работы
  - Среда разработки
  - Библиотеки для статистического анализа
- Пример: загрузка данных в Python и простейший анализ
- Изучаемые средства Python
- Практическое задание

1. ОТКРЫТЫЕ БИБЛИОТЕКИ ДАННЫХ

1. Открытые библиотеки данных

- Государственные службы статистики представляют большое количество данных из различных сфер: демография, экономика и финансы, рынок труда, образование, наука, окружающая среда и т.п.
  - https://www.gks.ru/ и https://nizhstat.gks.ru/ – сайты Федеральной службы государственной статистики (Росстата) и ее Территориального органа по Нижегородской области. Основные форматы данных: .xls, .doc, .html





1. Открытые библиотеки данных

- https://www.usa.gov/statistics – сайт правительства США с ссылками на федеральные агентства по сбору данных (Бюро экономического анализа, Бюро статистики по транспорту, Национальный центр статистики по образованию и т.д.). Основные форматы данных: .csv, .xls, .zip, .doc, .html
- https://data.gov.uk/ – сайт поиска открытых данных правительства Великобритании. Основные форматы данных: .csv, .xls, .zip, .doc, .html
- http://data.un.org/ – статистические данные ООН. Форматы данных: .pdf, .csv.

1. Открытые библиотеки данных

Существует огромное количество открытых наборов данных для отладки алгоритмов машинного обучения. Как правило, такие данные являются «сырыми»: файлы с данными содержат лишь сами данные, без заголовков и пояснений. Примеры:

– https://www.kaggle.com/datasets – платформа онлайн-обучения и соревновательного решения статистических задач. Основные форматы данных: .csv, .json

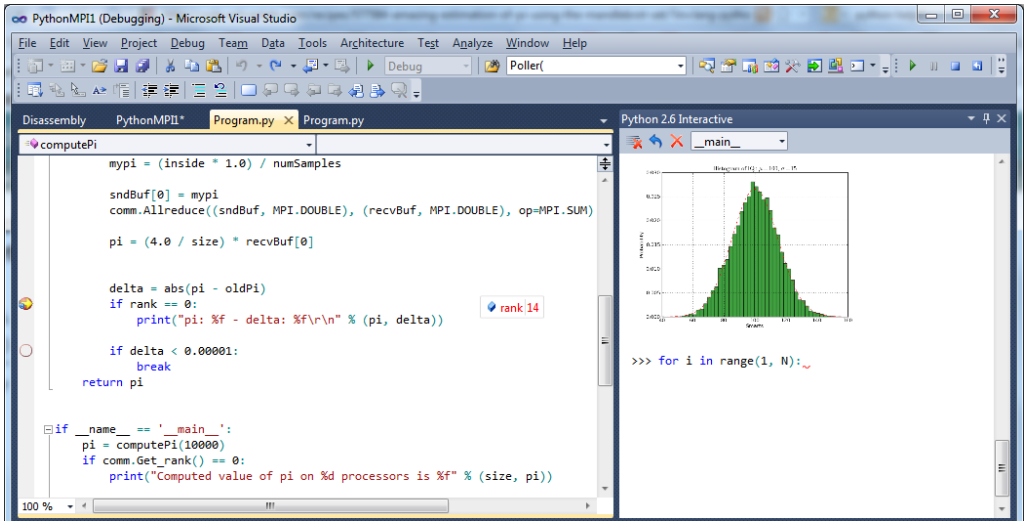
– http://mldata.org/ – репозиторий с данными для отладки алгоритмов машинного обучения. Основные форматы данных: .csv, .xml, .matlab

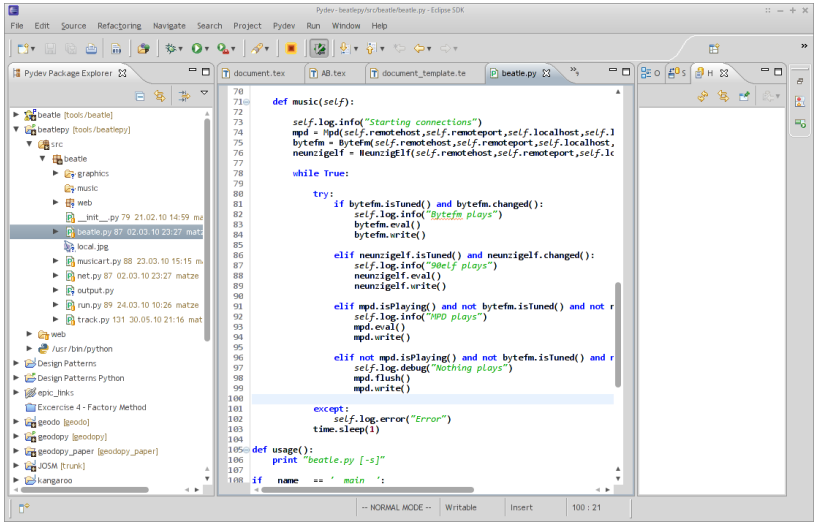
– http://www.causality.inf.ethz.ch/repository.php – сайт проекта, представляющего наборы данных из разных областей науки и финансируемого Национальным Научным Фондом США. Основные форматы данных: .csv, .txt, .xls, .matlab

2. PYTHON: НАЧАЛО РАБОТЫ

2. Python: начало работы
2.1. Среда разработки

- Для тех, кто уже использует известные многофункциональные среды разработки, наиболее доступно установить расширения с поддержкой Python. Например,
  - Visual Studio + Python Tools: бесплатный вариант в Community Edition.
  - Eclipse + PyDev: open-source IDE.

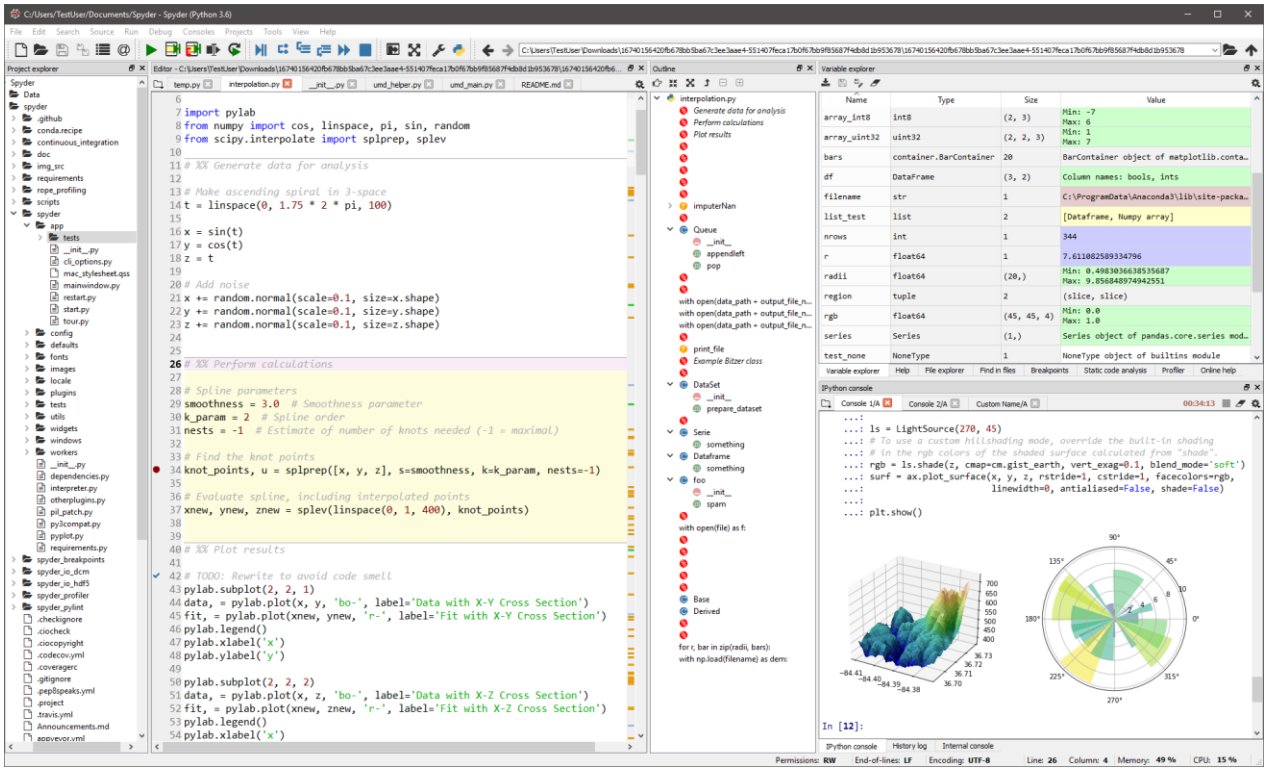




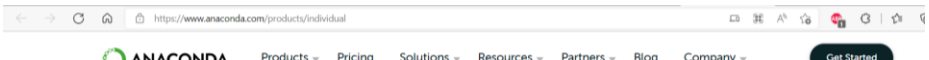
2. Python: начало работы
2.1. Среда разработки

- Альтернатива – среды для разработки на Python. Одним из вариантов для работы в рамках анализа данных (Data Science) является open-source **среда Spyder**.

Spyder IDE включена в состав удобного для установки и дальнейшей работы менеджера пакетов **Anaconda**.



2. Python: начало работы
2.1. Среда разработки



[Anaconda | Individual Edition]

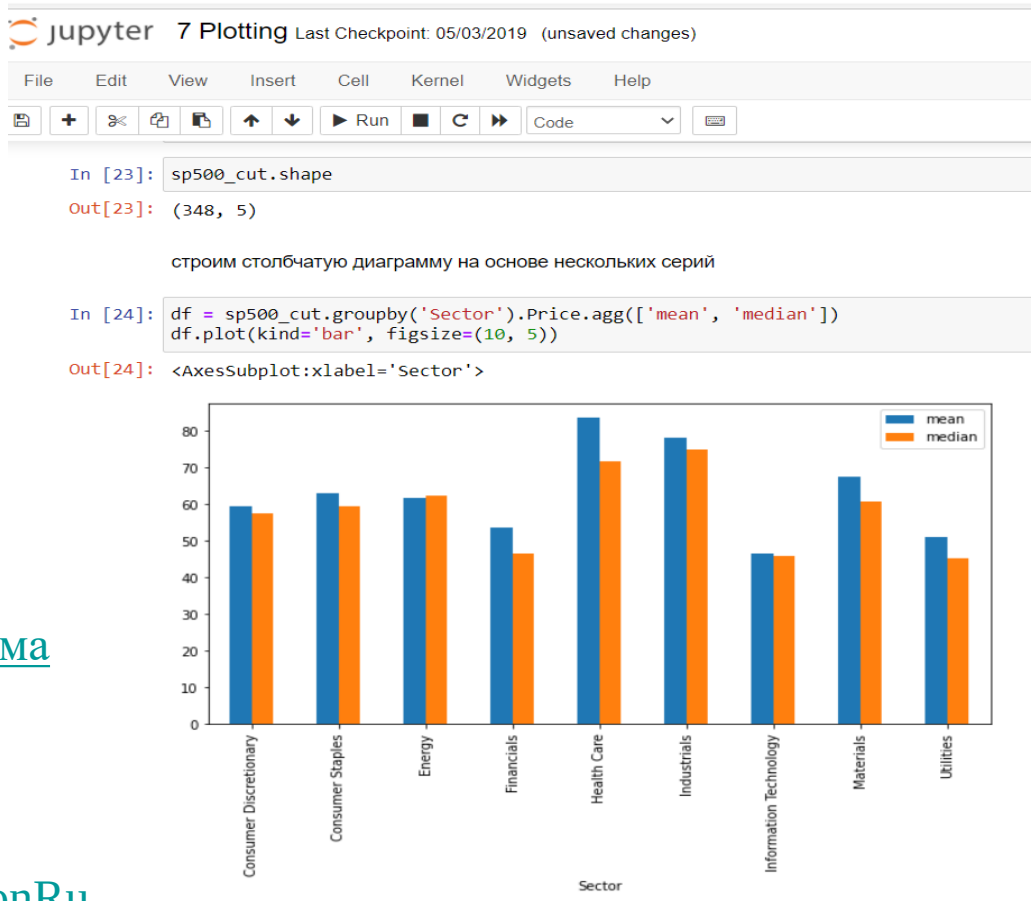
https://www.anaconda.com/products/individual

[Знакомство с Anaconda: что это такое и как установить | by Olga Sayfudinova | NOP::Nuances of Programming | Medium]

https://medium.com/nuances-of-programming/%D0%B7%D0%BD%D0%B0%D0%BA%D0%BE%D0%BC%D1%81%D1%82%D0%B2%D0%BE-%D1%81-anaconda-%D1%87%D1%82%D0%BE-%D1%8D%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-%D0%B8-%D0%BA%D0%B0%D0%BA-%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%B8%D1%82%D1%8C-2c19b3e8226

2. Python: начало работы
2.1. Среда разработки

Jupyter Notebook (ранее IPython Notebook) - это веб-интерактивная вычислительная среда для создания документов Jupyter notebook (имеет расширение ***.ipynb**)



Какие языки поддерживаются **Jupyter Notebook:** Python, Ruby, Perl, R.

Что такое jupyter-ноутбук и зачем он нужен - Журнал «Код» программирование без снобизма (thecode.media)

https://thecode.media/jupyter/

Введение в работу с Jupyter Notebook ~ PythonRu

https://pythonru.com/baza-znanij/jupyter-notebook-dlja-nachinajushhih

2. Python: начало работы
2.1. Среда разработки

Полезные ссылки:

1. Самоучитель Python

https://pythonworld.ru/samouchitel-python

2. Документация Python

Справочник по языку Python:

https://docs.python.org/3/reference/index.html

Стандартная библиотека Python:

https://docs.python.org/3/library/index.html

Список документации Python:

https://docs.python.org/3/

3. Несколько популярных форумов, посвященных Python и программированию вообще

python-forum.io

https://www.dreamincode.net/forums/forum/29-python/

StackOverflow.com

4. Поддержка Jupyter Notebook предоставляется на следующих ресурсах

Jupyter-чат в реальном времени:

https://gitter.im/jupyter/jupyter

GitHub

https://github.com/jupyter/help

StackOverflow:

https://stackoverflow.com/questions/tagged/jupyter

3. PYTHON: БИБЛИОТЕКИ ДЛЯ СТАТИСТИЧЕСКОГО АНАЛИЗА

3. Python: Библиотеки для статистического анализа

- Основные возможности **разработки на Python** связаны с большим количеством библиотек (модулей). Подключение библиотек на примерах:

- **`import`** `random` – подключение модуля `random` целиком
- **`import`** `re` **`as`** `regular` – подключение модуля `re` с выбором для него псевдонима `regular` при дальнейшем использовании
- **`import`** `matplotlib.pyplot` **`as`** `plt` – подключение отдельной части `pyplot` модуля `matplotlib` с назначением псевдонима `plt`
- **`from`** `numpy.random` **`import`** `normal,rand` – подключение отдельных частей `normal` и `rand` из подмодуля `numpy.random`

3. Python: Библиотеки для статистического анализа

1. **Библиотека `Matplotlib`** предоставляет следующие возможности:
   a. Графический анализ данных
   b. Основные возможности: построение линейных графиков, различного вида диаграмм (столбчатых, точечных, круговых, спектральных)
   c. Модуль `pyplot` – интерфейс, подобный Mathlab
   d. Большое количество поддерживаемых форматов изображений: PNG, JPEG, PDF и др.
   e. Высокоуровневая надстройка `seaborn`: графический анализ данных.

3. Python: Библиотеки для статистического анализа

2. Библиотека `NumPy`

a. Многомерные массивы и матрицы, функции работы с ними

b. Поверх данной библиотеки – библиотека `SciPy` с более широкой функциональностью, библиотека `pandas`

3. Библиотека `SciPy`

a. Модуль `linalg` - методы линейной алгебры: решение СЛУ, поиск собственных векторов и значений, разложений матриц

b. Модуль `integrate` - интегральное исчисление

c. Модуль `stats` – теория вероятностей и математическая статистика: случайные величины, их распределения вероятностей, статистические числовые характеристики, проверка гипотез, подсчет статистик и др.

d. Модуль `fftpack` – преобразование Фурье

3. Python: Библиотеки для статистического анализа

4. Библиотека `Pandas`:

a. Сбор, очистка и загрузка данных

b. Специальные структуры данных, переформатирование данных, сводные таблицы

c. Анализ данных: группировка, агрегирование, фильтрация

5. Библиотека `Scikit-learn`:

a. Основана на `NumPy`, `SciPy`, `Matpotlib`, `Pandas`

b. Алгоритмы анализа данных и машинного обучения: регрессия, кластеризация, классификация, понижение размерности, детектирование аномалий, выделение признаков

c. Большое количество методов анализа данных: наивный Байес, метод K-средних, K ближайших соседей, нейронные сети, деревья решений и др.

4. ПРИМЕР: ЗАГРУЗКА ДАННЫХ В PYTHON И ПРОСТЕЙШИЙ АНАЛИЗ

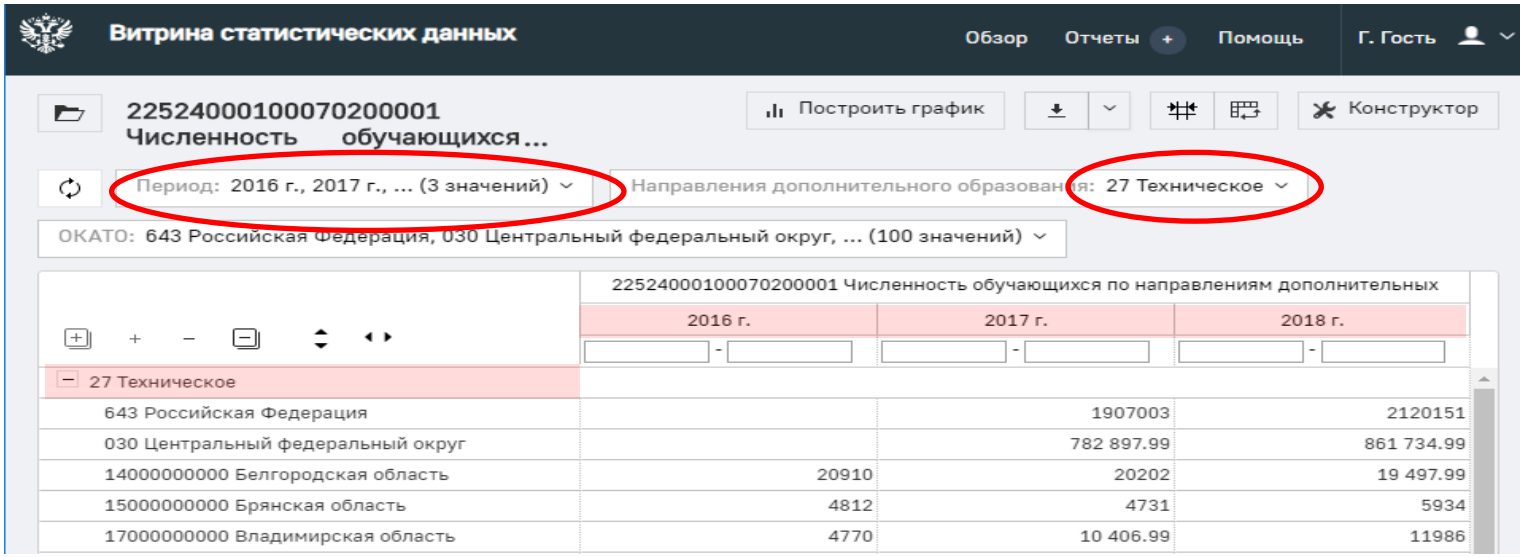
4. Пример: загрузка данных в Python и простейший анализ

- Имеется отчет по численности обучающихся ДО программ по субъектам РФ (данные Росстата).

  1. Для каждого субъекта РФ определить год, в котором обучающихся было максимальное число.

  2. Для каждого года определить число субъектов РФ, в которых именно в этом году число обучающихся было максимальным.

  3. Отсортировать субъекты РФ по возрастанию среднего количества обучающихся за 3 года.

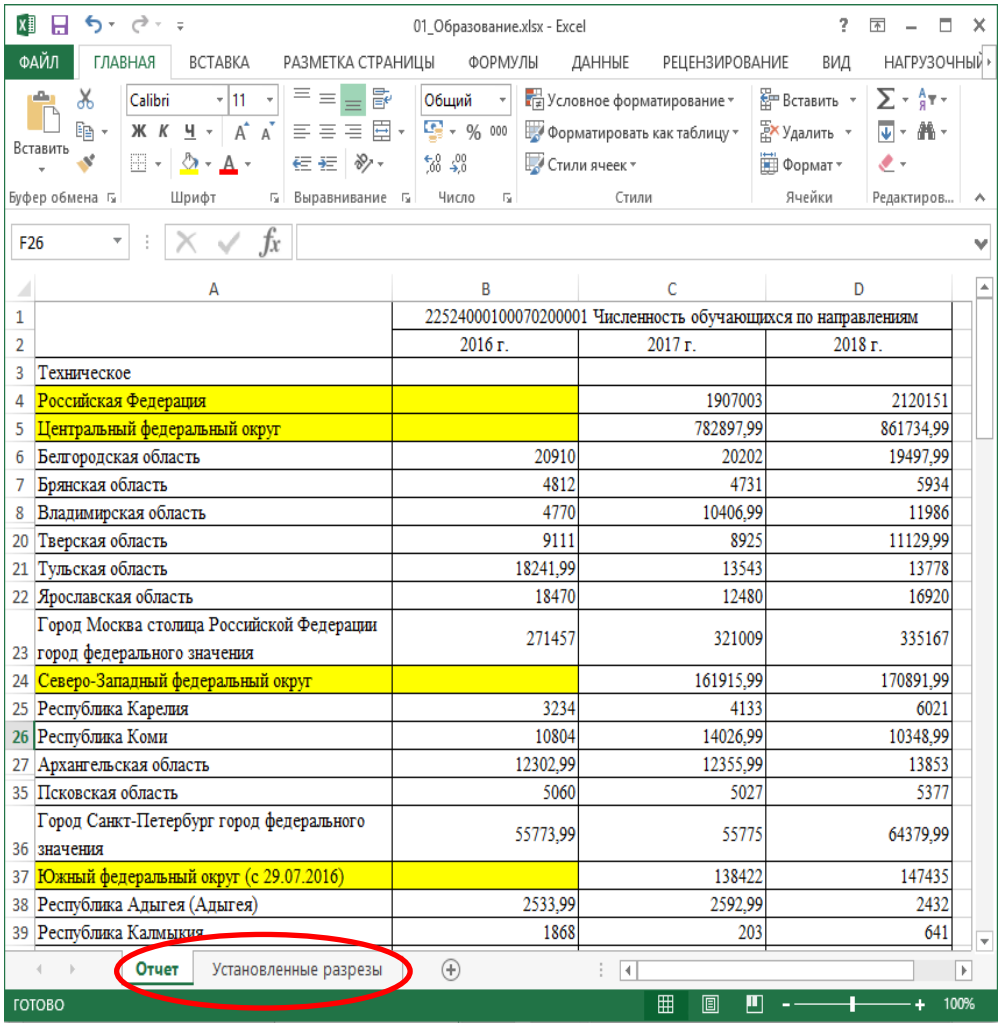


22524000100070200001 Численность обучающихся по направлениям дополнительных

	2016 г.	2017 г.	2018 г.
27 Техническое			
643 Российская Федерация		1907003	2120151
030 Центральный федеральный округ		782 897.99	861 734.99
14000000000 Белгородская область	20910	20202	19 497.99
15000000000 Брянская область	4812	4731	5934
17000000000 Владимирская область	4770	10 406.99	11986

4. Пример: загрузка данных в Python и простейший анализ

- Скачиваем документ 01_Образование.xlsx.
  - В этом документе 2 листа. Основной из них, лист с данными – «Отчет».
  - Не все строки таблицы с данными равноправны. Выделенные на рисунке строчки аккумулируют информацию по нескольким субъектам РФ.



	A	B	C	D
1		22524000100070200001 Численность обучающихся по направлениям		
2		2016 г.	2017 г.	2018 г.
3	Техническое			
4	Российская Федерация		1907003	2120151
5	Центральный федеральный округ		782897,99	861734,99
6	Белгородская область	20910	20202	19497,99
7	Брянская область	4812	4731	5934
8	Владимирская область	4770	10406,99	11986
20	Тверская область	9111	8925	11129,99
21	Тульская область	18241,99	13543	13778
22	Ярославская область	18470	12480	16920
23	Город Москва столица Российской Федерации город федерального значения	271457	321009	335167
24	Северо-Западный федеральный округ		161915,99	170891,99
25	Республика Карелия	3234	4133	6021
26	Республика Коми	10804	14026,99	10348,99
27	Архангельская область	12302,99	12355,99	13853
35	Псковская область	5060	5027	5377
36	Город Санкт-Петербург город федерального значения	55773,99	55775	64379,99
37	Южный федеральный округ (с 29.07.2016)		138422	147435
38	Республика Адыгея (Адыгея)	2533,99	2592,99	2432
39	Республика Калмыкия	1868	203	641

4. Пример: загрузка данных в Python и простейший анализ

- Считываем документ в Python с использованием библиотеки `xlrd`.

```
1
2  import xlrd
3
4  #указывается полный путь до файла
5  book = xlrd.open_workbook('../01_Образование.xlsx')
6  #извлекаем лист с данными по имени
7  sheet = book.sheet_by_name('Отчет')
8
9  num_rows = sheet.nrows
10 num_cols = sheet.ncols
11 #вывод на экран информации о документе
12 print('Количество листов = ', book.nsheets, '\nКоличество строк = ',
13       num_rows, '\nКоличество столбцов = ', num_cols)
14
```

4. Пример: загрузка данных в Python и простейший анализ

- Первая из поставленных выше задач решается следующим образом:

```
15
16 import numpy as np
17
18 regionCount = [] #создание пустого списка
19
20 #заполнение списка regionCount,
21 #i-тый элемент которого regionCount[i] является
22 #списком количеств обучащихся по годам для i-той по счету области
23
24 for i in range(num_rows - 4): #первые 4 строки заведомо не информативны
25     #если данная строка не характеризует округ
26     if (sheet.cell(i + 4, 0).value.find('федеральный округ') == -1):
27         #добавляем список количеств обучащихся по годам
28         regionCount.append(sheet.row_values(i + 4, 1, 4))
29
30 #yearMax - список годов, в которых  количество обучающихся было максимальным
31 yearMax = 2016 + np.argmax(regionCount, axis = 1)
32
```

- В результате имеем:

```
In [202]: print(yearMax)
[2016 2018 2018 2018 2018 2018 2018 2018 2017 2018 2018 2018 2018 2017
 2018 2016 2016 2018 2018 2017 2018 2017 2018 2018 2016 2017 2017 2016
 2018 2018 2017 2016 2018 2018 2018 2017 2016 2016 2018 2017 2018 2018
 2018 2017 2016 2018 2016 2017 2017 2017 2018 2018 2018 2018 2018 2018
 2018 2017 2018 2018 2018 2018 2018 2016 2018 2018 2016 2018 2018 2018
 2018 2018 2018 2017 2018 2018 2017 2018 2018 2018 2018 2018 2018 2016
 2018 2018 2018]
```

4. Пример: загрузка данных в Python и простейший анализ

- Вторая задача наиболее просто решается с помощью еще одной важной структуры данных – словарь (dictionary, ассоциативный список). Особенностью словарей является возможность обращения к их элементам по ключу.

```
32
33 from collections import Counter
34 #yearCounter - словарь, считающий количество вхождений каждого из годов
35 #в список yearMax
36 yearCounter = Counter(yearMax)
37 print(yearCounter)
38
```

- В строке 33 из библиотеки `collections` специальных типов данных подключается подкласс словарей Counter, где в качестве значения каждого ключа выступает его абсолютная частота.

```
Counter({2018: 58, 2017: 16, 2016: 13})
```

4. Пример: загрузка данных в Python и простейший анализ

- Перейдем к третьей задаче - сортировки субъектов РФ по возрастанию среднего количества обучающихся за 3 года.

```
40 regionMean = {} #создание пустого словаря
41
42 #заполнение словаря regionMean, ключом в котором является название области
43 #а значением - среднее количество обучающихся за 3 года
44
45 for i in range(num_rows - 4): #первые 4 строки заведомо не информативны
46     #если данная строка не характеризует округ
47     if (sheet.cell(i + 4, 0).value.find('федеральный округ') == -1):
48         #добавляем в словарь среднее количество обучающихся
49         regionMean[sheet.cell(i + 4, 0).value] = np.mean(sheet.row_values(i + 4, 1, 4))
50
51 #определяем функцию, которая для пары (x, y) возвращает второе значение
52 def secondInPair(x):
53     #раскладываем входой параметр на пару
54     (a, b) = x
55     #возвращаем второе значение
56     return b
57
58 #сортируем елементы regionMean, используя в качестве ключа к сортировке
59 #функцию secondInPair
60 regionSorted = sorted(regionMean.items(), key = secondInPair)
61 print(regionSorted)
```

4. Пример: загрузка данных в Python и простейший анализ

```
In [162]: regionMean
Out[162]:
{'Белгородская область': 20203.33,
 'Брянская область': 5159.0,
 'Владимирская область': 9054.33,
 'Воронежская область': 21204.329999999998,
 'Ивановская область': 7497.0,
 'Калужская область': 7002.666666666667,
 'Костромская область': 7289.0,
 'Курская область': 6277.666666666667,
```

```
In [161]: regionMean.items()
Out[161]: dict_items([('Белгородская область', 20203.33), ('Брянская
область', 5159.0), ('Владимирская область', 9054.33), ('Воронежская
область', 21204.329999999998), ('Ивановская область', 7497.0), ('Калужская
область', 7002.666666666667), ('Костромская область', 7289.0), ('Курская
область', 6277.666666666667), ('Липецкая область', 8660.996666666666),
('Московская область', 265757.66333333333), ('Орловская область',
4043.6633333333334), ('Рязанская область', 8011.993333333333), ('Смоленская
область', 5800.996666666666), ('Тамбовская область', 7768.329999999999),
('Тверская область', 9721.996666666666), ('Тульская область',
```

```
In [205]: regionSorted
Out[205]:
[('Ненецкий автономный округ (Архангельская область)', 408.6666666666667),
 ('Чукотский автономный округ', 432.3333333333333),
 ('Республика Калмыкия', 904.0),
 ('Еврейская автономная область', 904.6666666666666),
 ('Магаданская область', 1199.6666666666667),
 ('Республика Алтай', 2250.3333333333335),
 ('Сахалинская область', 2367.3333333333335),
 ('Республика Адыгея (Адыгея)', 2519.66),
 ('Город федерального значения Севастополь', 2632.0),
 ('Карачаево-Черкесская Республика', 3319.3333333333335),
 ('Камчатский край', 3488.3333333333335),
 ('Республика Северная Осетия-Алания', 4025.33),
```

5. ИСПОЛЬЗУЕМЫЕ СРЕДСТВА PYTHON

5. Изучаемые средства Python

- Библиотека `xlrd` предоставляет средства для чтения и форматирования данных из Excel файлов. Основные библиотечные классы: «книга» `xlrd.Book`, «лист» `xlrd.Sheet` и «ячейка» `xlrd.Cell`.

```
import xlrd

# Библиотечная функция: возвращает объект класса "книга" - xlrd.Book
book = xlrd.open_workbook('01_Образование.xlsx')
book.nsheets # количество листов в книге = 2
book.sheet_names() # список названий листов в книге = ['Отчет', 'Установленные разрезы']

sheet = book.sheet_by_name('Отчет') # возвращает объект класса "лист" - xlrd.Sheet - по имени листа
sheet = book.sheet_by_index(0) # возвращает объект класса "лист" - xlrd.Sheet - по номеру листа
sheet.nrows # количество строк
sheet.ncols # количество столбцов

cell = sheet.cell(5, 1) # возвращает объект класса «ячейка» - xlrd.Cell - по номеру строки и столбца
# = number:20910.0
# ячейка хранит информацию о типе значения и самом значении
cell.value # возвращает ЗНАЧЕНИЕ ячейки
cell.ctype # возвращает номер типа значения ячейки: пустая строка - 0, строка - 1, число  - 2, дата - 3 и т.д.
sheet.cell_value(5, 1) # возвращает ЗНАЧЕНИЕ ячейки с заданными номером строки и столбца

col = sheet.col(1) # возвращает список ячеек столбца с заданным номером
part_of_col = sheet.col_slice(1, 3, 7) # возвращает список ячеек столбца с номером 1 с номерами строк от 3 до 7 (не включительно)
# = [empty:'', empty:'', number:20910.0, number:4812.0]
part_of_col_values = sheet.col_values(1, 3, 7) # в отличие от предыдущего возвращает список ЗНАЧЕНИЙ ячеек столбца
# = ['', '', 20910.0, 4812.0]
```

5. Изучаемые средства Python

- Использованный в примере словарь (dictionary) – это структура данных, в которой значения связаны с ключами.

```
34
35 years = {"Bob": 16, "Dick": 21, "Dave": 17, "Anna": 16, "Clark": 20}
36 print(years["Anna"])  # = 16
37
38 years["John"] = 25  # добавляем новую запись
39 years["Dick"] = 22  # переписываем старую запись
40 num_people = len(years) # количество записей = 6
41 print(years["Daniel"]) # ошибка, записи с таким ключом не существует. Для безопасного доступа использовать метод get(...)
42
43 sorted(years) # = ['Anna', 'Bob', 'Clark', 'Dave', 'Dick', 'John']
44 # сотировка словаре происходит по ключу
45 years.values() # = [16, 22, 17, 16, 20, 25] список значений
46 years.keys() # = ['Bob', 'Dick', 'Dave', 'Anna', 'Clark', 'John'] список ключей
47 years.items() # = [('Bob', 16), ('Dick', 22), ('Dave', 17), ('Anna', 16), ('Clark', 20), ('John', 25)]
48 # список кортежей (упорядоченных пар)
49 years.pop("Anna") # = 16 удаляет запись и возвращает соответствующее значение
50 years.popitem() # = ('John', 25) удаляет последнюю запись и возвращает соответствующий ей кортеж (ключ, значение)
51
```

5. Изучаемые средства Python

- Один из часто употребляемых при статистическом анализе подклассов словарей – словарь `Counter` из библиотеки `collections`. Особенность таких словарей состоит в том, что каждому ключу в соответствие ставится его абсолютная частота встречаемости.

```
51
52 sample = [18, 19, 23, 25, 16, 19, 21, 20, 18, 24, 17, 19, 19, 20, 18, 21, 22, 21]
53
54 from collections import Counter
55
56 years_count = Counter(sample) # преобразовываем список в словарь частот
57 # = Counter({19: 4, 18: 3, 21: 3, 20: 2, 23: 1, 25: 1, 16: 1, 24: 1, 17: 1, 22: 1})
58 years_count.most_common(4) # 4 элемента с наибольшей частотой повторения
59 # = [(19, 4), (18, 3), (21, 3), (20, 2)]
60 sorted(years_count.elements()) # отсортированные  элементы словаря с повторением
61 # = [16, 17, 18, 18, 18, 19, 19, 19, 19, 20, 20, 21, 21, 21, 22, 23, 24, 25]
62 len(years_count) # количество записей словаря  = 10
```

ПРАКТИЧЕСКОЕ ЗАДАНИЕ №1

Практическое задание №1

- Ознакомиться с основными рассмотренными открытыми хранилищами данных: перейти по ссылке, как минимум, на пару сайтов из каждой категории, изучить, в каком виде предоставляются данные, скачать по образцу файлов, изучить скаченные материалы.
- Установить необходимое ПО для работы с языком Python.
- К заданию прилагается 2 файла с данными о числе организаций, осуществляющих образовательную деятельность по субъектам РФ. В документе *01_Образование_организации_01.xlsx* содержатся данные за 2016, 2017 года, а в документе *01_Образование_организации_02.xlsx* – за 2015, 2018 года.

	A	B	C
1		22524000100120200002 Число организаций, осуществляющих образовательную	
2		2016 г.	2017 г.
3	Российская Федерация	44918	56302
4	Центральный федеральный округ	11438,99	13772,99
5	Белгородская область	906	916
6	Брянская область	607	715
7	Владимирская область	137	488
8	Воронежская область	521	1106
9	Ивановская область	433	437

Практическое задание №1

- Для данных документов необходимо:
  a. Загрузить данные из документов для работы в Python средствами библиотеки `xlrd`.
  b. Составить словарь (dictionary), ключом в котором является название субъекта РФ, а значением – список из четырех элементов: число образовательных организаций в 2015, 2016, 2017, 2018 годах.

Примечания: 1) исключить из рассмотрения сводную информацию по федеральным округам и РФ; 2) в списках к каждому субъекту соблюсти хронологический порядок данных: начиная с 2015го и заканчивая 2018м годом. Пример итоговых записей в словаре:

```
'Белгородская область': [193, 906, 916, 885],
'Брянская область': [163, 607, 715, 727]}
```

Практическое задание №1

- Для данных документов необходимо:
  - c. В полученном словаре для каждого субъекта РФ добавить в список 2 дополнительных элемента: среднее количество образовательных организаций за 4 года и год, в котором было достигнуто максимальное количество образовательных организаций.

Пример итоговых записей в словаре:

```
'Белгородская область': [193, 906, 916, 885, 725, 2017],
'Брянская область': [163, 607, 715, 727, 553, 2018]}
```

Практическое задание №1

- Для данных документов необходимо:

  d. С использованием библиотечного средства Counter составить словарь, в котором ключом является год из диапазона 2015-2018, а значением – количество субъектов РФ, в которых именно в этом году количество образовательных организаций было максимальным.

  e. Отсортировать субъекты РФ по возрастанию среднего числа образовательных организаций за 4 года.

  Примечание: для решения указанной задачи написать функцию, возвращающую пятый элемент списка, использовать встроенную функцию сортировки `sorted`.

Список литературы

1. *Лутц М.* Изучаем Python, 4е издание. – Пер. с англ. – Спб.: Символ-Плюс, 2011. – 1280 с.
2. *Pilgrim M.* Dive into Python. – Apress, 2004.
3. *Доусон М.* Программируем на Python. – Спб.: Питер, 2014. – 416 с.
4. *Васильев А. Н.* Python на примерах. Практический курс по программированию. – СПб.: Наука и Техника, 2016. – 432 с.
5. *Маккинни У.* Python и анализ данных. – Пер. с англ. А.А. Слинкина. – М.: ДМК Пресс, 2015. – 482 с.
6. *Грас Дж.* Data Science. Наука о данных с нуля. Пер. с англ. – Спб.:БХВ-Петербург, 2019. – 336 с.
7. *Плас Дж. Вандер*. Python для сложных задач: наука о данных и машинное обучение. – СПб.: Питер, 2018. – 576 с.