Муниципальное бюджетное общеобразовательное учреждение
«Гимназия №7»

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ
(Python)

Методические указания по подготовке к практическим занятиям, в том числе в интерактивной форме, и самостоятельному изучению дисциплины для школьников 7–11 классов общеобразовательных средних учебных заведений (базовый уровень)

Красноярск 2024

Издаётся по решению учебно–методического совета МБОУ СОШ 7 гимназия.

Коренюгина Л.М.
ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ(Python)**:** методические указания по подготовке к практическим занятиям, в том числе в интерактивной форме, и самостоятельному изучению дисциплины для школьников 7–11 классов общеобразовательных средних учебных заведений (базовый уровень) / Л.М.Коренюгина. – Красноярск: МБОУ СОШ 7 гимназия, 2024. – 103 с.;

Методические указания включают содержание основных тем по дисциплине примеров решения практических заданий и заданий для самостоятельного решения, а также список ссылок на видеоуроки.
Рецензент:

©МБОУ СОШ 7 гимназия, 2024
© МБОУ ДО ЦДО "Аэрокосмическая школа
имени Героя Социалистического Труда Гупалова В.К.", 2024

©Коренюгина Л.М., 2024

Содержание

МОДУЛЬ 1. ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ АНАЛИЗА ДАННЫХ 7
Тема 1.1. Установка Python, IDE, создание модулей. Проверка идиоматичности кода 7
Тема 1.2. Условный оператор if 22
Тема 1. 3. Последовательности и циклы 24
Тема 1. 4. Список как изменяемая последовательность 25
Тема 1. 5. Кортежи, словари и множества 32
Тема 1. 6. Функции 33
Тема 1. 7. Файлы, работа с файловой системой 34
Тема 1. 8. Обработка исключений (try/except/else/finally) 40
Тема 1. 9. Методы строк в Python 40
Тема 1. 10. Регулярные выражения 48
Тема 1. 11. Получение данных из разных источников 51
Тема 1. 12. Работа с датами (опционально) 51
Тема 1. 13. Анализ и визуализация данных 52
Тема 1. 14. Контрольная работа 1 67
МОДУЛЬ 2. МАШИННОЕ ОБУЧЕНИЕ 70
Тема 2.1. Искусственный интеллект. Введение в тематику искусственных нейронных сетей 70
Тема 2.2. Модель искусственного нейрона. Общее представление об искусственной нейронной сети 75
Тема 2.3. Библиотеки для обучения нейронных сетей (Keas, Tensor Flow, PyTorch, Scikit-learn) 77
Тема 2.4. Обучение искусственного нейрона 77
Практическая работа к теме 6 79
Тема 2.5. Обучение искусственной нейронной сети. Метод обратного распространения ошибки 83
Тема 2.6. Применение нейронных сетей для решения задачи классификации 84
Тема 2.7. Применение нейронных сетей для решения задачи регрессии 84
Тема 2.8. Анализ качества обучения нейронной сети 84
Тема 2.9 Контрольная работа 2 84
МОДУЛЬ 3. КОМПЬЮТЕРНОЕ ЗРЕНИЕ 84
Тема 3.1. Оцифровка изображений 84
Тема 3.2. Сверточные нейронные сети 94
Тема 3.3. Распознавание объектов на изображении 96
Тема 3.4. Предварительно обученные нейронные сети 96
Тема 3.5. Перенос обучения в нейронных сетях 100
Тема 3.6. Распознавание предметов одежды. Построение архитектуры нейронной сети и ее обучение. 100
Тема 3.7. Контрольная работа 3 100
МОДУЛЬ 4. АНАЛИЗ ЕСТЕСТВЕННОГО ЯЗЫКА 100
Тема 4.1. Нейронные сети для задач обработки естественного языка 100
Тема 4.2. Одномерные сверточные нейронные сети 106
Тема 4.3. Рекуррентные нейронные сети для задач обработки естественного языка 106
ЧАСТЬ 2 РЕШЕНИЕ ЗАДАЧ КОМПЬЮТЕРНОГО ЗРЕНИЯ 106
9. Statsmodels 110
Что можно делать с помощью Statsmodels? 110
10. Plotly 110
Что можно делать с помощью Plotly? 111
1. SpaCy 111
2. Bokeh 111
3. Gensim 111
4. NLTK 112

Диаграмма рассеяния — Scatter Plot.....112
Задача №1 115
https://pythonru.com/primery/kak-ispolzovat-modul-datetime-v-python 115

ВВЕДЕНИЕ

Дисциплина ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ(Python) является частью информационного цикла (вариативная часть) дисциплин подготовки учащихся. Дисциплина реализуется в 7-11 классах школы.

Программой дисциплины предусмотрены следующие виды контроля: текущий контроль успеваемости в форме выполнения практических работ, тестов и рубежный контроль в форме контрольных работ.

Тема	Количество часов
Модуль 1. Язык программирования Python для анализа данных	
Тема 1.1. Установка Python, IDE, создание модулей. Проверка идиоматичности кода	2
Тема 1.2. Условный оператор if	2
Тема 1. 3. Последовательности и циклы	2
Тема 1. 4. Список как изменяемая последовательность	2
Тема 1. 5. Кортежи, словари и множества	2
Тема 1. 6. Функции	2
Тема 1. 7. Файлы, работа с файловой системой	2
Тема 1. 8. Обработка исключений	2
Тема 1. 9. Методы строк в Python	2
Тема 1. 10. Регулярные выражения	2
Тема 1. 11. Получение данных из разных источников	2
Тема 1. 12. Работа с датами (опционально)	2
Тема 1. 13. Анализ и визуализация данных	2
Тема 1. 14. Контрольная работа 1	2
Модуль 2. Машинное обучение	
Тема 2.1. Искусственный интеллект. Введение в тематику искусственных нейронных сетей	2
Тема 2.2. Модель искусственного нейрона. Общее представление об искусственной нейронной сети	2
Тема 2.3. Библиотеки для обучения нейронных сетей	2
Тема 2.4. Обучение искусственного нейрона	2
Тема 2.5. Обучение искусственной нейронной сети. Метод обратного распространения ошибки	2
Тема 2.6. Применение нейронных сетей для решения задачи классификации	2
Тема 2.7. Применение нейронных сетей для решения задачи регрессии	2
Тема 2.8. Анализ качества обучения нейронной сети	2
Тема 2.9 Контрольная работа 2	2
Модуль 3. Компьютерное зрение	
Тема 3.1. Оцифровка изображений	2
Тема 3.3. Сверточные нейронные сети	2

Тема	Количество часов
Тема 3.3. Распознавание объектов на изображении	2
Тема 3.4. Предварительно обученные нейронные сети	2
Тема 3.5. Перенос обучения в нейронных сетях	2
Тема 3.6. Распознавание предметов одежды. Наборы данных. Архитектура нейронной сети	2
Тема 3.7. Контрольная работа 3	2
Модуль 4. Анализ естественного языка	
Тема 4.1. Нейронные сети для задач обработки естественного языка	2
Тема 4.2. Одномерные сверточные нейронные сети	2
Тема 4.3. Рекуррентные нейронные сети для задач обработки естественного языка	2
Тема 4.4. Контрольная работа 4	**2**
Итоговая аттестация	

МОДУЛЬ 1. ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ АНАЛИЗА ДАННЫХ

Тема 1.1. Установка Python, IDE, создание модулей. Проверка идиоматичности кода

Создание модулей. Проверка идиоматичности кода

Модуль (*англ.* Module) - специальное средство языка программирования, позволяющее объединить вместе данные и функции и использовать их как одну функционально-законченную единицу (например, математический модуль, содержащий тригонометрические и прочие функции, константы $\pi\pi$, $\epsilon\epsilon$ и т.д.).

Пакеты (*англ.* Package) являются еще более крупной единицей и представляют собой набор взаимосвязанных модулей, предназначенных для решения задач определенного класса некоторой предметной области (например, пакет для решения систем уравнений, который может включать математический модуль, модуль со специальными типами данных и т.д.).

Модуль - отдельный файл с кодом на Python, содержащий функции и данные:

1. имеет расширение *.py (имя файла является именем модуля);
2. может быть импортирован (подключен) (директива import ...);
3. может быть многократно использован.

Пакеты в Python - это способ структуризации модулей. Пакет представляет собой папку, в которой содержатся модули и другие пакеты и обязательный файл __init.py__, отвечающий за инициализацию пакета.

Установка пакетов (pip)

Необходимые дополнительные пакеты можно установить, используя утилиту pip, например (установка пакета flake8):

```
pip install flake8
```

Команда pip install... по умолчанию устанавливает пакет для всех пользователей системы, требуя наличия административных прав. Если Ваш пользователь обладает ограниченными правами (такое может быть, например, в компьютерном классе вуза), используйте параметр –user

Пример

```
pip install --user flake8
```

После установки пакета выполните его обновление:

```
pip install flake8 --upgrade
```

IDE Geany

Язык интерфейса

Geany использует язык ОС в качестве языка интерфейса по умолчанию. Его можно изменить, используя терминал:

```
set lang=en
geany
```

Все модули/пакеты в Python можно разделить на 4 категории:

1. Встроенные (*англ.* Built-in).

Модули, встроенные в язык и предоставляющие базовые возможности языка (написаны на языке Си).

К встроенным относятся как модули общего назначения (например, **math** или **random**), так и плаиформозависимые модули (например, модуль **winreg**, предназначенный для работы с реестром ОС Windows, устанавливается только на соответствующей ОС).

Список установленных встроенных модулей можно посмотреть следующим образом:

```
>>> import sys
>>>
>>> print(sys.builtin_module_names)  # Встроенные модули
('_ast', '_bisect', '_codecs', '_codecs_cn', '_codecs_hk', '_codecs_iso2022',
'_codecs_jp', '_codecs_kr', '_codecs_tw', '_collections', '_csv', '_datetime',
'_functools', '_heapq', '_imp', '_io', '_json', '_locale', '_lsprof', '_md5',
'_multibytecodec', '_opcode', '_operator', '_pickle', '_random', '_sha1',
'_sha256', '_sha512', '_signal', '_sre', '_stat', '_string', '_struct',
'_symtable', '_thread', '_tracemalloc', '_warnings', '_weakref', '_winapi',
'array', 'atexit', 'audioop', 'binascii', 'builtins', 'cmath', 'errno',
'faulthandler', 'gc', 'itertools', 'marshal', 'math', 'mmap', 'msvcrt',
'nt', 'parser', 'sys', 'time', 'winreg', 'xxsubtype', 'zipimport', 'zlib')
```

2. Стандартная библиотека (*англ.* Standard Library).

Модули и пакеты, написанные на Python, предоставляющие расширенные возможности, например, **json** или **os**.

3. Сторонние (*англ.* 3rd Party).

Модули и пакеты, которые не входят в дистрибутив Python, и могут быть установлены из каталога пакетов Python (*англ.* PyPI - the Python Package Index, более 90.000 пакетов) с помощью утилиты pip:

```
C:\Users\yuri>pip install vk
Collecting vk
  Downloading vk-2.0.2.tar.gz
Requirement already satisfied (use --upgrade to upgrade):
  requests<3.0,>=2.8 in
c:\users\yuri\appdata\local\programs\python\python35\lib\site-packages (from vk)
Installing collected packages: vk
  Running setup.py install for vk ... done
Successfully installed vk-2.0.2
```

При установке пакета автоматически устанавливаются зависимые пакеты.

4. Пользовательские (собственные).

Модули и пакеты, создаваемые разработчиком.

Примечание

Создание собственных пакетов не рассматривается в рамках настоящего курса.

В собственной программе рекомендуется выполнять импорт именно в таком порядке: от встроенных до собственных модулей/пакетов.

Собственные модули

Python предоставляет возможность создания собственных модулей.

Примечание

PEP8.

Соглашение рекомендует использовать:

- змеиный_регистр (*англ.* snake_case) для наименования модулей: `my_module`;
- строки документации.

Для создания собственных модулей нет особенных правил - любой файл с расширением `*.py` является модулем. Пример создания и подключения собственного модуля приведен в Листинге 9.1.2 (а)-(б).

Листинг 9.1.2 (а) - Модуль вычисления чисел Фибоначчи (файл `fibonacci.py`)

```
""" Модуль для работы с числами Фибоначчи.

- список чисел Фибоначчи, не превышающих 'n';
- проверка на вхождение в ряд Фибоначчи;
- ...
"""


def list_le_than(n):
    """Вернуть список чисел Фибоначчи, не превышающих 'n'."""
    assert isinstance(n, int) and n > 0

    a, b = 1, 1
    result = [a]
    while b <= n:
        result.append(b)
        a, b = b, a + b
    return result


def is_in_row(n):
    """Вернуть True, если 'n' - число Фибоначчи."""
    return n in list_le_than(n)
```

Листинг 9.1.2 (б) - Использование модуля `fibonacci` (файл `main.py`)

```
import fibonacci
```

num = 20

print("Числа до **{}**: **{}**".format(num, fibonacci.list_le_than(num))) *# [1, 1, 2, 3, 5, 8, 13]*

print("**{}** входит: **{}**".format(num, fibonacci.is_in_row(num))) *# False*

Примечание

Основной модуль программы, как правило, удобно называть `main.py`.

Использование модуля: запуск или импорт

Любой модуль в Python может быть:

запущен автономно (как скрипт, например, в командной строке или через IDE);

импортирован (через **`import`**).

При написании собственного модуля необходимо предполагать, будет он запускаться автономно или использоваться для импорта. При этом следует избегать обособленного кода, который выполнялся бы в обоих вариантах (Листинг 9.1.3 (а)-(б)).

Листинг 9.1.3 (а) - Модуль `fibonacci`, включающий код, который будет выполнен и при запуске, и при импорте

```
""" Модуль для работы с числами Фибоначчи.

- список чисел Фибоначчи, не превышающих 'n';
- проверка на вхождение в ряд Фибоначчи;
- ...
"""


def list_le_than(n):
    """Вернуть список чисел Фибоначчи, не превышающих 'n'."""
    assert isinstance(n, int) and n > 0

    a, b = 1, 1
    result = [a]
    while b <= n:
        result.append(b)
        a, b = b, a + b
    return result


def is_in_row(n):
    """Вернуть True, если 'n' - число Фибоначчи."""
    return n in list_le_than(n)


print(list_le_than(10))  # Этот код выполнится, как при запуске, так и при импорте
```

Листинг 9.1.3 (б) - Использование модуля `fibonacci` (файл `main.py`) | скачать**.**

```
import fibonacci  # Здесь будет "неожиданный" вывод на экран: [1, 1, 2, 3, 5, 8]


num = 20
print("Числа до {}: {}".format(num, fibonacci.list_le_than(num)))  # [1, 1, 2, 3, 5, 8, 13]
print("{} входит: {}".format(num, fibonacci.is_in_row(num)))  # False
```

Чтобы выполнить различный код в зависимости от того, запущен модуль или импортирован, достаточно использовать специальный идентификатор `__name__`, который содержит:

- имя модуля, если он был импортирован (например, `"fibonacci"` или `"math"`);
- специальное наименование `"__main__"`, если модуль был запущен автономно (Листинг 9.1.4 (а)-(б)).

Листинг 9.1.4 (а) - Модуль вычисления чисел Фибоначчи с проверкой `__name__` (файл `fibonacci.py`) | скачать

```
""" Модуль для работы с числами Фибоначчи.

- список чисел Фибоначчи, не превышающих 'n';
- проверка на вхождение в ряд Фибоначчи;
- ...
"""


def list_le_than(n):
    """Вернуть список чисел Фибоначчи, не превышающих 'n'."""
    assert isinstance(n, int) and n > 0

    a, b = 1, 1
    result = [a]
    while b <= n:
        result.append(b)
        a, b = b, a + b
    return result


def is_in_row(n):
    """Вернуть True, если 'n' - число Фибоначчи."""
    return n in list_le_than(n)
```

```
if __name__ == "__main__":
    print("Данный код выполняется т.к. модуль был запущен автономно (как скрипт) и не будет выполнен при импорте.")
    print("Тест функций:")
    print("- list_le_than(10) =", list_le_than(10))
    print("- is_in_row(10) =", is_in_row(10))
```

Листинг 9.1.4 (б) - Сравнение выполнения модуля при самостоятельном запуске и при импорте | скачать**.**

```
C:\>python fibonacci.py
Данный код выполняется т.к. модуль был запущен автономно (как скрипт) и не будет выполнен при импорте.
Тест функций:
- list_le_than(10) = [1, 1, 2, 3, 5, 8]
- is_in_row(10) = False

C:\>python
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import fibonacci
>>> fibonacci.list_le_than(20)  # Нет вывода как выше
[1, 1, 2, 3, 5, 8, 13]
```

Как правило, проверка **__name__** используется в основном файле программы, а вспомогательные модули избавляются от кода вне функций и/или классов (Листинг 9.1.5 (а)-(б)).

Листинг 9.1.5 (а) - Модуль вычисления чисел Фибоначчи (файл fibonacci.py) | скачать

```
""" Модуль для работы с числами Фибоначчи.

- список чисел Фибоначчи, не превышающих 'n';
- проверка на вхождение в ряд Фибоначчи;
- ...
"""


def list_le_than(n):
    """Вернуть список чисел Фибоначчи, не превышающих 'n'."""
    assert isinstance(n, int) and n > 0

    a, b = 1, 1
    result = [a]
    while b <= n:
        result.append(b)
        a, b = b, a + b
```

```
    return result


def is_in_row(n):
    """Вернуть True, если 'n' - число Фибоначчи."""
    return n in list_le_than(n)
```

Листинг 9.1.5 (б) - Проверка режима запуска модуля `main` (файл `main.py`)

```
import fibonacci

if __name__ == "__main__":
    num = 20
    print("Числа до {}: {}".format(num, fibonacci.list_le_than(num)))  # [1, 1, 2, 3, 5, 8, 13]
    print("{} входит: {}".format(num, fibonacci.is_in_row(num)))  # False
else:
    # else добавлять не нужно, если никакие специальные действия
    #     в случае неавтономного запуска не планируется
    print("Данный модуль не предназначен для импорта.")
```

Анализ данных (Data Science) извлекают знания находят закономерности в данных

прогнозируют Используют методы Статистики Эконометрики Машинного обучения, Deep learning

1. Pandas

*Pandas (*Pandas - *библиотека анализа данных Python).* Кто-нибудь знал об этом?

Когда использовать? Pandas - это инструмент для обработки данных. Он предназначен для быстрой и простой обработки данных, чтения, агрегирования и визуализации.

Pandas берет данные в файле CSV или TSV или базу данных SQL и создает объект Python со строками и столбцами, который называется фреймом данных. Фрейм данных очень похож на таблицу в статистическом программном обеспечении, скажем, в Excel или SPSS.

Задачи применения Pandas

1. Индексирование, манипулирование, переименование, сортировка, объединение фрейма данных;
2. Обновить, добавить, удалить столбцы из фрейма данных;
3. Восстановить недостающие файлы, обработать недостающие данные или NAN;
4. Построить гистограмму или прямоугольную диаграмму.

Data Science. статью.

Структуры данных Pandas. Series

Это одномерный массив или одна колонка какойто таблицы. Похоже на словари.

Пример

Data = pd.Series(

["Январь", "Февраль" , "Март", "Апрель"] ,

Index=["Первый", "Второй", "Третий", "Четвёртый"])

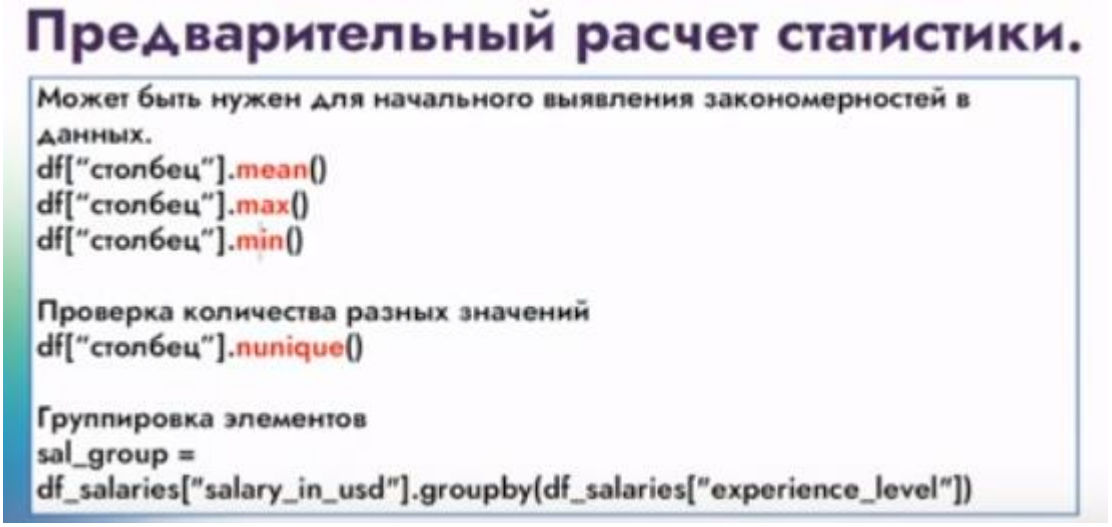
Функции работы с Dataframe. Изменение

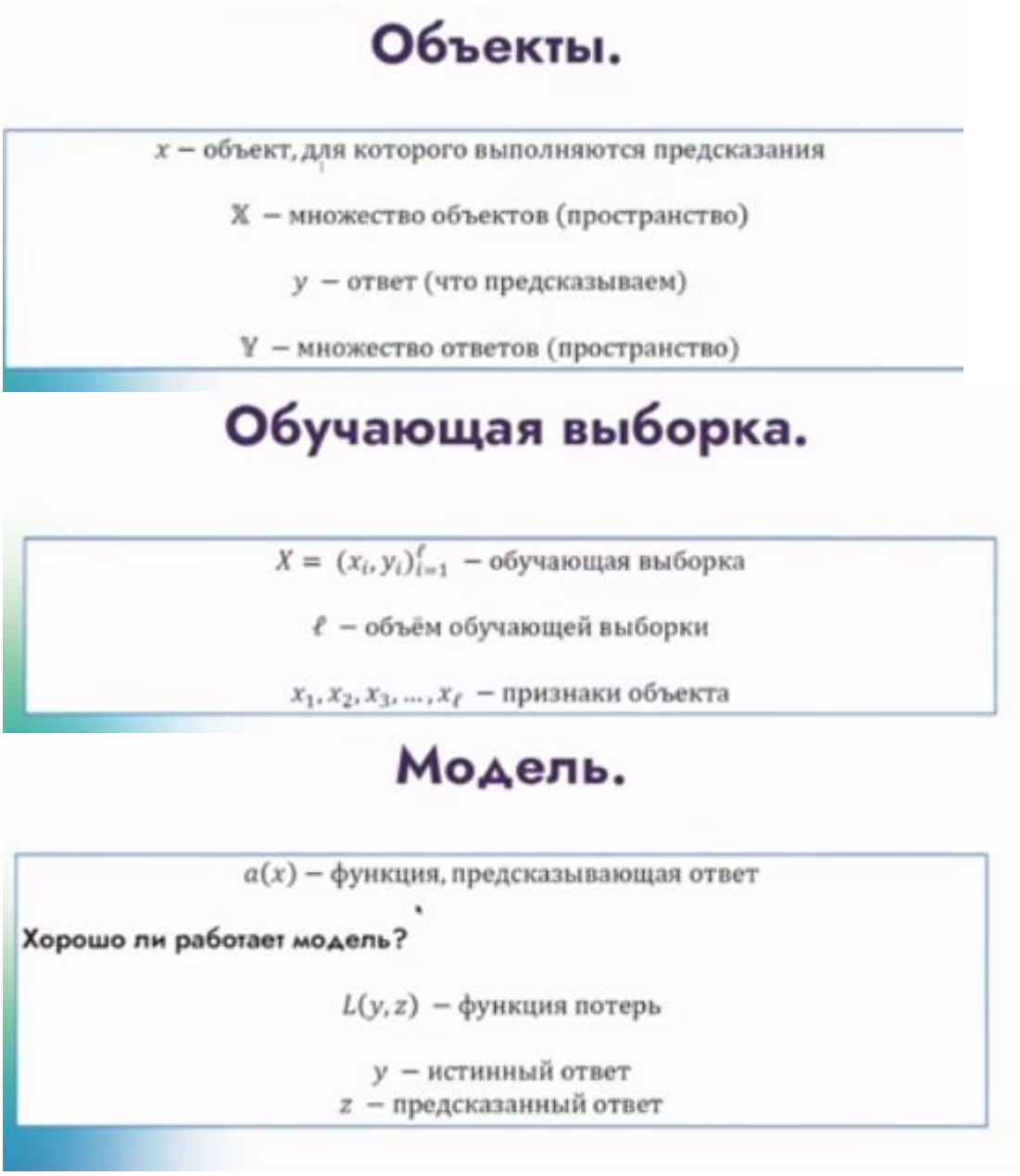
Df.drop_duplicates() Удаляет дубликаты строк из дата сета, если такие есть

Df.columns - Свойство содержит названия столбцов таблицы

Df.rename(columns={old_name:new_name})кция для переименования столбцов в DataFrame

Df.dropna(Axis=None) удалеение строк (или столбцов), содержащих null значения.





2. NumPy

NumPy - один из самых фундаментальных пакетов в Python - универсальный пакет для обработки массивов. Он предоставляет высокопроизводительные объекты многомерных массивов и инструменты для работы с массивами. NumPy - это эффективный контейнер универсальных многомерных данных.

Основной объект NumPy - это однородный многомерный массив. Это таблица элементов или чисел одного и того же типа данных, проиндексированная набором натуральных чисел. В NumPy размеры называются *осями,* а число осей называется *рангом.* Класс массива NumPy называется *ndarray*, он же *array*.

Когда использовать? NumPy используется для обработки массивов, в которых хранятся значения одного и того же типа данных. NumPy облегчает математические операции над массивами и их векторизацию. Это значительно повышает производительность и, соответственно, ускоряет время выполнения.

Что можно делать с помощью NumPy?

1. Основные операции с массивами: добавление, умножение, срез, выравнивание, изменение формы, индексирование массивов;
2. Расширенные операции с массивами: стековые массивы, разбиение на секции, широковещательные массивы;
3. Работа с DateTime или линейной алгеброй;
4. Основные нарезки и расширенное индексирование в NumPy Python.

О 4 приемах Python NumPy, которые должен знать каждый новичок, читайте здесь.

3. SciPy

Библиотека SciPy является одним из ключевых пакетов, которые составляют стек SciPy. Теперь есть разница между SciPy Stack и библиотекой SciPy. *SciPy* основывается на объекте массива NumPy и является частью стека, который включает в себя такие инструменты, как Matplotlib, Pandas и SymPy с дополнительными инструментами.

Библиотека SciPy содержит модули для эффективных математических процедур, таких как линейная алгебра, интерполяция, оптимизация, интеграция и статистика. Основной функционал библиотеки SciPy построен на NumPy и его массивах.

Когда использовать? SciPy использует массивы в качестве базовой структуры данных. Он имеет различные модули для выполнения общих задач научного программирования, таких как линейная алгебра, интеграция, матанализ, обыкновенные дифференциальные уравнения и обработка сигналов.

Что можно делать с помощью SciPy?

1. Математические, научные, инженерные вычисления;
2. Процедуры численной интеграции и оптимизации;
3. Поиск минимумов и максимумов функций;
4. Вычисление интегралов функции;
5. Поддержка специальных функций;
6. Работа с генетическими алгоритмами;
7. Решение обыкновенных дифференциальных уравнений.

4. Matplotlib

Это, несомненно, моя любимая и основная библиотека Python. Вы можете создавать истории с данными, визуализированными с помощью Matplotlib. Еще одна библиотека из стека SciPy - Matplotlib - строит 2D-фигуры.

Когда использовать? Matplotlib - это библиотека Python, предоставляющая API для встраивания графиков в приложения. Очень напоминает MATLAB, встроенный в язык программирования Python.

Что можно делать с помощью Matplotlib?

Гистограммы, столбцовые диаграммы, точечные диаграммы, круговые диаграммы - Matplotlib может отображать широкий спектр визуализаций. Приложив немного усилий, с Matplotlib, вы можете создавать любые визуализации:

1. Линейные диаграммы;
2. Точечные диаграммы;

3. Диаграммы с областями;
4. Столбцовые диаграммы и гистограммы;
5. Круговые диаграммы;
6. Диаграммы «стебель-листья»;
7. Контурные графики;
8. Поля векторов;
9. Спектрограммы.

Matplotlib также облегчает использование меток, сеток, легенд и некоторых других объектов форматирования. В общем, речь идет обо всем, что можно нарисовать!

5. Seaborn

Итак, когда вы читаете официальную документацию по Seaborn, она определяется как библиотека визуализации данных на основе Matplotlib, предоставляющем высокоуровневый интерфейс для изображения интересных и информативных статистических графиков. Проще говоря, seaborn - это расширение Matplotlib с дополнительными возможностями.

Так в чем разница между Matplotlib и Seaborn? Matplotlib используется для основного построения столбцовых, круговых, линейных, точечных диаграмм и пр., в то время как Seaborn предоставляет множество шаблонов визуализации с меньшим количеством синтаксических правил, причем более простых.

Что можно делать с помощью Seaborn?

1. Определять отношения между несколькими переменными (корреляция);
2. Соблюдать качественные переменные для агрегированных статистических данных;
3. Анализировать одномерные или двумерные распределения и сравнивать их между различными подмножествами данных;
4. Построить модели линейной регрессии для зависимых переменных;
5. Обеспечить многоуровневые абстракции, многосюжетные сетки.

Seaborn — это отличный вариант для библиотек визуализации R, таких как *corrplot* и *ggplot.*

6. Scikit Learn

Scikit Learn, представленный миру как проект Google Summer of Code, представляет собой надежную библиотеку машинного обучения для Python. Он включает в себя алгоритмы ML, такие как SVM, random forests, k-means кластеризацию, спектральную кластеризацию, сдвиг среднего значения, перекрестную проверку и многие другие. Даже NumPy, SciPy и связанные с ними научные операции поддерживаются Scikit Learn, при этом Scikit Learn является частью SciPy Stack.

Когда использовать? Scikit-learn предоставляет ряд контролируемых и неконтролируемых алгоритмов обучения через согласованный интерфейс в Python. Scikit learn будет вашим руководством для того, чтобы модели контролируемого обучения, такие как Naive Bayes, группировали непомеченные данные, такие как KMeans.

Что можно делать с помощью Scikit Learn?

1. Классификация: обнаружение спама, распознавание изображений;

2. Кластеризация: воздействия лекарственных препаратов, цена акций;
3. Регрессия: сегментация клиентов, группировка результатов эксперимента;
4. Уменьшение размерности: визуализация, повышенная эффективность;
5. Выбор модели: повышенная точность благодаря настройке параметров;
6. Предварительная обработка: подготовка входных данных в виде текста для обработки с помощью алгоритмов машинного обучения.
Scikit Learn фокусируется на моделировании данных; не манипулировании данными. Для обобщения и манипуляции у нас есть NumPy и Pandas.
7. TensorFlow
Еще в 2017 году я получил TensorFlow USB в знак признательности за то, что был потрясающим докладчиком на мероприятии Google WTM, хаха. На USB был загружен официальный документ о TensorFlow. Не имея представления о том, что такое TensorFlow, я его погуглил.
TensorFlow - это библиотека AI, которая помогает разработчикам создавать крупномасштабные нейронные сети со многими слоями, используя графики потоков данных. TensorFlow также облегчает построение моделей глубокого обучения, продвигает современную технологию ML / AI и позволяет легко развертывать приложения на базе ML.
Одним из наиболее развитых веб-сайтов среди всех библиотек является TensorFlow. Гиганты, такие как Google, Coca-Cola, Airbnb, Twitter, Intel, DeepMind, все используют TensorFlow!
Когда использовать? TensorFlow достаточно эффективен, когда дело доходит до классификации, восприятия, понимания, обнаружения, прогнозирования и создания данных.
Что можно делать с помощью TensorFlow?
1. Распознавание голоса / звука - IoT, автомобильная промышленность, безопасность, UX/UI, телекоммуникации;
2. Анализ настроений - в основном для CRM или CX;
3. Текстовые Приложения — Обнаружение угроз, Google Translate, Gmail Smart Reply;
4. Распознавание лиц - Facebook’s Deep Face, Photo tagging, Smart Unlock;
5. Временной ряд - рекомендации от Amazon, Google и Netflix;
6. Обнаружение видео - обнаружение движения, обнаружение угроз в реальном времени в играх, безопасности, аэропортах.
Более подробно о TensorFlow рассказываем в статье.
https://youtu.be/BQg9OZdzLLE?si=yU1EH7yIrzcRvsfq

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.layers import Dense

c = np.array([-40, -10, 0, 8, 15, 22, 38])
f = np.array([-40, 14, 32, 46, 59, 72, 100])

model = keras.Sequential()
model.add(Dense(units=1, input_shape=(1,), activation='linear'))
model.compile(loss='mean_squared_error', optimizer=keras.optimizers.Adam(0.1))

log = model.fit(c, f, epochs=500, verbose=False)
print("Обучение завершено")

print(model.predict([100]))
print(model.get_weights())
```

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.layers import Dense
```

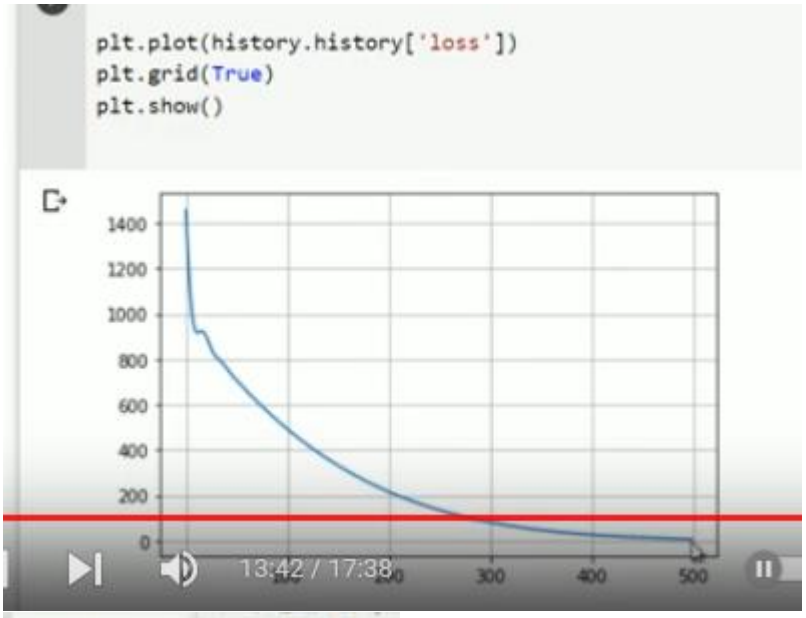
Обучающая выборка

```
c = np.array([-40, -10, 0, 8, 15, 22, 38])
f = np.array([-40, 14, 32, 46, 59, 72, 100])
```

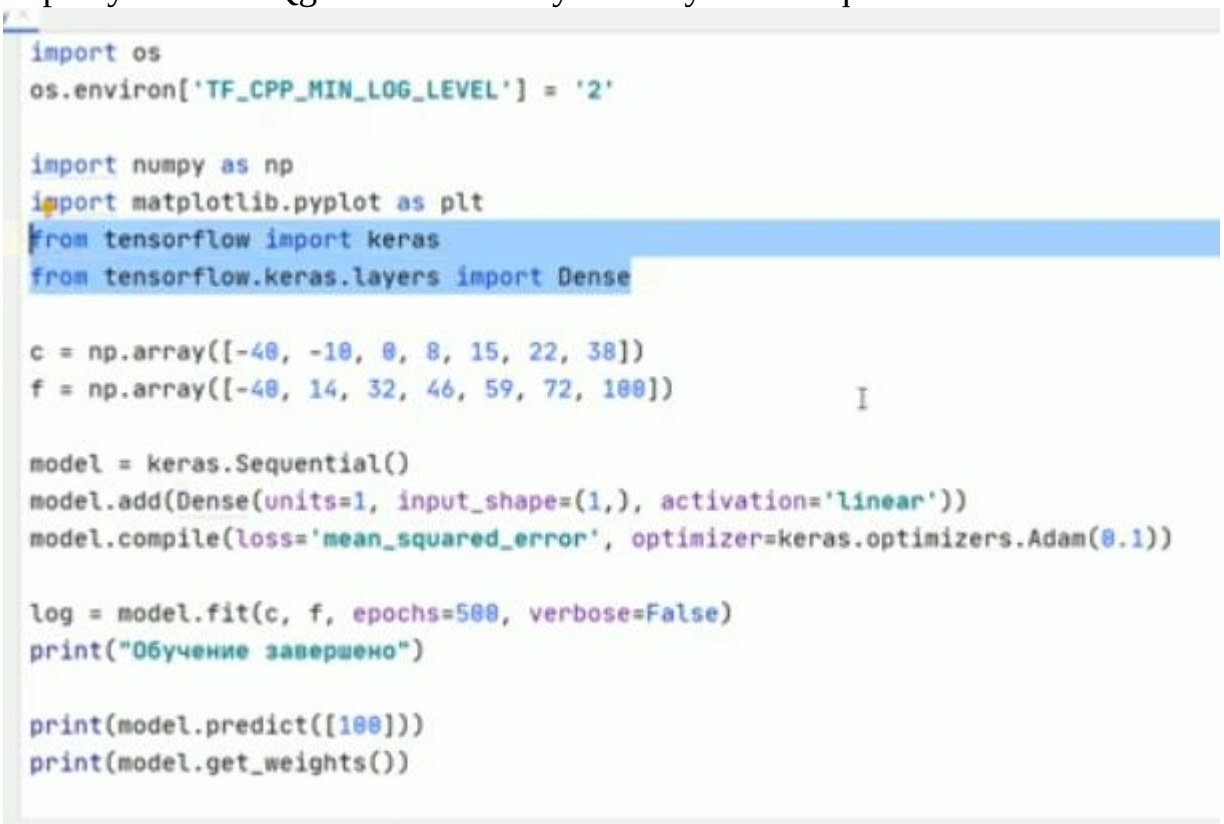
```
model = keras.Sequential()
model.add(Dense(units=1, input_shape=(1,), activation='linear'))
```

```
model.compile(loss='mean_squared_error', optimizer=keras.optimizers.Adam(0.1))
```

```
history = model.fit(c, f, epochs=500, verbose=0)

plt.plot(history.history['loss'])
plt.grid(True)
plt.show()
```



https://youtu.be/BQg9OZdzLLE?si=yU1EH7yIrzcRvsfq

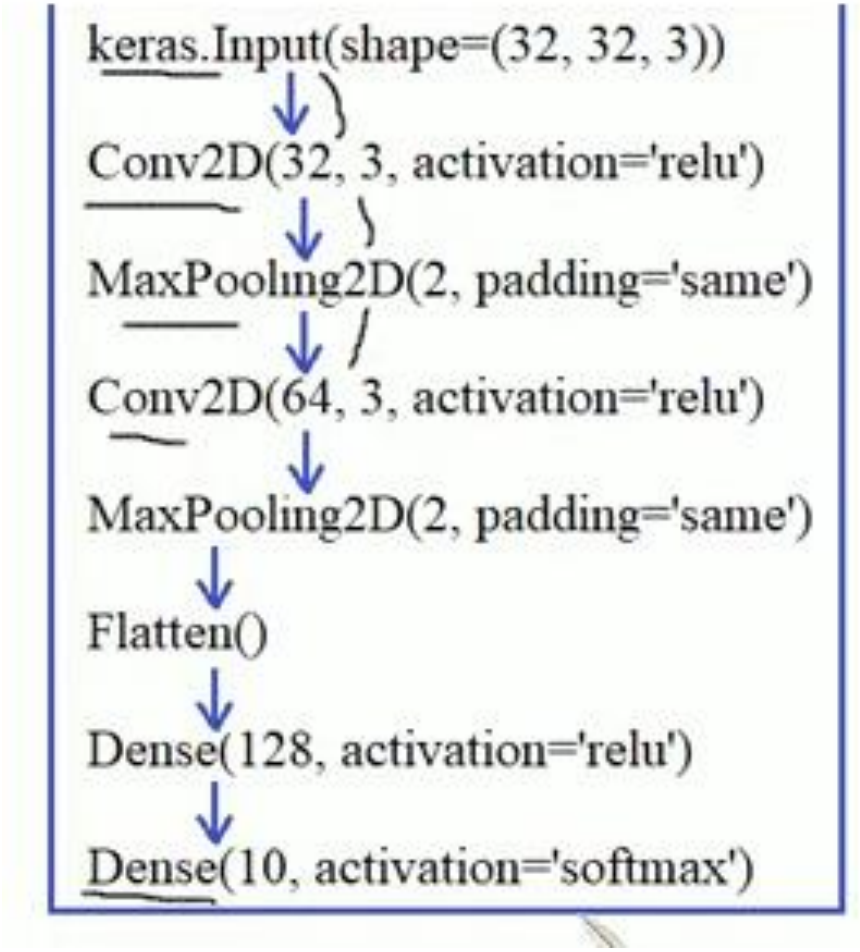


```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.layers import Dense

c = np.array([-40, -10, 0, 8, 15, 22, 38])
f = np.array([-40, 14, 32, 46, 59, 72, 100])

model = keras.Sequential()
model.add(Dense(units=1, input_shape=(1,), activation='linear'))
model.compile(loss='mean_squared_error', optimizer=keras.optimizers.Adam(0.1))

log = model.fit(c, f, epochs=500, verbose=False)
print("Обучение завершено")

print(model.predict([100]))
print(model.get_weights())
```

8. Keras

Keras - это высокоуровневый API TensorFlow для создания и обучения кода глубоких нейронных сетей. Это библиотека нейронных сетей с открытым исходным кодом на Python. С Keras статистическое моделирование, работа с

изображениями и текстом намного легче с упрощенным кодированием для глубокого обучения.
В чем разница между Keras и TensorFlow?
Keras - это нейросетевая библиотека, написанная на языке Python, а TensorFlow - это библиотека с открытым исходным кодом для различных задач машинного обучения. TensorFlow предоставляет как высокоуровневые, так и низкоуровневые API, в то время как Keras предоставляет только высокоуровневые API. Keras *создан для Python* и делает его более удобным, модульным и компонуемым, чем TensorFlow.
Что можно делать с помощью Keras?
1. Определить процентную точность;
2. Функция вычисления потерь;
3. Создать пользовательские функциональные слои;
4. Встроенные функции обработки данных и изображений;
5. Функции с повторяющимися блоками кода: глубиной 20, 50, 100 слоев.
https://youtu.be/Zu5RzqdQJXo?si=axJHWxihkafI39Zt



Кегазпри($Bape=(32, 32, 3))

Copy20(32, 3, асиуаНоп= те")
)
ao ра44те='заше')
у

Сопу20(64, 3, асиуаноп=те!и')
ИИ

МахРоойп?20(2, раддше=‘ате')

Нацеп()
Пепзе(128, аснуаноп=те!а")

Пепзе(10, аспуаноп='зоЙшах')

“

```
input = keras.Input(shape=(32, 32, 3))
x = Conv2D(32, 3, activation='relu')
x = x(input)

<class 'tensorflow.python.keras.engine.keras_tensor.KerasTensor'>

def __call__ (self, ...)

x = Conv2D(32, 3, activation='relu')(input)
```

Тема 1.2. Условный оператор if

Конструкция if
Синтаксис оператора if выглядит так:
if логическое выражение:
команда_1 команда_2
...
команда_n

После оператора if записывается логическое выражение.
Логическое выражение — конструкция языка программирования, результатом вычисления которойявляется «истина» или «ложь».

Конструкция if – else
Синтаксис оператора if – else выглядит так:
if логическое выражение:
команда_1 команда_2
...
команда_n
else:
команда_1команда_2

...
команда_n
Пример if/else в одну строку с дополнительным условием:

```
>>> i = 55
>>> rez = f'{i} < 50' if i < 50 else (f'{i} > 60' if i > 60 else f'50 > {i} < 60')
>>> rez
# '50 > 55 < 60'
>>> i=45
>>> rez = f'{i} < 50' if i < 50 else (f'{i} > 60' if i > 60 else f'50 > {i} < 60')
>>> rez
# '45 < 50'
>>> i=65
>>> rez = f'{i} < 50' if i < 50 else (f'{i} > 60' if i > 60 else f'50 > {i} < 60')
>>> rez
# '65 > 60'
if a < b:
    rez = a + b
else:
    rez = a - b
```

в языке Python была введено трехместное выражение if/else вида

```
# общий вид if/else в одну строку
x = a if condition else b

# Выражение примера выше будет
# выглядеть следующим образом
rez = a + b if a < b else a - b.
```

Примеры использования if/else в одну строку:
Простое (с одним условием) if/else в одну строку:

```
>>> a = 6
>>> b = 5
>>> rez = a + b if a < b else a - b
>>> rez
# 1

>>> i = []
>>> rez = a + b if not i else a - b
>>> rez
# 11
```

При помощи однострочника if/else можно вызывать связанные функции:

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

```
>>> a, b = 4, 5
>>> (subtract if a > b else add)(a, b)
# 9
```

Конструкция if – elif – else

Для реализации выбора из нескольких альтернатив можно использовать конструкцию if – elif – else.

Синтаксис оператора if – elif – else выглядит так:

```
if  логическое выражение_1:
команда_1 команда_2
...
команда_n
elif логическое выражение_2:
команда_1 команда_2
...
команда_n
elif  логическое выражение_3:
команда_1 команда_2
...
команда_n
else:
команда_1команда_2
...
команда_n
```

Тема 1. 3. Последовательности и циклы

Функция последовательности в Python - это встроенная функция, которая позволяет создавать последовательности чисел на основе определенных правил или шагов. Эти функции возвращают итерируемые объекты или генераторы, которые можно использовать для итерации по последовательности чисел

Циклы — один из основных инструментов любого Python-разработчика. С их помощью всего за пару строчек кода можно совершить сразу множество повторяющихся действий. Циклы состоят из условия и тела. Код в теле выполняется только до тех пор, пока соблюдено условие.

Пример:

```
shoplist = ['яблоки', 'манго', 'морковь', 'бананы']
name = 'swaroop'

# Операция индексирования
print('Элемент 0 -', shoplist[0])
```

```
print('Элемент 1 -', shoplist[1])
print('Элемент 2 -', shoplist[2])
print('Элемент 3 -', shoplist[3])
print('Элемент -1 -', shoplist[-1])
print('Элемент -2 -', shoplist[-2])
print('Символ 0 -', name[0])

# Вырезка из списка
print('Элементы с 1 по 3:', shoplist[1:3])
print('Элементы с 2 до конца:', shoplist[2:])
print('Элементы с 1 по -1:', shoplist[1:-1])
print('Элементы от начала до конца:', shoplist[:])

# Вырезка из строки
print('Символы с 1 по 3:', name[1:3])
print('Символы с 2 до конца:', name[2:])
print('Символы с 1 до -1:', name[1:-1])
print('Символы от начала до конца:', name[:])
```

Тема 1. 4. Список как изменяемая последовательность

Язык программирования Python предоставляет четыре встроенных типа данных для хранения коллекций из объектов: list (список), tuple (кортеж), dictionary (словарь) и set (множество).

Таблица 1

Название встроенных типов данных для хранения коллекций	Тип данных	Размер
Список (list)	список — изменяемый	размер списка — динамический
Кортеж (tuple)	кортеж — неизменяемый тип данных	размер кортежа фиксированный
Словарь (dictionary)		
Множество (set)		

Список (*list*) – изменяемая последовательность значений. Элементы списка заключаются в квадратные скобки. Элементы списка могут быть разного типа.

Объекты могут быть разными — от целых чисел до строк. Списки могут также хранить в себе списки. Список внутри другого списка является *вложенным* (*nested*). Список без элементов, называется *пустым*. Пустой список создается с помощью пустых квадратных скобок [ ] Выражение внутри скобок определяют индекс. Индексация начинается с нуля.

Оператор «+» объединяет списки. Оператор «*» повторяет список несколько раз. Оператор среза для списков работает похожим образом, как для строк. Оператор in возвращает результат вхожде- ния одного списка в другой список

Методы работы со списками в Python:

Index — возвращает положение первого совпавшего элемента.

Count — считает, сколько раз указанное значение появляется в списке.

Append — добавляет указанное значение в конец списка.

Sort — сортирует список.

Insert — вставляет элемент перед указанным индексом.

Remove — удаляет первое попавшееся вхождение элемента в списке.

Extend — позволяет добавлять списки.

Pop — удаляет последний элемент списка.

Join — преобразует список в строку.

Создание списков на Python.

Создать список можно несколькими способами

Получение списка через присваивание конкретных значений.

Так выглядит в коде Python пустой список: s = [] # Пустой список

Примеры создания списков со значениями:

```
l=[5,75,-4,7,-51]# список целых чисел
l=[1.13,5.34,12.63,4.6,34.0,12.8]# список из вещественных чисел
l=["Оля", "Владимир", "Михаил", "Дарья"]# список из строк
l=["Москва", "Иванов", 12, 124]   # смешанный список
l=[[0, 0, 0], [1, 0, 1], [1, 1, 0]]   # список, состоящий из списков
l=['s', 'p', ['isok'], 2] # список из значений и списка
```

Списки можно складывать (конкатенировать) с помощью знака «+»:

```
l=[1, 3]+[4,23]+[5]
print('l=[1, 3]+[4,23]+[5] =',l)
```

Результат:

```
>>>
l=[1, 3]+[4,23]+[5] = [1, 3, 4, 23, 5]
>>>
```

Создание списка при помощи функции Split().

Используя функцию split в Python можно получить из строки

список.stroka ="Привет, страна"
lst=stroka.split(",")

```
stroka ="Здравствуй, Дедушка Мороз"  #stroka - строка
lst=stroka.split(",") #lst - список
print('stroka = ',stroka)
print('lst=stroka.split(","):',lst)
```

Результат:

```
============== RESTART: C:/Users/maxim/Desktop/ex_list_
stroka =  Здравствуй, Дедушка Мороз
lst=stroka.split(","): ['Здравствуй', ' Дедушка Мороз']
```

Генераторы списков.

В Python создать список можно также при помощи генераторов.

Первый способ.

Сложение одинаковых списков заменяется умножением: Список из 10 элементов, заполненный единицамиl = [1]*10

Второй способ.Пример 1.

l = [i for i in range(10)]

Пример 2.

c = [c * 3 for c in 'list']

print (c) # ['lll', 'iii', 'sss', 'ttt']

```
Создание списка из строки.
l = list (строка):
 ['с', 'т', 'р', 'о', 'к', 'а']

Создание списка при помощи функции Split().
stroka=" Hello, friend "
lst=stroka.split(","):
 ['Hello', ' friend']

Генераторы списков.
Первый способ.
l = [1]*10:
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Второй способ. Пример 1.
l = [i for i in range(10)]:
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Второй способ. Пример 2.
c=[c*3 for c in "list"]:
 ['lll', 'iii', 'sss', 'ttt']
```

Примеры использования генераторов списка.

Пример 1.
Заполнить список квадратами чисел от 0 до 9, используя генератор списка.
Решение:
l = [i*i for i in range (10)]

Пример 2.
Заполнить список числами, где каждое последующее число больше на 2.l = [(i+1)+i for i in range(10)]
print(l)

Модуль random предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.
random.randint(A, B) - случайное целое число N, A ≤ N ≤ B. random.random() - случайное число от 0 до 1.

Случайные числа в списке:
10 чисел, сгенерированных случайным образом в диапазоне (10,80)
from random import randint
l = [randint(10,80) for x in range(10)]
10 чисел, сгенерированных случайным образом в диапазоне (0,1)l = [random() for i in range(10)]

Ввод списка (массива) в языке Python.
Для ввода элементов списка используется цикл for и команда range ():
for i in range(N):
x[i] = int(input())
Более простой вариант ввода списка:
x = [int(input()) for i in range(N)]
Функция int здесь используется для того, чтобы строка, введеннаяпользователем, преобразовывалась в целые числа.

Вывод списка (массива) в языке Python. Вывод целого списка (массива):
print (L)
Поэлементный вывод списка (массива):
for i in range(N):
print (L[i], end = " ")

2. Методы списков.

Метод	**Что делает**
list.append(x**)**	Добавляет элемент в конец списка
list.extend(L**)**	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x**)**	Вставляет перед i-ым элементом значение x

list.remove(x**)**	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i]**)**	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]]**)**	Возвращаетположение первого элемента со значением
list.count(x**)**	Возвращает количество элементов со значением x
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

```
# Список покупок
shoplist = ['яблоки', 'манго', 'морковь', 'бананы']
print('Нужно сделать', len(shoplist), 'покупки.')

print('Покупки:', end=' ')
for item in shoplist:
    print(item, end=' ')

print('\nТакже нужно купить риса.')
shoplist.append('рис')
print('Теперь список покупок таков:', shoplist)

print('Отсортируем список')
shoplist.sort()
print('Отсортированный список покупок выглядит так:', shoplist)

print('Первое, что нужно купить, это', shoplist[0])
olditem = shoplist[0]
del shoplist[0]
print('Купили', olditem)
print('Теперь список покупок:', shoplist)
```

Задание 1

Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти максимальный элемент. Вывести массив на экран в обратном порядке.

В массиве действительных чисел все нулевые элементы заменить на среднееарифметическое всех элементов массива.

Задание 2

Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти минимальный элемент. Вывести индекс минимального элемента на экран.

Дан массив целых чисел. Переписать все положительные элементы во второймассив, а остальные - в третий.

Задание 3

В одномерном числовом массиве D длиной n вычислить сумму элементов снечетными индексами. Вывести на экран массив D, полученную сумму.

Дан одномерный массив из 8 элементов. Заменить все элементы массиваменьшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Задание 4

Дан массив целых чисел. Найти максимальный элемент массива и егопорядковый номер.

Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Задание 5

Дан одномерный массив из 10 целых чисел. Вывести пары отрицательныхчисел, стоящих рядом.

Дан целочисленный массив размера 10. Создать новый массив, удалив всеодинаковые элементы, оставив их 1 раз.

Задание 6

Дан одномерный массив из 10 целых чисел. Найти максимальный элемент и сравнить с ним остальные элементы. Вывести количество меньших максимального и больших максимального элемента.

Одномерный массив из 10-и целых чисел заполнить с клавиатуры,определить сумму тех чисел, которые >5.

Задание 7

Дан массив целых чисел. Найти сумму элементов с четными номерами и произведение элементов с нечетными номерами. Вывести сумму и произведение.

Переставить в одномерном массиве минимальный элемент и максимальный.

Задание 8

Найдите сумму и произведение элементов списка. Результаты вывести наэкран.

В массиве действительных чисел все нулевые элементы заменить на среднееарифметическое всех элементов массива.

Задание 9

Дан одномерный массив, состоящий из N вещественных элементов. Ввести массив с клавиатуры. Найти и вывести минимальный по модулю элемент. Вывести массив на экран в обратном порядке.

Даны массивы A и B одинакового размера 10. Вывести исходные массивы. Поменять местами их содержимое и вывести в начале элементы преобразованного массива A, а затем — элементы преобразованного массиваB.

Задание 10

Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран это значение, иначе сообщение об их отсутствии.

Дан одномерный массив из 15 элементов. Элементам массива меньше 10 присвоить нулевые значения, а элементам больше 20 присвоить 1. Вывести на экран монитора первоначальный и преобразованный массивы в строчку.

Задание 11

1 Найти наибольший элемент списка, который делиться на 2 без остатка ивывести его на экран.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из четных чисел исходного массива, меньше 10, или сообщить, что такихчисел нет. Полученный массив вывести в порядке возрастания элементов.

Задание 12

Найти наименьший нечетный элемент списка и вывести его на экран.

Даны массивы A и B одинакового размера 10. Поменять местами их содержимое и вывести вначале элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Задание 13

Дан одномерный массив целых чисел. Проверить, есть ли в нем одинаковыеэлементы. Вывести эти элементы и их индексы.

Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран мониторапреобразованный массив.

Задание 14

Найти максимальный элемент численного массива и поменять его местами сминимальным.

Программа заполняет одномерный массив из 10 целых чисел числами, считанными с клавиатуры. Определить среднее арифметическое всех чисел массива. Заменить элементы массива большие среднего арифметического на 1.

Задание 15

Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран эти значения.

Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

– поменять местами наибольший и наименьший (экстремальные) элементы (если их несколько, то поменять первые из найденных);
найти сумму и произведение всех положительных элементов.

Найти сумму элементов списка, расположенных между первым и последним положительными элементами

Тема 1. 5. Кортежи, словари и множества

Кортеж (**tuple**) – неизменяемый тип данных. Эта последовательность является упорядоченной и итерируемой, к элементам можно получить доступ по индексу. Кортежи обозачаются указанием элементов, разделённых запятыми; по желанию их можно ещё заключить в круглые скобки.

Пример:

```
zoo = ('питон', 'слон', 'пингвин') # помните, что скобки не обязательны
print('Количество животных в зоопарке -', len(zoo))

new_zoo = 'обезьяна', 'верблюд', zoo
print('Количество клеток в зоопарке -', len(new_zoo))
print('Все животные в новом зоопарке:', new_zoo)
print('Животные, привезённые из старого зоопарка:', new_zoo[2])
print('Последнее животное, привезённое из старого зоопарка -', new_zoo[2][2])
print('Количество животных в новом зоопарке -', len(new_zoo)-1+len(new_zoo[2]))
```

Результат:

Словари — это неупорядоченная перечень пар в формате ключ-значение.

Пример:

```
# 'ab' - сокращение от 'a'ddress'b'ook

ab = {  'Anna'   : 'anna@gmail.com',
        'Bob'    : 'bob@wall.org',
        'Maks' : 'maks@yandex.ru',
        'Nina'   : 'nina@gmail.com'
        }

print("Адрес Anna'a:", ab['Anna'])

# Удаление пары ключ-значение
del ab['Nina']

print('\nВ адресной книге {0} контакта\n'.format(len(ab)))

for name, address in ab.items():
    print('Контакт {0} с адресом {1}'.format(name, address))

# Добавление пары ключ-значение
ab['Kot'] = 'koto@mail.com'

if 'Guido' in ab:
        print ("\nАдрес Kot:", ab['Kot'])
```

Результат:

Множество — это набор уникальных и неупорядоченных элементов неизменяемого типа. Множества — это *неупорядоченные* наборы простых объектов. Они необходимы тогда, когда присутствие объекта в наборе важнее порядка или того, сколько раз данный объект там встречается.

Базовая структура типа данных “множество”— это хеш-таблица (*Hash Table*). Поэтому множества очень быстро справляются с проверкой элементов на вхождение. Пример: содержится ли объект x в последовательности a_set. Используя множества, можно осуществлять проверку принадлежности, определять, является ли данное множество подмножеством другого множества, находить пересечения множеств и так далее.

```
>>> bri = set(['Бразилия', 'Россия', 'Индия'])
>>> 'Индия' in bri
True
>>> 'США' in bri
False
>>> bric = bri.copy()
>>> bric.add('Китай')
>>> bric.issuperset(bri)
True
>>> bri.remove('Россия')
>>> bri & bric # OR bri.intersection(bric)
{'Бразилия', 'Индия'}
```

Тема 1. 6. Функции

Функция (def) в Python — это фрагмент кода, который выполняет определённые операции и отдаёт результат.
Для объявления функции используют ключевое слово def (от англ. define — определить, обозначить). В общем виде объявление выглядит следующим образом:

```
def имя_функции (аргументы):
    тело_функции
    return результат
```

имя_функции — название, с помощью которого можно вызывать функцию в коде;
аргументы — значения, которые функция принимает на вход. Это поле может быть пустым;
тело_функции — набор инструкций, которые выполняются при вызове;
результат — значения, которые функция возвращает при завершении работы.
Чтобы вызвать функцию, надо ввести её название и передать аргументы в скобках. В общем виде синтаксис вызова выглядит так:

имя_функции(аргументы)
написать программу, которая запрашивает результаты ЕГЭ по разным предметам, считает итог и сообщает, насколько хорошо ученик справился с экзаменами.

```
# Запрашиваем результаты ЕГЭ
math = int(input())
russian = int(input())
informatics = int(input())

# Объявляем функцию, которая принимает результаты ЕГЭ, считает сумму и возвращает результат
def passed (math, russian, informatics):
    total = math + russian + informatics

    if 120 <= total < 210:
        print('Хорошо')
    elif 210 <= total < 240:
        print('Очень хорошо')
    elif total >= 240:
        print('Отлично')
    else:
        print('Неудовлетворительно')
    return total
# Вызываем функцию и выводим результат
print(passed(math, russian, informatics))
```

Тема 1. 7. Файлы, работа с файловой системой

Файл **(*англ.* File)** - именованная область данных на носителе информации.

Файлы хранятся в файловой системе - каталоге, определяющим способ организации, хранения и именования данных, а также задающем ограничения на формат и доступ к данным. На сегодняшний день наиболее популярными являются древовидные каталоги (также *директории* или *папки*) - файлы, содержащие записи о входящих в них файлах

Файл — это поименованная совокупность данных определённого размера, размещаемая на внешних устройствах и рассматриваемая в процессе обработки как единое целое.

Каталог (папка) — это поименованная совокупность файлов и подкаталогов (вложенных каталогов).

Файловая система — часть операционной системы, определяющая способ организации, хранения и именования данных на носителе информации.

Для того, чтобы «добраться» до нужного файла в многоуровневой файловой структуре, пользователь может перемещаться по каталогам, образующим путь к файлу. Кроме того, можно воспользоваться поиском по маске имени файла.

Функция **open()** - открывает файл для чтения, записи, добавления нового содержимого. Может принимать дополнительные параметры: для задания режима открытия, указания кодировки, вывода ошибок и др. Возвращает дескриптор файла, который обязательно нужно закрыть, иначе файл останется в памяти. Дескриптор в данном случае представляет собой путь к документу в виде строки;

- функция **close()** - закрывает файловый объект;

- инструкция **with** (позволяет автоматически закрывать файловый объект после работы с ним);

- метод **read()** - для чтения содержимого документа;

- метод **readlines()** - преобразует все строки файла в список;

- метод **readline()** - построчно выводит данные файла (удобно при работе с массивными документами);

- метод **write()** - записывает новую информацию в файл;

- функция **rename()** из модуля **os** - переименовывает документ и др.

При решении заданий потребуются следующие знания:

1. Способы открытия файлов в разных режимах;
2. Варианты задания кодировок;
3. Методы чтения содержимого документов;
4. Инструменты для записи файлов;
5. Популярные библиотеки для работы с файлами (**csv**, **json**, **Pillow** и др.).

Файловый объект в Python

В Python работа с файлами осуществляется через специальный абстрактный файловый объект. В зависимости от способа создания такого объекта, он может быть привязан как к физическому файлу на диске, так и другому устройству, поддерживающему схожие операции (стандартный ввод/вывод и пр.).

Стандартный способ создания файлового объекта - функция open():
open(*file, mode='r', buffering=- 1, encoding=None, errors=None, newline=None, closefd=True, opener=None*)

Открывает файл и возвращает файловый объект. Если файл не может быть открыт, возбуждается исключение OSError и его потомки (FileNotFoundError и др.).
Параметры
file – путь к файлу (например, строка) или файловый дескриптор;
mode – режим открытия файла:

Символ	**Описание**
'r'	Открыть для чтения (по умолчанию)
'w'	Открыть для записи (если файл существует, то очищается)
'x'	Открыть для создания с эксклюзивными правами (ошибка, если файл существует)
'a'	Открыть для добавления (если файл существует)
'+'	Открыть для чтения и записи
't'	Текстовый режим (по умолчанию)
'b'	Двоичный режим

Двоичный/текстовый режимы могут быть скомбинированы с другими: например, режим 'rb' позволит открыть на чтение бинарный файл;

encoding – наименование кодировки, используемой при чтении/записи файла (например, 'utf-8'); параметр имеет смысл только для текстового режима.

```
#Открытие файла для чтения данных
file = open("input.txt", "r")
#Получение строки из файла
s = file.read()
#Закрытие файла file.close()

ВАША ПРОГРАММА!

#Открытие файла в режиме чтения
file = open("output.txt", "w")
#Запись информации в файл
resultfile.write(result)
#Закрытие файла
file.close()
```

Задача 1. (Сложность 1/10)
Медведь собирал в лесу N банок с мёдом. Медведь очень привередливо выбирал мёд, и брала только те, в которых ровно Млитров мёда. Также известно, что для пропитания зимой ему необходимо не менее K мёда. Определите, хватит ли на зиму мёдамедведю.

В файле INPUT.txt необходимо задать 3 числа - N, M, K.

После завершения работы программы в выходной файл OUTPUT.txt выведите «YES» если медведю хватит мёда и «NO» в противном случае.

INPUT.txt	OUTPUT.txt
4 5 20	YES
4 5 21	NO

Решение:

```
file = open("input.txt", "r")
s = file.read().split(" ")
file.close()

result = "False"

if s[0] * s[1] > s[2]:
        result = "True"

file = open("output.txt", "w")
file.write(result)
file.close()
```

Задача 2. (Сложность 3/10)

В спортзал ходят 3 качка, которые поднимают гантели. Требуетсяопределить: на сколько вес гантели самого сильного из них отличается от веса гантели самого слабого из них.

В файле INPUT.txt необходимо задать 3 числа - Вес гантели самого сильного, вес гантели среднего по силе и вес гантели самого слабогоиз них.

После завершения работы программы в выходной файл OUTPUT.txt выведите разницу весов гантелей самого сильного и самого слабого.

INPUT.txt	OUTPUT.txt
1000 500 100	900
36 11 5	31

Решение:

```
file = open("input.txt", "r")
s = file.read().split(" ")
file.close()
result = s[0] - s[2]
```

```
file = open("output.txt", "w")
file.write(str(result))
file.close()
```

Задача 3. (Сложность 5/10)

Известны результаты каждого из 3 футбольных матчей. Нужно определить победителя среди всех матчей. Побеждает в матчекоманда, набравшая больше очков в течение всего матча.

Победителем среди всех матчей является команда выигравшаябольшее количество матчей.

В файле INPUT.txt необходимо задать 6 чисел:

- сколько забила 1-я команда в первом матче
- сколько забила 2-я команда в первом матче
- сколько забила 1-я команда во втором матче
- сколько забила 2-я команда во втором матче
- сколько забила 1-я команда в третьем матче
- сколько забила 2-я команда в третьем матче

После завершения работы программы в выходной файл OUTPUT.txt нужно написать победителя всех матчей (или число 1 или число 2 иличисло 0, если никто не победил).

INPUT.txt	OUTPUT.txt
20 10 25 15 3 12	1
10 20 9 14 7 3	2

Решение:

```
file = open("input.txt", "r")
s = file.read().split(" ") file.close()

win_1 = 0
win_2 = 0

result = 0

for i in range(0, 3):
        if s[i*2] > s[i*2+1]:
        win_1 += 1
        elif s[i*2] < s[i*2+1]:
        win_2 += 1
```

```
if win_1 > win_2:
        result = 1 elif win_2 > win_1:
        result = 2

file = open("output.txt", "w") file.write(str(result))
file.close()
```

Переименование файла

```
import os
os.rename("/home/user/testfile.txt", "/home/user/test.txt")
```

Удаление файла

```
import os
os.remove("/home/user/testfile.txt")
```

Чтение требуемой строки из текстового файла

Чтобы прочесть строку под определенным номером — можно воспользоваться как стандартным чтением файла в лист, так и использовать модуль linecache:

```
line = linecache.getline("C:\\boot.ini", 2)
# or
line = open("C:\\boot.ini").readlines()[1]
```

Перебор файлов в каталоге

```
for filename in os.listdir("../plugins"):
   print(filename)
```

Перебор файлов в каталоге по маске

```
import glob
for filename in glob.glob("../plugins/*.zip"):
   print(filename)
```

Сравнение файлов

Сравнивать файлы можно как по содержимому, так и по их свойствам, что значительно быстрее.

Оба варианта возможны при помощи filecmp

```
import filecmp
similar = filecmp.cmp('C:\\file1.txt', 'C:\\file2.txt')
```

print(similar)

Тема 1. 8. Обработка исключений (try/except/else/finally)

Программа, написанная на языке Python, останавливается сразу как обнаружит ошибку. Ошибки могут быть (как минимум) двух типов:

Синтаксические ошибки— возникают, когда написанное выражение не соответствует правилам языка (например, написана лишняя скобка);

Исключения— возникают во время выполнения программы (например, при делении на ноль).

Пример try/except

```
try:
   a = input("Введите первое число: ")
   b = input("Введите второе число: ")
   print("Результат: ", int(a)/int(b))
except ValueError:
   print("Пожалуйста, вводите только числа")
except ZeroDivisionError:
   print("На ноль делить нельзя")
```

Пример try/except/else

```
try:
   a = input("Введите первое число: ")
   b = input("Введите второе число: ")
   result = int(a)/int(b)
except (ValueError, ZeroDivisionError):
   print("Что-то пошло не так...")
else:
   print("Результат в квадрате: ", result**2)
```

Тема 1. 9. Методы строк в Python

Цель работы: познакомится с методами работы со строками.
Учащийся должен:
Владеть: Навыками составления линейных алгоритмов на языке программированияPython с использованием строковых данных;
Уметь: Применять функции и методы строк при обработке строковых данных;
Знать: Операции и методы обработки строк.
Строка базовый тип, представляющий из себя неизменяемую последовательность символов; str от «string» — «строка».
Таблица 2 Функции и методы работы со строками

Функция или метод	**Назначение**
S1 + S2	Конкатенация (сложение строк)

S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
len(S)	Длина строки
S.join(список)	Соединение строк из последовательности str через разделитель, заданный строкой
S1.count(S[, i, j])	количество вхождений подстроки s в строку s1. Результатом является число. Можно указать позицию начала поиска i и окончания поиска j
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращаетномер первого вхождения или -1
S.index(str, [start],[end])	Поиск подстроки в строке. Возвращаетномер первого вхождения или вызывает ValueError
S.rindex(str,[start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.replace(шаблон, замена)	Замена шаблона
S.split(символ)	Разбиение строки по разделителю
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру

методы для обработки строк в Python

isalpha(str): если строка в Python включает в себя лишь алфавитные символы, возвращается True;

islower(str): True возвращается, если строка включает лишь символы в нижнем регистре;

isupper(str): True, если символы строки в Python находятся в верхнем регистре;

startswith(str): True, когда строка начинается с подстроки **str;**

isdigit(str): True, когда каждый символ строки — цифра;

endswith(str): True, когда строка в Python заканчивается на подстроку str;

upper(): строка переводится в верхний регистр;

lower(): строка переводится в нижний регистр;

title(): для перевода начальных символов всех слов в строке в верхний регистр;

capitalize(): для перевода первой буквы самого первого слова строки в верхний регистр;

lstrip(): из строки в Python удаляются начальные пробелы;

rstrip(): из строки в Python удаляются конечные пробелы;

strip(): из строки в Python удаляются и начальные, и конечные пробелы;

rjust(width): когда длина строки меньше, чем параметр width, слева добавляются пробелы, строка выравнивается по правому краю;

ljust(width): когда длина строки в Python меньше, чем параметр width, справа от неё добавляются пробелы для дополнения значения **width**, при этом происходит выравнивание строки по левому краю;

find(str[, start [, end]): происходит возвращение индекса подстроки в строку в Python. В том случае, если подстрока не найдена, выполняется возвращение числа -1;

center(width): когда длина строки в Python меньше, чем параметр width, слева и справа добавляются пробелы (равномерно) для дополнения значения width, причём происходит выравнивание строки по центру;

split([delimeter[, num]]): строку в Python разбиваем на подстроки в зависимости от разделителя;

replace(old, new[, num]): в строке одна подстрока меняется на другую;

join(strs): строки объединяются в одну строку, между ними вставляется определённый разделитель.

Обрабатываем строку в Python

преобразование введенной строки в число операция **преобразования.**

Для обработки строки используем такой метод в Python, как isnumeric():

```
string = input("Введите какое-нибудь число: ")
if string.isnumeric():
    number = int(string)
    print(number)
```

Следующий пример позволяет удалять пробелы в конце и начале строки:

```
string = "   привет мир!  "
string = string.strip()
print(string)          # привет мир!
```

Так можно дополнить строку пробелами и выполнить выравнивание:

```
print("iPhone 7:", "52000".rjust(10))
print("Huawei P10:", "36000".rjust(10))
```

В консоли Python будет выведено следующее:

```
iPhone 7:      52000
Huawei P10:      36000
```

Поиск подстроки в строке

Чтобы в Python выполнить поиск в строке, используют метод find(). Он имеет три формы и возвращает индекс 1-го вхождения подстроки в строку: • find(str): поиск подстроки str производится с начала строки и до её конца; • find(str, start): с помощью параметра start задаётся начальный индекс, и именно с него и выполняется поиск; • find(str, start, end): посредством параметра end задаётся конечный индекс, поиск выполняется до него.

Когда подстрока не найдена, метод возвращает -1:

```
   welcome = "Hello world! Goodbye world!"
index = welcome.find("wor")
print(index)       # 6

# ищем с десятого индекса
index = welcome.find("wor",10)
print(index)       # 21

# ищем с 10-го по 15-й индекс
```

```
index = welcome.find("wor",10,15)
print(index)       # -1
```

Замена в строке

Чтобы в Python заменить в строке одну подстроку на другую, применяют метод replace(): • replace(old, new): подстрока old заменяется на new; • replace(old, new, num): параметр num показывает, сколько вхождений подстроки old требуется заменить на new.

Пример замены в строке в Python:

```
   phone = "+1-234-567-89-10"

# дефисы меняются на пробелы
edited_phone = phone.replace("-", " ")
print(edited_phone)     # +1 234 567 89 10

# дефисы удаляются
edited_phone = phone.replace("-", "")
print(edited_phone)     # +12345678910

# меняется только первый дефис
edited_phone = phone.replace("-", "", 1)
print(edited_phone)     # +1234-567-89-10
```

Разделение на подстроки в Python

Для разделения в Python используется метод split(). В зависимости от разделителя он разбивает строку на перечень подстрок. В роли разделителя в данном случае может быть любой символ либо последовательность символов. Этот метод имеет следующие формы: • split(): в роли разделителя применяется такой символ, как пробел; • split(delimeter): в роли разделителя применяется **delimeter**; • split(delimeter, num): параметром num указывается, какое

количество вхождений delimeter применяется для разделения. При этом оставшаяся часть строки добавляется в перечень без разделения на подстроки.

Соединение строк в Python

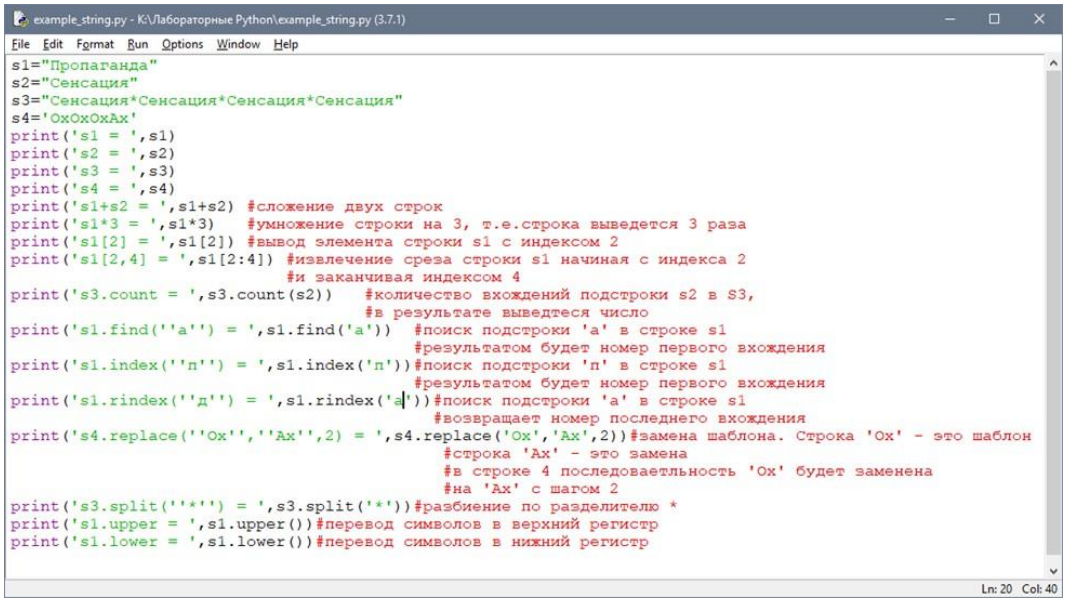
соединения строк — метод join():, объединяющий списки строк. В качестве разделителя используется текущая строка, у которой вызывается этот метод:

```
words = ["Let", "me", "speak", "from", "my", "heart", "in", "English"]

# символ разделителя - пробел
sentence = " ".join(words)
print(sentence)  # Let me speak from my heart in English

# символ разделителя - вертикальная черта
sentence = " | ".join(words)
print(sentence)  # Let | me | speak | from | my | heart | in | English
```

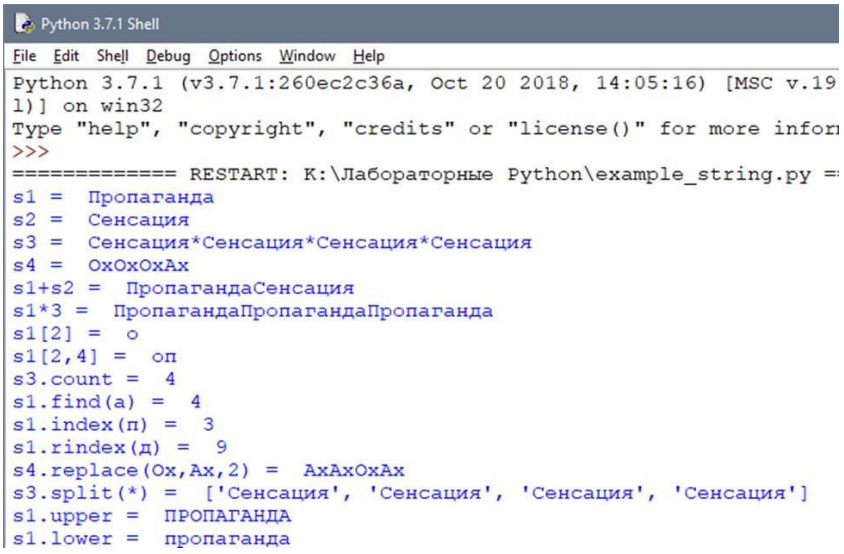
А если вместо списка в метод join передать простую строку, разделитель будет вставляться уже между символами:

```
word = "hello"
joined_word = "|".join(word)
print(joined_word)      # h|e|l|l|o
```

Ниже приведена программа, демонстрирующая использование функций иметодов работы со строками.



Пример программы на Python



Результат выполнения программы с использованием функций и методов работы со строками

Пример 1 Паллиндром

Проверить, будет ли строка читаться одинаково справа налево и слева направо(т. е. является ли она палиндромом).

Решение

Сначала введём строку командой: s=input('Введите строку '). Затем определим логическую переменную flag и присвоим ей значение1: flag=1. Для начала в введённой строке нужно удалить пробелы. Для этого воспользуемся циклической конструкцией for, которая выполнитсястолько раз,какую имеет длину строка. Длину строки определим функцией len(s).

В теле цикла будем проверять следующее условие: s[i]!=' '. Данное логическое выражение будет истинно в том случае, если i-ый элемент строки не будет

равен пробелу, тогда выполнится команда следующая после двоеточия: string+=s[i].
К сроке string, которая была объявлена в начале программы, будет добавляться посимвольно строка s, но уже без пробелов. Для проверки строки на "палиндром" воспользуемся циклической конструкцией for.
Длина половины строки находится делением нацело на 2. Если количествосимволов нечетно, то стоящий в середине не учитывается, т.к. его сравниваемая пара - он сам.
Количество повторов цикла равно длине половины строки. Длину строки определим функцией len(s), где аргумент введённая нами строка s. Зная длину строки, можно вычислить количество повторов цикла. Для этого целочисленно разделим длину строки на 2: len(s)//2. Для задания диапазона для цикла используем функцию range(), в которойаргументом будет являться половина длины строки: range(len(s//2)).
for i in range(len(s//2)).
Если символ с индексом i не равен "симметричному" символу с конца строки(который находится путем индексации с конца)
if s[i] != s[-1-i],
то переменной flag присваивается значение 0 и происходит выход из циклакомандой break.
Далее, при помощи условной конструкции if-else в зависимости от значения flag либо - 0, либо -1 выводится сообщение, что строка палиндром, либо нет.

```
s=input('Введите строку \n')
flag=1
string=''
for i in range(len(s)):
    if s[i]!=' ':
        string+=s[i]
print(string)
for i in range(len(s)//2):
    if string[i]!=string[-i-1]:
        flag=0
        break
if flag: print('Палиндром')
else: print('не палиндром')
```

Пример программы на Python

Результат выполнения программы

Задания для самостоятельной работы (по вариантам)

Вариант 1

Дана строка, содержащая русскоязычный текст. Найти количество слов,начинающихся с буквы "е".

Вариант 2

В строке заменить все двоеточия (:) знаком процента (%). Подсчитать количество замен.

Вариант 3

В строке удалить символ точку (.) и подсчитать количество удаленных символов.

Вариант 4

В строке заменить букву(а) буквой (о). Подсчитать количество замен.
Подсчитать, сколько символов в строке.

Вариант 5

В строке заменить все заглавные буквы строчными.

Вариант 6

В строке удалить все буквы "а" и подсчитать количество удаленных символов.

Вариант 7

Дана строка. Преобразовать ее, заменив звездочками все буквы "п", встречающиеся среди первых n/2 символов. Здесь n - длина строки.

Вариант 8

Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.

Вариант 9

Определить, сколько раз в тексте встречается заданное слово.

Вариант 10

Дана строка-предложение на английскомязыке. Преобразовать строку так,чтобы каждое слово начиналось с заглавной буквы.

Вариант 11

Дана строка. Подсчитать самую длинную последовательность подряд идущихбукв «н». Преобразовать ее, заменив точками все восклицательные знаки.Вариант 12

Дана строка. Вывести все слова, оканчивающиеся на букву "я".

Вариант 13

Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобки. Вывести на экран все символы, расположенные внутриэтих скобок.

Вариант 14

Дана строка. Вывести все слова, начинающиеся на букву "а" и слова оканчивающиеся на букву "я".

Вариант 15

Дана строка текста. Подсчитать количество букв «т» в строке.

Тема 1. 10. Регулярные выражения

Python имеет встроенный пакет с именем re, который можно использовать для работы с регулярными выражениями.

Модуль Re (import re) — операции с регулярными выражениями

Основная функция модуля re — предложить поиск, в котором используются регулярное выражение и строка. Здесь он либо возвращает первое совпадение, либо ничего. Модуль re вызывает исключение re.error, если ошибка возникает при компиляции или использовании регулярного выражения.

Наборы символов

В Python есть специальные обозначения для наборов символов:
\d - любая цифра
\D - любое нечисловое значение
\s - пробельные символы
\S - все, кроме пробельных символов
\w - любая буква, цифра или нижнее подчеркивание
\W - все, кроме букв, цифр или нижнего подчеркивания

Модуль re предоставляет набор функций/методов, которые позволяют нам искать строку по совпадению:

- **findall()** — Возвращает список, содержащий все совпадения
- **search()** — Возвращает объект Match, если где-либо в строке есть совпадение
- **split()** — Возвращает список, в котором строка была разделена при каждом совпадении
- **sub()** — Заменяет одно или несколько совпадений строкой
- **subn()** — Делает то же самое, что и sub(), но возвращает новую строку и количество замен
- **match()** — Ищет совпадение с начала строки
- **finditer()** — Ищет все совпадения с pattern, возвращает итератор
- **compile()** — Компилирует regular expression, на выходе получаем объект, к которому затем можно применять все перечисленные функции
- **fullmatch()** — Проверяет, что вся строка соответствует описанному регулярному выражению
- **flags (флаги)** — Указываются в функциях, влияют на поведение регулярного выражения

Задания для самостоятельного решения

1. Найдите все натуральные числа (возможно, окружённые буквами);
2. Найдите все «слова», написанные капсом (то есть строго заглавными), возможно внутри настоящих слов (аааБББввв);
3. Найдите слова, в которых есть русская буква, а когда-нибудь за ней цифра;
4. Найдите все слова, начинающиеся с русской или латинской большой буквы (\b — граница слова);

5. Найдите слова, которые начинаются на гласную (\b — граница слова);;
6. Найдите все натуральные числа, не находящиеся внутри или на границе слова;
7. Найдите строчки, в которых есть символ * (. — это точно не конец строки!);
8. Найдите строчки, в которых есть открывающая и когда-нибудь потом закрывающая скобки;
9. Выделите одним махом весь кусок оглавления (в конце примера, вместе с тегами);
10. Выделите одним махом только текстовую часть оглавления, без тегов;
11. Найдите пустые строчки;

```
import re

match = re.search(r'\d\d\D\d\d', r'Телефон 123-12-12')
print(match[0] if match else 'Not found')
# -> 23-12
match = re.search(r'\d\d\D\d\d', r'Телефон 1231212')
print(match[0] if match else 'Not found')
# -> Not found

match = re.fullmatch(r'\d\d\D\d\d', r'12-12')
print('YES' if match else 'NO')
# -> YES
match = re.fullmatch(r'\d\d\D\d\d', r'T. 12-12')
print('YES' if match else 'NO')
# -> NO

print(re.split(r'\W+', 'Где, скажите мне, мои очки??!'))
# -> ['Где', 'скажите', 'мне', 'мои', 'очки', '']

print(re.findall(r'\d\d\.\d\d\.\d{4}',
                 r'Эта строка написана 19.01.2018, а могла бы и 01.09.2017'))
# -> ['19.01.2018', '01.09.2017']

for m in re.finditer(r'\d\d\.\d\d\.\d{4}', r'Эта строка написана 19.01.2018, а могла бы и 01.09.2017'):
    print('Дата', m[0], 'начинается с позиции', m.start())
# -> Дата 19.01.2018 начинается с позиции 20
# -> Дата 01.09.2017 начинается с позиции 45

print(re.sub(r'\d\d\.\d\d\.\d{4}',
             r'DD.MM.YYYY',
             r'Эта строка написана 19.01.2018, а могла бы и 01.09.2017'))
# -> Эта строка написана DD.MM.YYYY, а могла бы и DD.MM.YYYY
```

Тема 1. 11. Получение данных из разных источников

Веб-скрапинг – это процесс автоматического сбора информации из онлайн-источников. Для выбора нужных сведений из массива «сырых» данных, полученных в ходе скрапинга, нужна дальнейшая обработка – парсинг. В процессе парсинга выполняются синтаксический анализ, разбор и очистка данных. Результат парсинга – очищенные, упорядоченные, структурированные данные, представленные в формате, понятном конечному пользователю (или приложению).

1. Если на сайте есть API, нужно запрашивать данные у него.
2. Частота и количество GET запросов должны быть разумными.
3. Следует передавать информацию о клиенте в User-Agent.
4. Если на сайте есть личные данные пользователей, необходимо учитывать настройки приватности в **robots.txt**.

Пример 1

```
from urllib.request import urlopen
url = 'http://example.com'
page = urlopen(url)
print(page.read().decode('utf-8'))
```

Пример 2

```
from bs4 import BeautifulSoup
import requests
url = 'http://example.com'
res = requests.get(url)
soup = BeautifulSoup(res.text,'html.parser')
print(soup)
```

Тема 1. 12. Работа с датами (опционально)

Datetime — важный элемент любой программы, написанной на Python. Этот модуль позволяет управлять датами и временем, представляя их в таком виде, в котором пользователи смогут их понимать.

datetime включает различные компоненты. Он состоит из объектов следующих типов:

- date — хранит дату
- time — хранит время
- datetime — хранит дату и время

Как получить текущие дату и время?

С помощью модуля Python это сделать очень просто. Сначала нужно импортировать класс datetime из модуля datetime, после чего создать объект datetime. Модуль предоставляет метод now(), который возвращает текущие дату и время с учетом локальных настроек.

```
import datetime

dt_now = datetime.datetime.now()
```

print(dt_now)
результат:
Получить текущую дату в Python
Класс date можно использовать для получения или изменения объектов даты. Например, для получения текущей с учетом настроек подойдет следующее:
from datetime import date
current_date = date.today()
print(current_date)

Тема 1. 13. Анализ и визуализация данных

Визуализация данных — это метод, который позволяет специалистам по анализу данных преобразовывать данные в диаграммы и графики, которые несут ценную информацию.

Визуализация данных — это метод, который позволяет специалистам по анализу данных преобразовывать сырые данные в диаграммы и графики, которые несут ценную информацию.

Рисунок 15.2 – Обработка и визуализация

Модули Python для анализа данных

Matplotlib используется для основного построения столбцовых, круговых, линейных, точечных диаграмм и пр.

Seaborn предоставляет множество шаблонов визуализации с меньшим количеством синтаксических правил, причем более простых.

Библиотеки **pandas и matplotlib**, чтение табличных данных, статистические показатели, построение диаграмм.

Построение графиков функций на Python

Цель: Научимся строить графики функций на python с помощью модуля matplotlib.

График функции — множество всех точек координатной плоскости, абсциссы которых равны значениям аргумента, а ординаты — соответствующим значениям функции.

Установка модулей

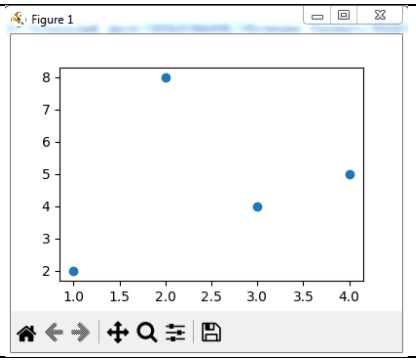
Первым делом нужно установить модули matplotlib и numpy.

1) Нажимаем значек виндовс и латинскую буквау R

Win + R — открыть утилиту "Выполнить"
2) Открывается CMD
3) В терминале прописываем:
pip install matplotlib
pip install numpy
После установки нужных модулей импортируем их в проекте:
import matplotlib.pyplot as plt
import numpy as np

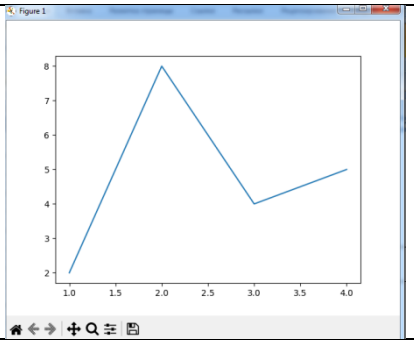
Задание 1 Построить множество точек с координатами, заданными с помощью списков

```
import matplotlib.pyplot as plt
fig=plt.figure()
x=[1,2,3,4]
y=[2,8,4,5]
plt.scatter(x,y)
plt.show()
```



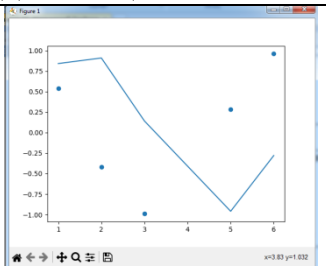
Задание 2 Построить график ломаной линии, соединяющей точки

```
import matplotlib.pyplot as plt
fig=plt.figure()
x=[1,2,3,4]
y=[2,8,4,5]
plt.plot(x,y)
plt.show()
```



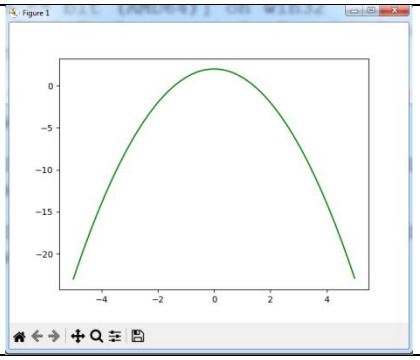
Задание 3 Построить график ломаной линии, соединяющей точки

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=[1,2,3,5,6]
plt.plot(x,np.sin(x))
plt.scatter(x,np.cos(x))
plt.show()
```



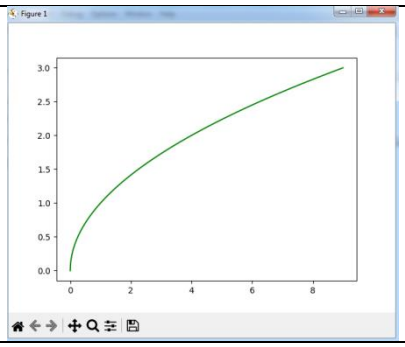
Задание 4 Построить график параболы $y = 2 - x^2$ в диапазоне от -5 до 5 с шагом 0.01

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=np.arange(-5,5,0.01)
y=2-x**2
plt.plot(x,y,color='green')
plt.show()
```



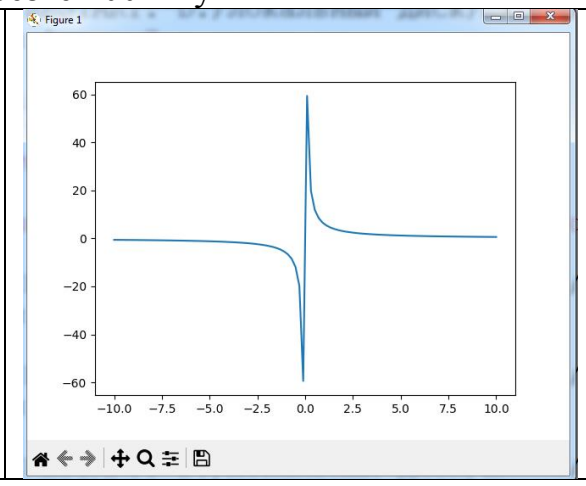
Задание 5 Построить график функции $y=\sqrt{x}$ в диапазоне от 0 до 9 с шагом 0.01

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=np.arange(0,9,0.01)
y=x**0.5
plt.plot(x,y,color='green')
plt.show()
```



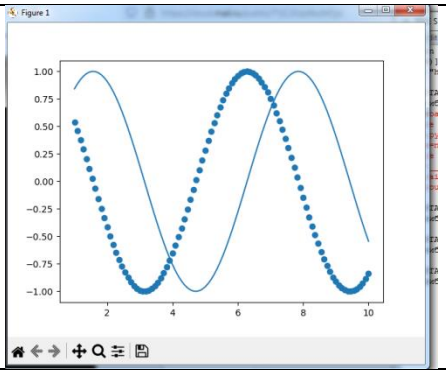
Задание 6 Построить график функции $y=\frac{6}{x}$ в диапазоне от -10 до 10 и количеством точек в единичном отрезке 100 штук

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=np.linspace(-10,10,100)
y=6/x
plt.plot(x,y)

plt.show()
```



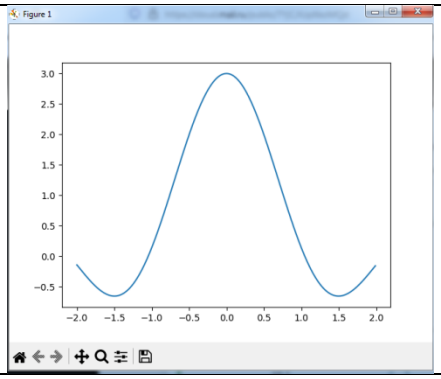
Задание 7 Построить графики синусоиды и косинусоиды в одной системе координат

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=np.linspace(1,10,100)
plt.plot(x,np.sin(x))
plt.scatter(x,np.cos(x))
plt.show()
```



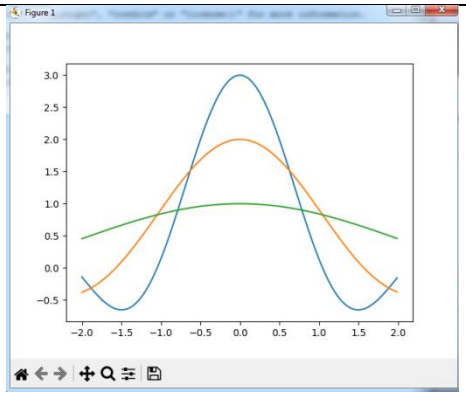
Задание 8 Построить график функции с синусом.

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=np.arange(-2,2,0.01)
y=np.sin(3*x)/x
plt.plot(x,y)
plt.show()
```



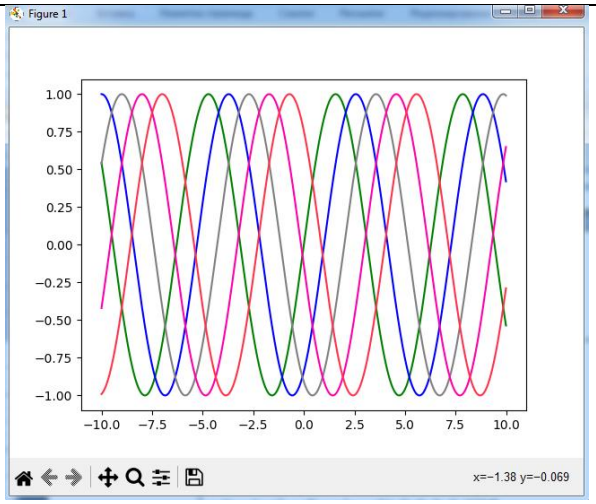
Задание 9

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=np.arange(-2,2,0.01)
y1=np.sin(3*x)/x
y2=np.sin(2*x)/x
y3=np.sin(x)/x
plt.plot(x,y1)
plt.plot(x,y2)
plt.plot(x,y3)
plt.show()
```



Цвета линий на графиках можно задать
1 способ: названием цвета;
2 способ: буквенного сокращения названия
названием цвета можно задать с помощью букв:
r-red, g-green, b-blue, c-cyan, y-yellow, k-black, w-white, m-magenta.
3 способ: числа float в диапазоне от 0 до 1 для указания отттенков серого цвета
4 способ: с помощью 16 ричного кода rgb в формате #RRGGBB
5 способ: кортеж RGB в формате три числа от 0 до 1, перечисмленные через запятую.

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=np.arange(-10,10,0.01)
y1=np.sin(x)
y2=np.sin(x-1)
y3=np.sin(x-2)
y4=np.sin(x-3)
y5=np.sin(x-4)
plt.plot(x,y1,color='green')
plt.plot(x,y2,color='b')
plt.plot(x,y3,color='0.5')
plt.plot(x,y4,color='#FF00AA')
```

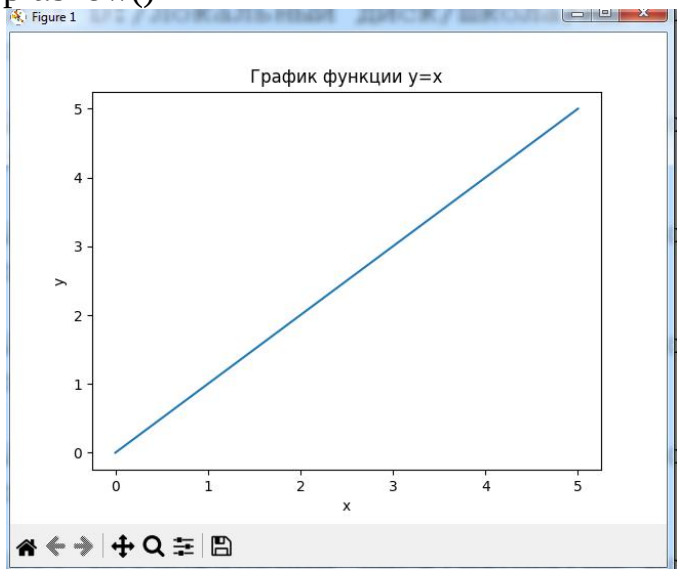


plt.plot(x,y5,color=(1.0,0.2,0.3)) plt.show()	

Задание 10 Построение прямой

#Для начала построим график прямой используя функцию y =x:

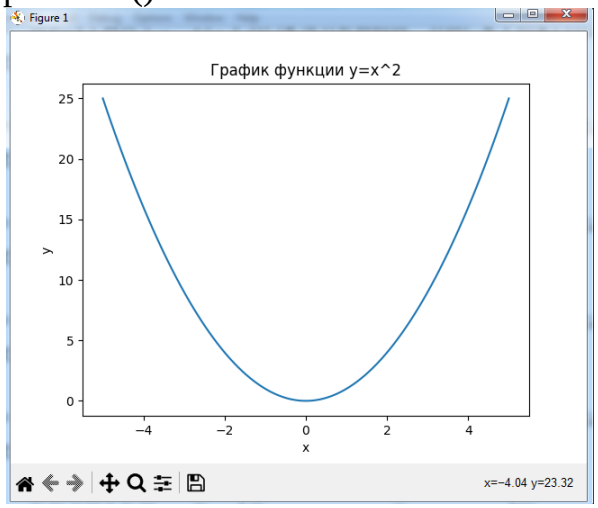
```
import matplotlib.pyplot as plt
import numpy as np

# Создаём экземпляр класса figure
# добавляем к Figure область Axes
fig, ax = plt.subplots()
# Добавим заголовок графика
ax.set_title('График функции y=x')
# Название оси X:
ax.set_xlabel('x')
# Название оси Y:
ax.set_ylabel('y')
# Начало и конец изменения значения X, разбитое на 100 точек
x = np.linspace(0, 5, 100) # X от 0 до 5
# Построение прямой
y = x
# Вывод графика
ax.plot(x, y)
plt.show()
```



Задание 11 Построение графика параболы используя функцию $y=x^2$:

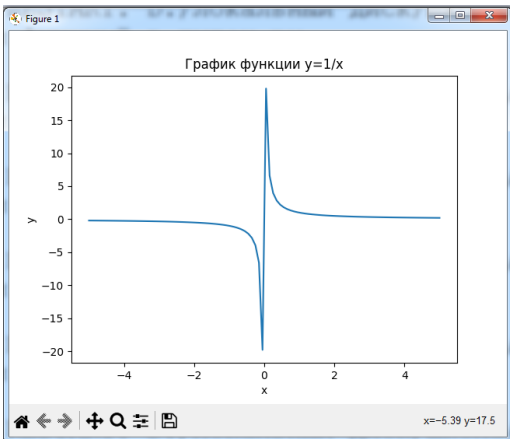
```
import matplotlib.pyplot as plt
import numpy as np

# Создаём экземпляр класса figure
#и добавляем к Figure область Axes
fig, ax = plt.subplots()
# Добавим заголовок графика
```

```
ax.set_title('График функции y=x^2')
# Название оси X:
ax.set_xlabel('x')
# Название оси Y:
ax.set_ylabel('y')
# Начало и конец изменения значения X, разбитое на 100 точек
x = np.linspace(-5, 5, 100) # X от -5 до 5
# Построение прямой
y = x**2
# Вывод графика
ax.plot(x, y)
plt.show()
```



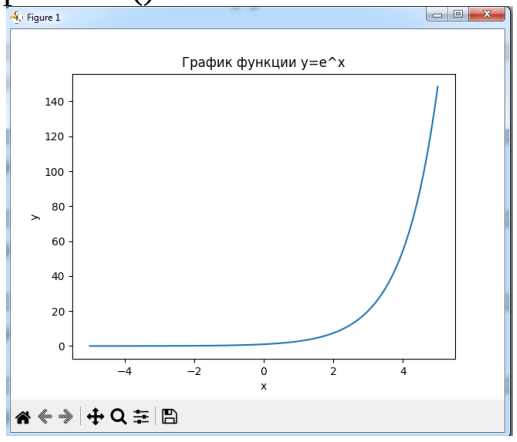
Задание 12 Построение график гиперболы, используя функцию y=1/x:

```
import matplotlib.pyplot as plt
import numpy as np

# Создаём экземпляр класса figure и добавляем к Figure область Axes
fig, ax = plt.subplots()
# Добавим заголовок графика
ax.set_title('График функции y=1/x')
# Название оси X:
ax.set_xlabel('x')
# Название оси Y:
ax.set_ylabel('y')
# Начало и конец изменения значения X, разбитое на 100 точек
x = np.linspace(-5, 5, 100) # X от -5 до 5
# Построение прямой
y = 1 / x
# Вывод графика
ax.plot(x, y)
plt.show()
```



Задание 13 график экспоненты используя функцию $y=e^x$

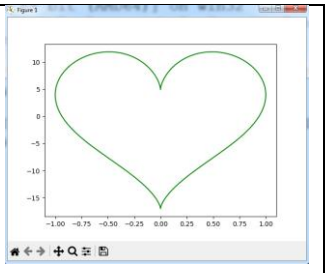
```
import matplotlib.pyplot as plt
import numpy as np

# Создаём экземпляр класса figure и добавляем к Figure область Axes
fig, ax = plt.subplots()
# Добавим заголовок графика
ax.set_title('График функции y=e^x')
# Название оси X:
ax.set_xlabel('x')
# Название оси Y:
ax.set_ylabel('y')
# Начало и конец изменения значения X, разбитое на 100 точек
x = np.linspace(-5, 5, 100) # X от -5 до 5
# Построение прямой
y = np.exp(x)
# Вывод графика
ax.plot(x, y)
plt.show()
```



Задание 14 Построить график сердца

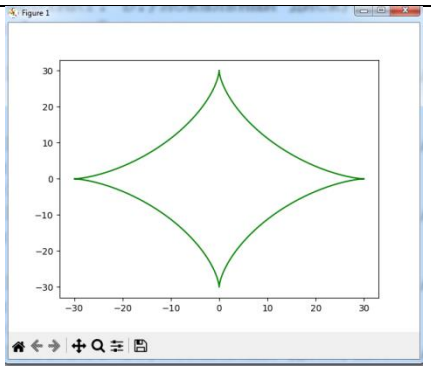
$$\begin{cases} x = 16 \cdot \sin^3(t) \\ y = 13 \cdot \cos(t) - 5 \cdot \cos(2 \cdot t) - 2 \cdot \cos(3 \cdot t) - \cos(4 \cdot t) \end{cases}$$

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
t=np.arange(0,6.28,0.01)
x=(np.sin(t))**3
y=13*(np.cos(t))-5*(np.cos(2*t))-2*(np.cos(3*t))-
(np.cos(4*t))
plt.plot(x,y,color='green')
plt.show()
```



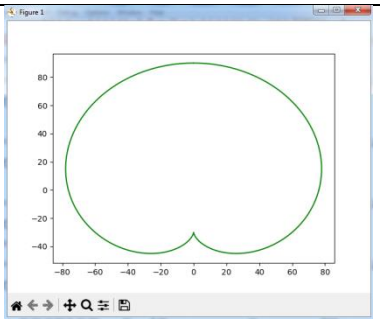
Задание 15 Построить график астроиды

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
r=30
t=np.arange(0,6.29,0.01)
x=r*(np.sin(t))**3
y=r*(np.cos(t))**3
plt.plot(x,y,color='green')
plt.show()
```



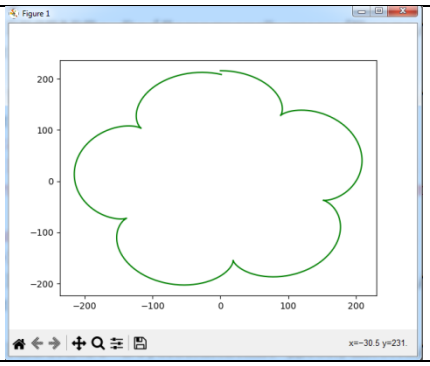
Задание 16 Построить график кардиоиды

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
r=30
t=np.arange(0,6.29,0.01)
x=2*r*np.sin(t)+r*np.sin(2*t)
y=2*r*np.cos(t)+r*np.cos(2*t)
plt.plot(x,y,color='green')
plt.show()
```



Задание 17 Построить график эллипсоиды

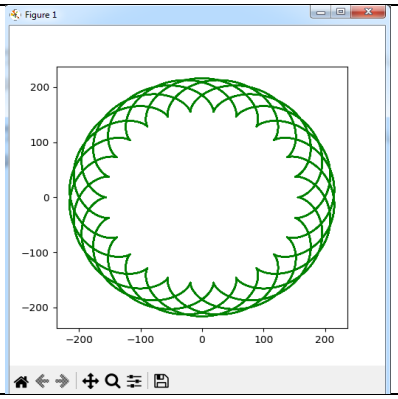
```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
r=30
t=np.arange(0,6.2,0.01)
x=6.2*r*np.sin(t)+r*np.sin(6.2*t)
y=6.2*r*np.cos(t)+r*np.cos(6.2*t)
plt.plot(x,y,color='green')
```



plt.show()	

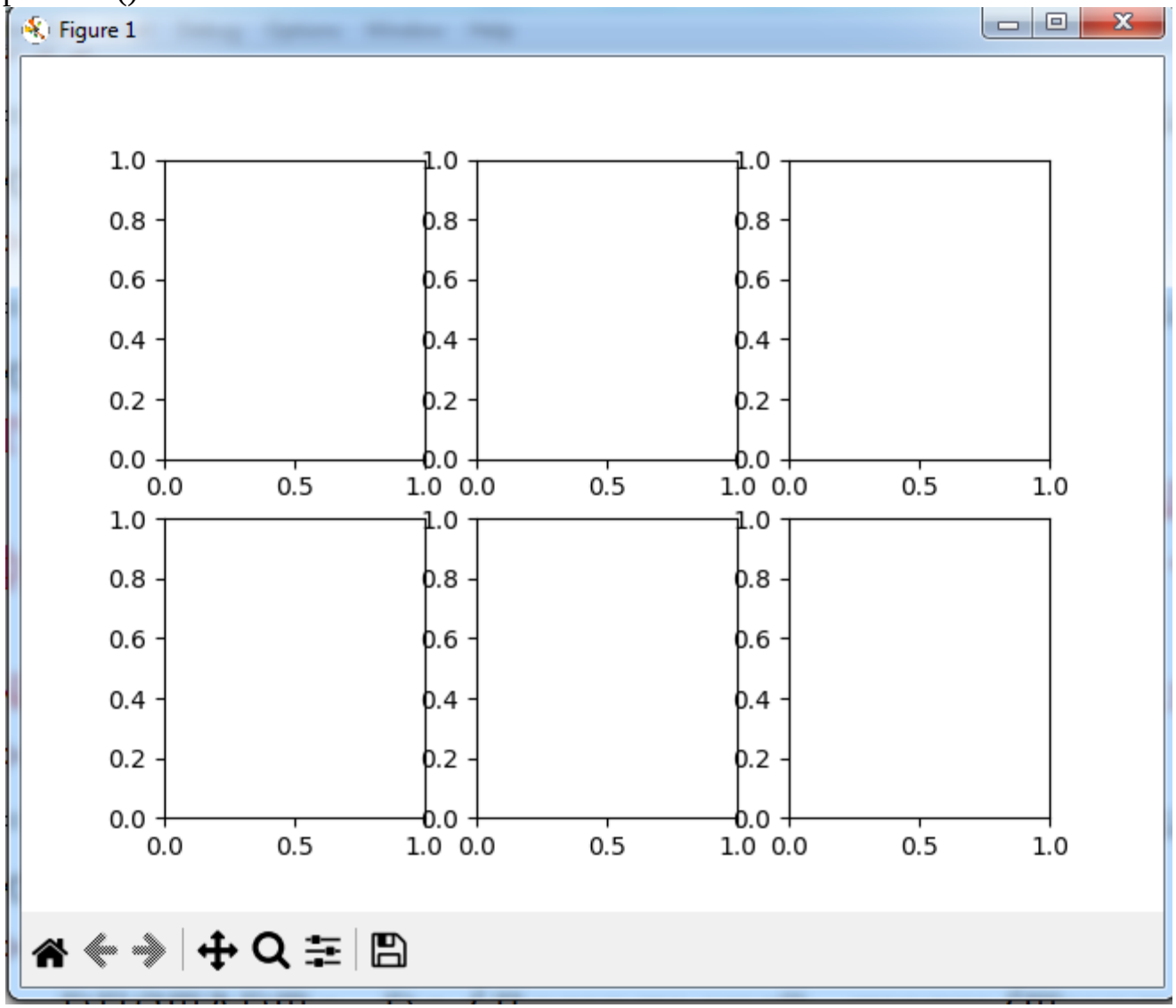
Задание 18 Построить график эллипсоиды

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
r=30
t=np.arange(0,360,0.01)
x=6.2*r*np.sin(t)+r*np.sin(6.2*t)
y=6.2*r*np.cos(t)+r*np.cos(6.2*t)
plt.plot(x,y,color='green')
plt.show()
```



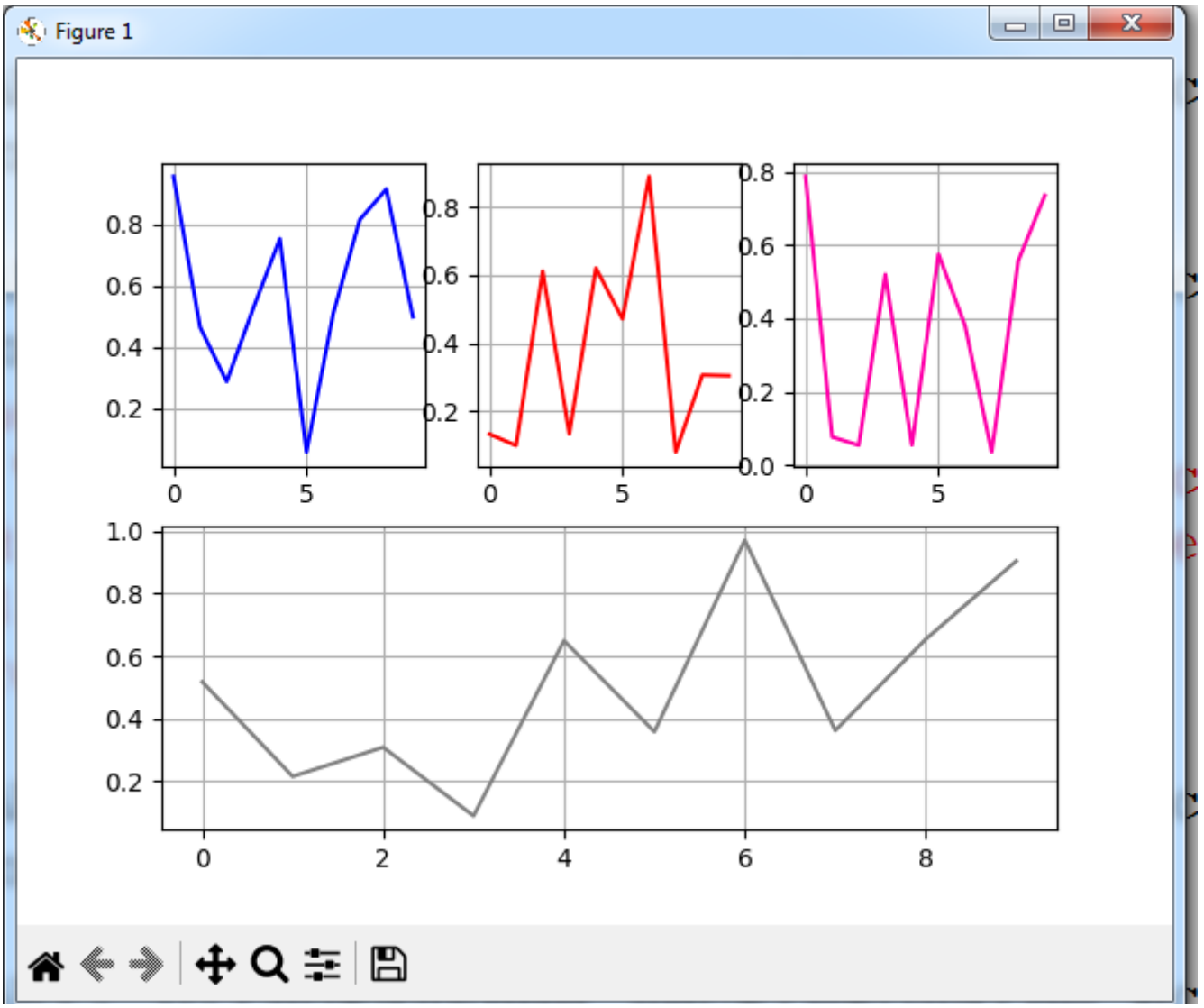
Задание 19 Построить несколько декартовых систем координат

```
import numpy as np
import matplotlib.pyplot as plt
f,ax=plt.subplots(2,3)
plt.show()
```



Задание 20 Построить несколько графиков в разных декартовых системах координат

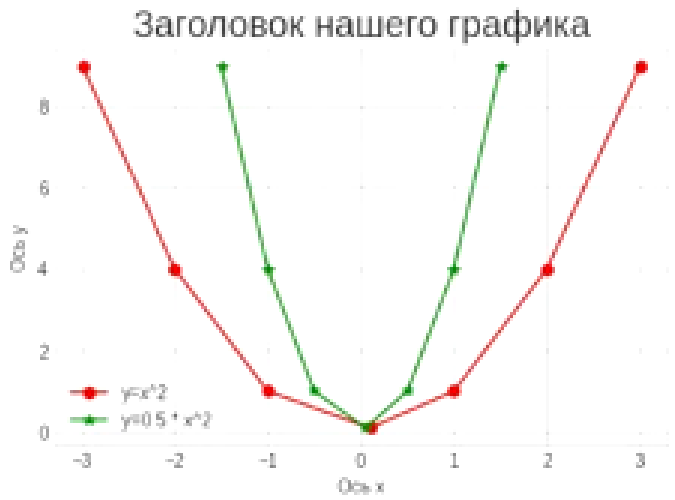
```
import numpy as np
import matplotlib.pyplot as plt
ax1=plt.subplot(2,3,1)
plt.plot(np.random.random(10),color='blue')
ax2=plt.subplot(2,3,2)
plt.plot(np.random.random(10),color='r')
ax3=plt.subplot(2,3,3)
plt.plot(np.random.random(10),color='#FF00AA')
ax4=plt.subplot(2,1,2)
plt.plot(np.random.random(10),color='0.5')
ax1.grid()
ax2.grid()
ax3.grid()
ax4.grid()
plt.show()
```



```
#Круговые диаграммы
import numpy as np
import matplotlib.pyplot as plt
vals = [10, 40, 23, 30, 7]
labels = ['Toyota', 'BMW', 'Lexus', 'Audi', 'Lada']
fig = plt.figure(figsize=(6, 4))
ax = fig.add_subplot()
ax.pie(vals, labels=labels)
plt.show()
```

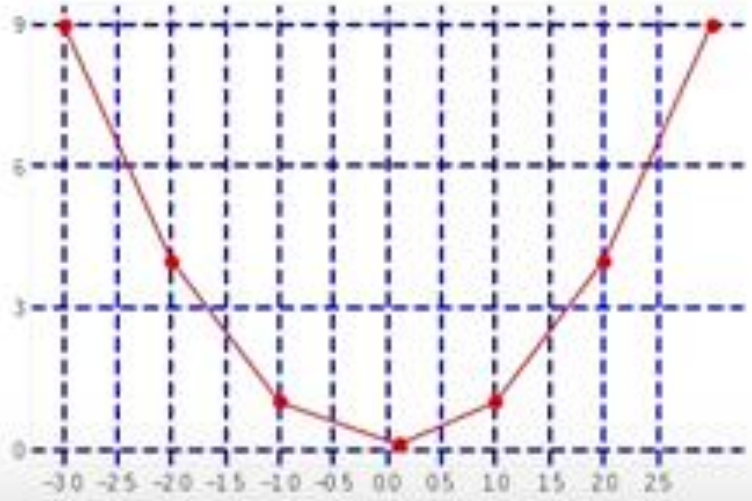
In [17]:
```
fig, ax = plt.subplots()

x = np.array([-3, -2, -1, 0.1, 1, 2, 3])
y = np.array([9, 4, 1, 0.1, 1, 4, 9])

ax.plot(x, y, color='red', linestyle='-', marker='o', label='y=x^2')

ax.plot(0.5 * x, y, color='green', linestyle='-', marker='*', label='y=0.5 * x^2')

ax.set_title('Заголовок нашего графика', fontsize=20)

ax.set_xlabel("Ось x")
ax.set_ylabel("Ось y")

ax.legend(loc='lower left')
```

Out[17]:
```
<matplotlib.legend.Legend at 0x7f77b0cb96d8>
```

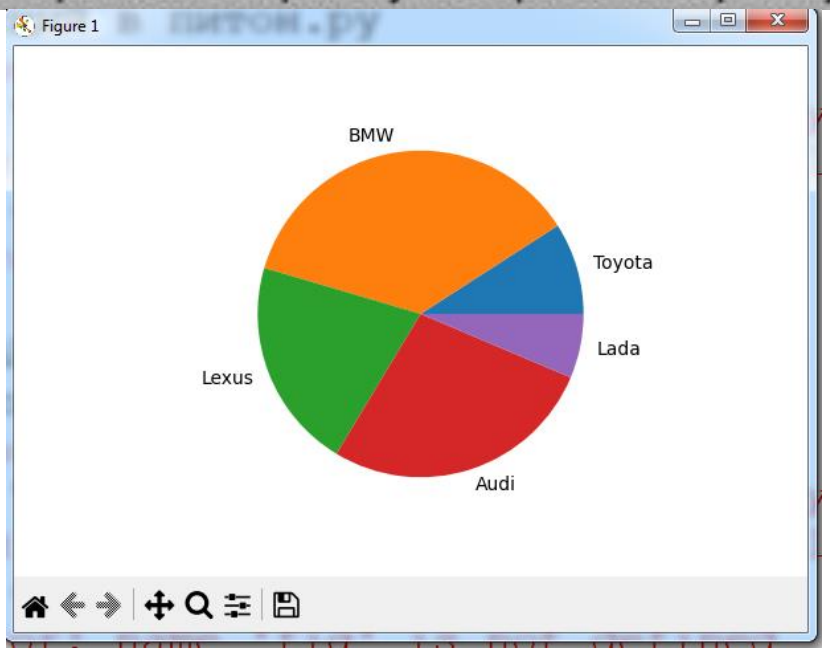


In [18]:
```
#plt.cla() очищает график

# plt.clf()  очищает рисунок

fig, ax = plt.subplots()

x = np.array([-3, -2, -1, 0.1, 1, 2, 3])
y = np.array([9, 4, 1, 0.1, 1, 4, 9])

ax.plot(x, y, color='red', linestyle='-', marker='o', label='y=x^2')
```

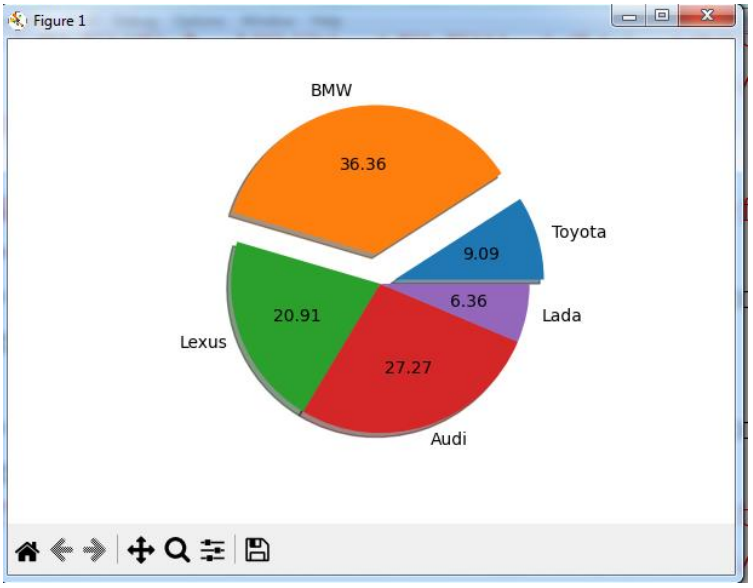
```
In [18]: #plt.cla() очищаем график

         # plt.clf()  очищаем рисунок

         fig, ax = plt.subplots()

         x = np.array([-3, -2, -1, 0.1, 1, 2, 3])
         y = np.array([9, 4, 1, 0.1, 1, 4, 9])

         ax.plot(x, y, color='red', linestyle='-', marker='o', label='y=x^2')

         ax.set_xticks(np.arange(-3, 3, 0.5))
         ax.set_yticks(np.arange(0, 10, 3))

         ax.spines['right'].set_visible(False)

         ax.grid(color = 'blue', linewidth = 2, linestyle = '--')
         # чтобы убрать сетку ax.grid()
```



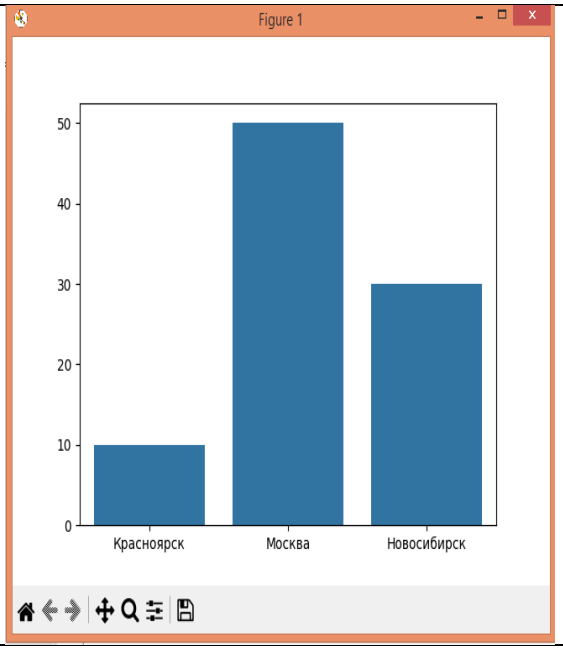
Сохраняем в картинку и открываем картинку

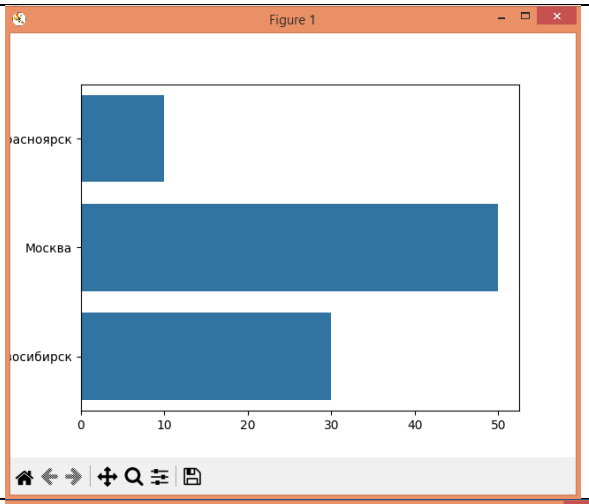
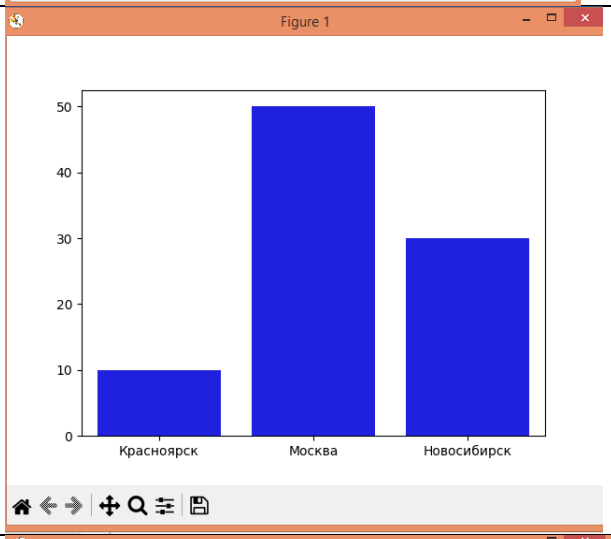
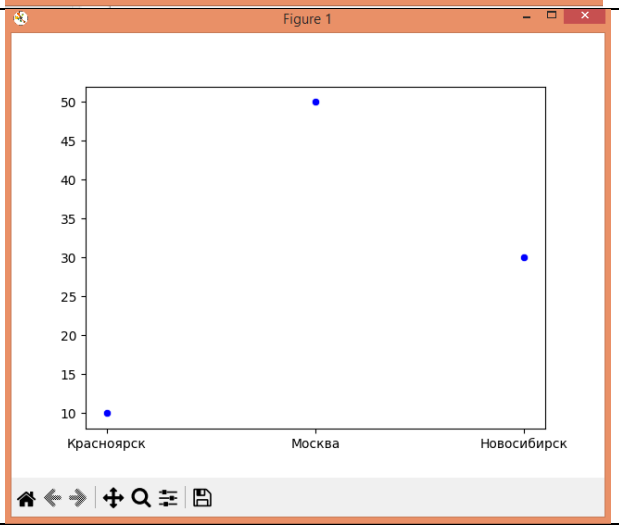


```
#Круговые диаграммы
import numpy as np
import matplotlib.pyplot as plt
vals = [10, 40, 23, 30, 7]
labels = ['Toyota', 'BMW', 'Lexus', 'Audi', 'Lada']
exp = (0.1, 0.2, 0, 0, 0)
fig = plt.figure(figsize=(6, 4))
ax = fig.add_subplot()
ax.pie(vals, labels=labels, autopct='%.2f', explode=exp, shadow=True)
plt.show()
```

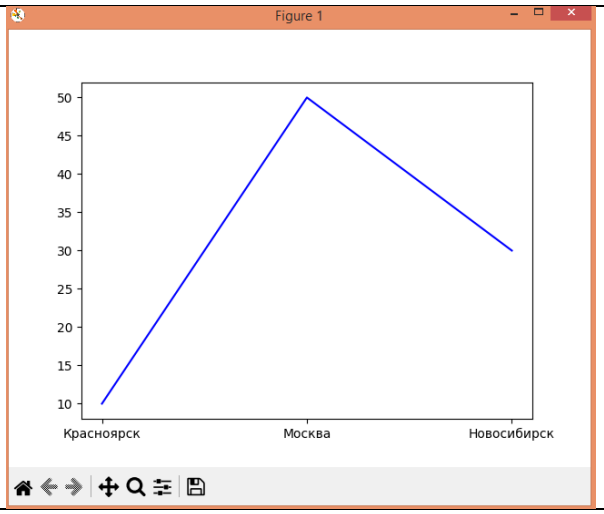



```
import seaborn as sns
import matplotlib.pyplot as plt
x = ['Красноярск', 'Москва',
'Новосибирск']
y = [10, 50, 30]
sns.barplot(x=x, y=y)
plt.show()
```



import seaborn as sns import matplotlib.pyplot as plt x = ['Красноярск', 'Москва', 'Новосибирск'] y = [10, 50, 30] sns.barplot(x=y, y=x) plt.show()	
import seaborn as sns import matplotlib.pyplot as plt x = ['Красноярск', 'Москва', 'Новосибирск'] y = [10, 50, 30] sns.barplot(x=x, y=y, color='blue') plt.show()	
import seaborn as sns import matplotlib.pyplot as plt x = ['Красноярск', 'Москва', 'Новосибирск'] y = [10, 50,30] sns.scatterplot(x=x, y=y, color='blue') plt.show()	

```
import matplotlib.pyplot as plt
x = ['Красноярск', 'Москва',
'Новосибирск']
y = [10, 50,30]
sns.lineplot(x=x, y=y, color='blue')
plt.show()
```



Задачи для самостоятельного решения к теме 12

Тема 1. 14. Контрольная работа 1

Задача 1. Создать следующего вида исходный файл из восьми строк, содержащий в каждой строке значение исходной системы счисления, результирующей системы счисления и пяти различныхцелых положительных чисел, корректных для исходной системы счисления:
10 2 121 13 10 2018 5

Задача 2. Создать следующего вида исходный файл из десяти строк, содержащий в каждой строке ФИО, дату рождения, дату по-лучения паспорта и баллы ЕГЭ по трём предметам:
Апельсинов А.А. 08.02.2000 17.03.2014 90 100 91
Виноградов В.В. 09.03.1999 15.04.2013 67 99 98
Яблоков Я.Я. 10.04.2000 19.05.2014 94 94 94
Морковкин М.М. 11.05.1999 17.06.2013 91 82 73

Задача 3. Не используя готовые сторонние подключаемые функции для факториала, int(), bin() и т.п., написать программу наязыке Python 3.x, которая бы вычисляла среднее значение баллов ЕГЭ, сортировала строки по указанной колонке в обратном по- рядке (от большего к меньшему) и выводила результат следую- щего вида (для сортировки по дате рождения):
Яблоков Я.Я. | 10.04.2000 | 19.05.2014 | 94 94 94 -> 94
Апельсинов А.А. | 08.02.2000 | 17.03.2014 | 90 100 91 -> 93,666666
Морковкин М.М. | 11.05.1999 | 17.06.2013 | 91 82 73 -> 82
Виноградов В.В. | 09.03.1999 | 15.04.2013 | 67 99 98 -> 88

Задача 4. Импортируйте набор данных о пассажирах «Титаника». Проведите разведывательный анализ данных. Поставьте гипотезу. Проведите необходимые преобразования данных.

Задача 5. На основании поставленной гипотезы в задании 1 выберите модели для предсказания. Обоснуйте выбор. Постройте выбранные модели, выберите лучшую. Постройте ансамбль моделей. Улучшились ли предсказания? Оформите результаты в виде отчета.

Задача 6. Вариант проекта: соберите данные для анализа с сайта Федеральной службы государственной статистики. Прове- дите предварительную обработку данных. Поставьте задачу клас- сификации. Проведите необходимые преобразования данных. По- стройте модель. Оцените, насколько хорошо модель справляется с поставленной задачей. Опишите результат, назовите возможные причины плохой работы модели (в случае, если результаты неудовлетворительные).

Задача 7. Создайте книгу Excel в Python. Подпишите «шапку»таблицы. Заполните все ячейки одним и тем же числом 123. До- бавьте строки с суммой и средним значением.

Задача 8. Возьмём три строки:

s1 = 'daleonov@specialist.ru's2 = 'not an email'

s3 = """ sadjflak asdkfjla asklfj daleonov@specialist.ru afasdkasdfla asf ;asldfka fas;dlfka user@gmail.com asklfjalsdf asfk info@mail.rusdfkljasl asldkfjal

"""

Первая, очевидно, почтой не является. Вторая является. Тре- тья - это текст, содержащий адреса электронной почты, но сам по себе он не почта, а текст. Создайте регулярные выражения.

Задача 9. Вы хотите узнать на сайте Центробанка РФ курсы доллара с 1 по 31 января 2021 года, на ваш запрос появилась такаяссылка в браузере: http://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=01/01/2021& date_req2=31/01/2021&VAL_NM_RQ=R01235 Это текст, который содержит интересующие вас данные. Какие? Охарактеризуйте их. Как называется данная выгрузка?

Задача 10. Напишите программу Pandas для отображения всехзаписей файла REGIONS.

Задача 11. Напишите программу Pandas для отображения имени, фамилии, оклада и идентификатора менеджера, где иден- тификаторы менеджера равны нулю.

Задача 12. Напишите программу Pandas для отображения всехидентификаторов местоположений из файла местоположений.

Задача 13. Напишите программу Pandas для извлечения пер- вых 7 записей из файла сотрудников.

Задача 14. Напишите программу Pandas, чтобы выбрать от- дельный идентификатор отдела из файла сотрудников.

Задача 15. Напишите программу Pandas, в которой будут

отображаться имя и фамилия, а также номер отдела для всех со-трудников, чья фамилия «Иванов».

Задача 16. Напишите программу Pandas для отображения имени, фамилии, оклада и идентификатора менеджера, где иден- тификаторы менеджера не равны нулю.

Задача 17. Напишите программу Pandas для создания и отображения логического ряда, где True для not null и False для null- значений или пропущенных значений в столбце state_province файла местоположений.

Задача 18. Напишите программу Pandas для отображения имени, фамилии, оклада и номера отдела для тех сотрудников, менеджеры которых имеют ID 120, 103 или 145.

Задача 19. Напишите программу Pandas, чтобы отобразить имя, фамилию, оклад и номер отдела для тех сотрудников, для которых буква n является третьим символом в их имени.

Задача 20. Напишите программу Pandas для отображения имени, идентификатора задания, зарплаты и отдела для тех сотрудников, которые не работают в отделах 50, 30 и 80.

МОДУЛЬ 2. МАШИННОЕ ОБУЧЕНИЕ

Машинное обучение — это исследование и разработка алгоритмов, которые учат компьютерные системы выполнять задачи самостоятельно на основе данных и логических выводов из них [1].

Тема 2.1. Искусственный интеллект. Введение в тематику искусственных нейронных сетей

Искусственный интеллект (ИИ) — технология, позволяющая системе, машине или компьютеру выполнять задачи, требующие разумного мышления, то есть имитировать поведение человека для постепенного обучения с использованием полученной информации и решения конкретных вопросов.
Интеграция ИИ в механизмы и системы позволяет автоматизировать рутинные, трудоемкие или сложные процессы, в том числе повысить их точность и производительность. Поэтому, эта технология — важный бизнес-ресурс.

Искусственный интеллект — это область информатики, которая занимается разработкой интеллектуальных компьютерных систем, то есть систем, обладающих возможностями, которые мы традиционно связываем с человеческим разумом, — понимание языка, обучение, способность рассуждать, решать проблемы и т. д.

Преимущества внедрения ИИ

Использование искусственного интеллекта и решений на его базе обеспечивает бизнесу ряд преимуществ.

1. Исключение человеческого фактора. Использование программируемых, самообучающихся алгоритмов исключает фактор человеческой ошибки и позволяет находить даже неочевидные для человека решения.
2. Снижение рисков. Машины с ИИ могут применяться в ситуациях, связанных с риском для человека. Например, роботы с ИИ могут заменить человека на отдельных производственных участках или при работе в условиях стихийных бедствий.
3. Круглосуточная доступность. Интеллектуальные машины можно использовать без перерывов, выходных, они не реагируют на отвлекающие факторы.
4. Адаптируемость. В рамках установленных условий применение ИИ-решений позволяет находить быстрые решения. Например, ИИ в чат-ботах помогает лучше понимать «живой» язык клиентов, находить ответы на сложно сформулированные вопросы, справляться с большим потоком одновременных обращений и вопросов.
5. Быстрое принятие решений. Приложения, машины, приборы и другие инструменты на базе ИИ принимают решения быстрее людей, что может использоваться в производственных процессах, в процессе аналитики данных, создании прогностических моделей, расчетах и других задачах.

Проблемы внедрения ИИ

Есть несколько причин, замедляющих внедрение и использование искусственного интеллекта.

1. Для контролируемого обучения (с учителем) нейросетей нужно размечать (маркировать) наборы данных вручную. На это требуется много времени.
2. Для обучения моделей нужен большой объем данных, которые нужно предварительно собрать из разных источников, структурировать, очистить от ненужной информации и привести к общему формату.
3. Результат, полученный в результате работы алгоритмов ИИ сложно трактовать и понять с точки зрения логики принятия решений.
4. Модели ориентированы на решение определенных задач. Например, если алгоритм ИИ используется для обнаружения мошенничества конкретного вида, другие варианты мошенничества он распознавать не будет — для каждой задачи и каждых условий нужна своя модель.
5. Если исходный набор данных для обучения искажен или недостаточен, результаты работы ИИ могут быть искажены. Например, если в выборке для обучения используются только объекты красного цвета, при появлении синего объекта в процессе самообучения могут возникнуть ошибки или разногласия.
6. Для работы с ИИ и разработки проектов на его базе важно иметь достаточную компетенцию, позволяющую оценивать риски и принимать решения на каждом этапе внедрения алгоритмов.

Типы ИИ

Выделяют три вида искусственного интеллекта: ***Слабый ИИ*** (Narrow AI). Единственный доступный сейчас тип ИИ, применяемый в голосовых помощниках, системах виртуальной реальности, рекомендательных механизмах и других решениях.

Сильный ИИ (AGI). ИИ с самосознанием и возможностями, приближенными к человеческим. По прогнозам экспертов, сильный ИИ будет окончательно сформирован и доступен для использования не раньше 2075 года.

Супер ИИ (Super AI). ИИ с полным самосознанием и сформированным мышлением, превосходящим человеческое. Предположительно Super AI сможет самостоятельно перепрограммироваться, создавать системы нового направления и алгоритмы без вмешательства человека.

Artificial Super Intelligence (ASI) - гипотетический ИИ, который сможет не только воспроизводить максимум способностей человека, но и даже превзойти его.

Слабый, или узкий ИИ (Artificial Narrow Intelligence, ANI) позволяет усмотреть в поведении машин слабые намеки на разум (поэтому его называют слабым). Он предназначен для выполнения только строго определенного узкого круга приложений (поэтому его называют узким). В случае ANI невозможно никакое неподвластное человеку автономное поведение или самостоятельное развитие. Системы, снабженные ANI, могут существовать только в той форме, в которой они были созданы человеком и даже теоретически не могут выйти из-под его контроля.

Нейронные сети и типы задач, которые решают нейронные сети

В искусственном интеллекте классификация осуществляется по следующим типам систем:

1. Экспертные системы: используют знания экспертов в конкретной области для решения задач.

2. Нейронные сети: моделируют работу человеческого мозга для распознавания образов, анализа данных и обучения.

3. Генетические алгоритмы: применяют принципы эволюции и генетики для решения задач оптимизации.

4. Распознавание образов: анализирует и классифицирует визуальную информацию, например, лица или объекты.

5. Робототехника: объединяет аппаратные и программные компоненты для взаимодействия роботов с окружающей средой и выполнения задач.

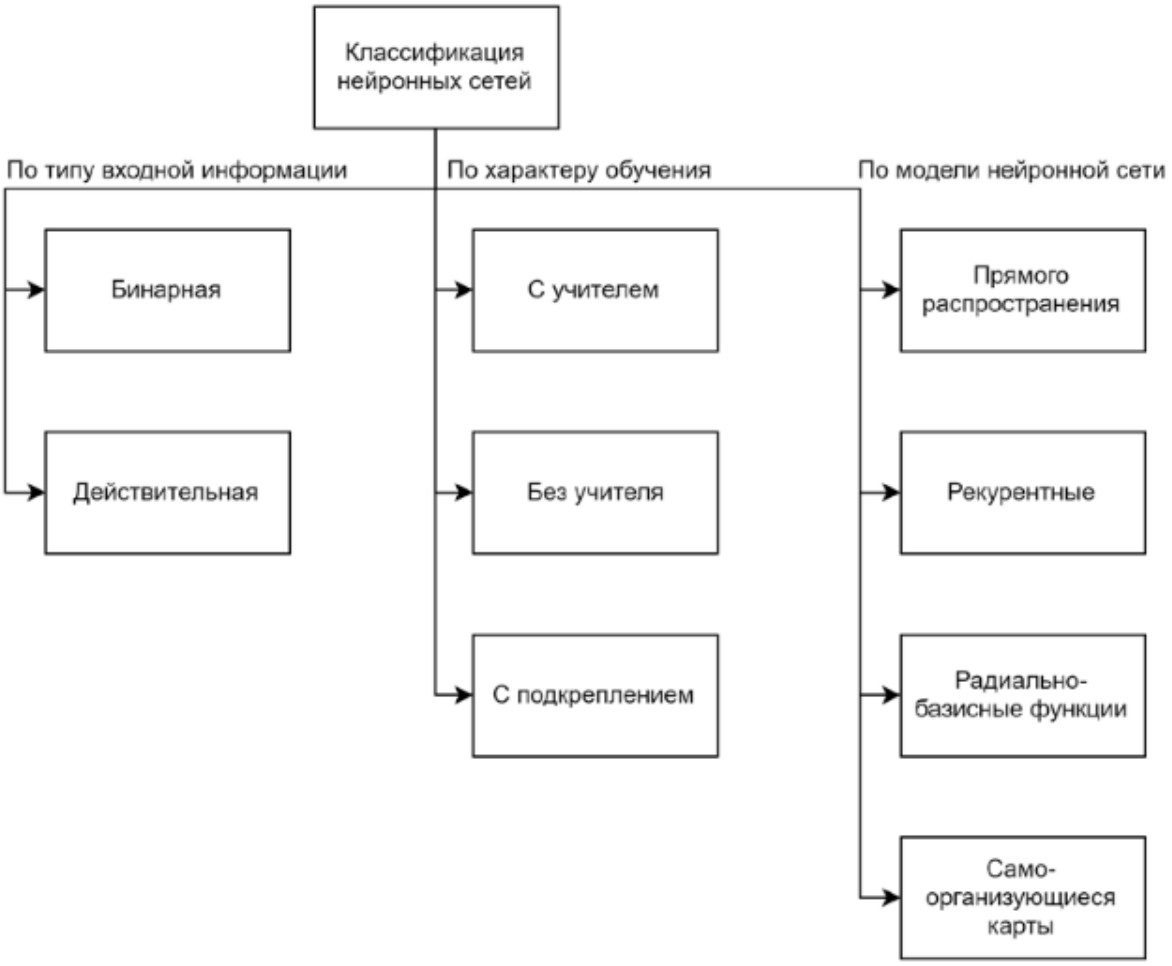


Рисунок 1

Выделяют несколько базовых типов задач, для решения которых могут использоваться нейросети.

- **Классификация.** Для распознавания лиц, эмоций, типов объектов: например, квадратов, кругов, треугольников. Также для распознавания образов, то есть выбора конкретного объекта из предложенного множества: например, выбор квадрата среди треугольников.
- **Регрессия.** Для определения возраста по фотографии, составления прогноза биржевых курсов, оценки стоимости имущества и других задач, требующих получения в результате обработки конкретного числа.

- **Прогнозирования временных рядов.** Для составления долгосрочных прогнозов на основе динамического временного ряда значений. Например, нейросети применяются для предсказания цен, физических явлений, объема потребления и других показателей. По сути, даже работу автопилота Tesla можно отнести к процессу прогнозирования временных рядов.
- **Кластеризация.** Для изучения и сортировки большого объема неразмеченных данных в условиях, когда неизвестно количество классов на выходе, то есть для объединения данных по признакам. Например, кластеризация применяется для выявления классов картинок и сегментации клиентов.
- **Генерация.** Для автоматизированного создания контента или его трансформации. Генерация с помощью нейросетей применяется для создания уникальных текстов, аудиофайлов, видео, раскрашивания черно-белых фильмов и даже изменения окружающей среды на фото.

Примечание: Например, нейронная сеть ruDALL-E, созданная Сбером, может генерировать уникальные изображения на основе текстового описания.

Есть десятки видов нейросетей, которые отличаются архитектурой, особенностями функционирования и сферами применения. При этом чаще других встречаются сети трех видов.

Нейронные сети прямого распространения (Feed forward neural networks, FFNN). Прямолинейный вид нейросетей, при котором соседние узлы слоя не связаны, а передача информации осуществляется напрямую от входного слоя к выходному. FFNN имеют малую функциональность, поэтому часто используются в комбинации с сетями других видов.

Сверточные нейронные сети (Convolutional neural network, CNN). Состоят из слоев пяти типов:

- входного;
- свертывающего;
- объединяющего;
- подключенного;
- выходного.

Каждый слой выполняет определенную задачу: например, обобщает или соединяет данные.

Сверточные нейросети применяются для классификации изображений, распознавания объектов, прогнозирования, обработки естественного языка и других задач.

Рекуррентные нейронные сети (Recurrent neural network, RNN). Используют направленную последовательность связи между узлами. В RNN результат вычислений на каждом этапе используется в качестве исходных данных для следующего. Благодаря этому, рекуррентные нейронные сети могут обрабатывать серии событий во времени или последовательности для получения результата вычислений.

RNN применяют для языкового моделирования и генерации текстов, машинного перевода, распознавания речи и других задач.

Тема 2.2. Модель искусственного нейрона. Общее представление об искусственной нейронной сети

Виды нейронов в нейронных сетях

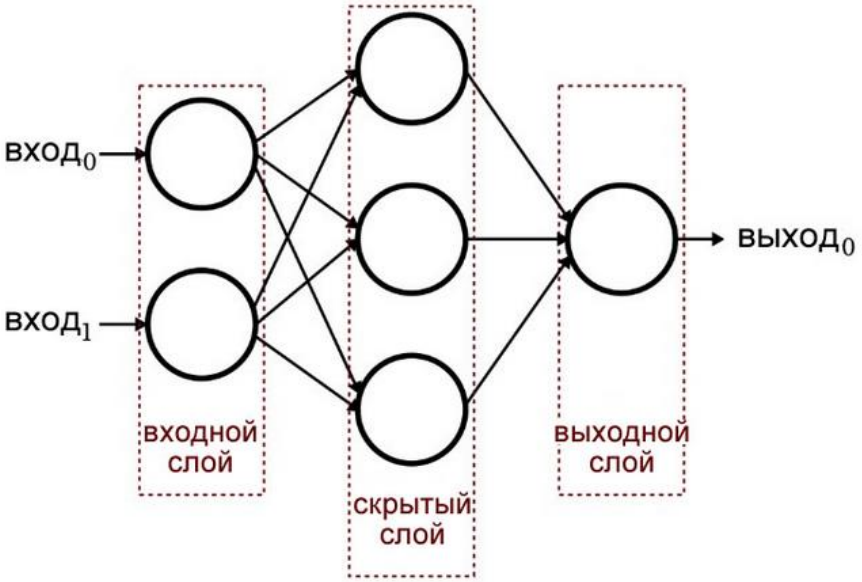


Рисунок 2 Нейронная сеть многослойный перцептрон, или MLP (multilayer perceptron)

Перцептрон с одним скрытым слоем –
Если вы не получаете адекватных результатов с одним скрытым слоем, попробуйте сначала другие усовершенствования – может быть, вам нужно оптимизировать скорость обучения, увеличить количество эпох обучения или расширить набор обучающих данных. Добавление второго скрытого слоя увеличивает сложность кода и время обработки.

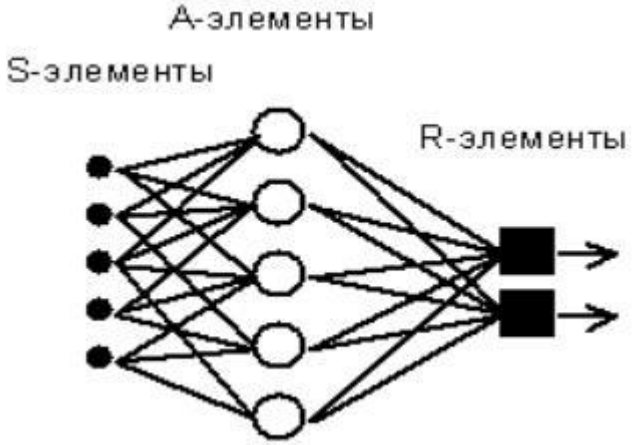


Рис. 4 Элементарный персептрон Розенблатта.

Функции активации в нейронных сетях

Пример 1 Написать программу, вычисляющую сигмоидальную функцию активации $f(x) = \frac{1}{1+e^{-x}}$

Листинг программы

```
import numpy as np
def sigmoid(x):
   return 1 / (1 + np.exp(-x))
```

По количеству слоев

1. **Однослойная структура нейронной сети**. Представляет собой структуру взаимодействия нейронов, в которой сигналы со входного слоя сразу направляются на выходной слой, который, собственно говоря, не только преобразует сигнал, но и сразу же выдаёт ответ.

2. **Многослойная нейронная сеть**. Здесь, помимо выходного и входного слоёв, имеются ещё несколько скрытых промежуточных слоёв. Число этих слоёв зависит от степени сложности нейронной сети. Она в большей степени напоминает структуру биологической нейронной сети.

Искусственные нейронные сети состоят из нескольких слоев:
входных;
скрытых;
выходных.
В каждом из них есть несколько узлов, которые соединены со всеми узлами в сети с помощью разных связей и имеют свой «вес», влияющий на силу передаваемого сигнала.
Такая архитектура позволяет вести параллельную обработку данных и постоянно сравнивать их с результатами обработки на каждом из этапов.
Нейронные сети изначально обучаются на размеченных наборах данных с очевидными закономерностями, а после используют полученные навыки для самообучения и достижения результата.
При этом нейросеть может совершать миллионы попыток для достижения таких же результатов, как и предоставленном для обучения примере.

***По направлению распределения информации по синапсам между нейронами*:**

1. **Нейросети прямого распространения (однонаправленные)**. В этой структуре сигнал перемещается строго по направлению от входного слоя к выходному. Движение сигнала в обратном направлении не осуществляется и в принципе невозможно. Сегодня разработки этого плана распространены широко и на сегодняшний день успешно решают задачи распознавания образов, прогнозирования и кластеризации.

2. **Рекуррентные нейронные сети (с обратными связями)**. Здесь сигнал двигается и в прямом, и в обратном направлении. В итоге результат выхода способен возвращаться на вход. Выход нейрона определяется весовыми характеристиками и входными сигналами, плюс дополняется

предыдущими выходами, снова вернувшимися на вход. Этим нейросетям присуща функция кратковременной памяти, на основании чего сигналы восстанавливаются и дополняются во время их обработки.

3. **Радиально-базисные функции**.

4. **Самоорганизующиеся карты**.

В зависимости от типов нейронов:

— однородные; — гибридные.

В зависимости от метода нейронных сетей по обучению:

— обучение с учителем;

— без учителя;

— с подкреплением.

По типу входной информации нейронные сети бывают:

— аналоговые; — двоичные; — образные.

По характеру настройки синапсов:

— с фиксированными связями;

— с динамическими связями.

Ещё существуют понятия гетероассоциативные или автоассоциативные нейросети.

Тема 2.3. Библиотеки для обучения нейронных сетей (Keas, Tensor Flow, PyTorch, Scikit-learn)

Тема 2.4. Обучение искусственного нейрона

Типы обучения нейронных сетей

Нейросети, в отличие от других алгоритмов ИИ, не программируются на выполнение конкретных задач, а просто настраиваются на изучение информации.

Стратегия обучения нейронных сетей базируется на трех методах:

- Контролируемое обучение. Классическая модель обучения, в которой используется набор размеченных данных, показывающий алгоритму что и как должно быть. Обучение продолжается до полного перестраивания алгоритма под решение конкретных задач и получения нужного результата.
- Обучение без контроля. Стратегия обучения, применяемая в ситуациях, когда нет размеченных наборов данных. В этой модели нейронная сеть выполняет анализ, а после получает внутренний отчет о точности расчета. Если значение недостаточно, нейронная сеть усиливается и повторяет операцию.

- Усиленное обучение. Модель, при которой нейронная сеть усиливается при получении положительного результата и наказывается за неправильные расчеты.

Методы обучения искусственного интеллекта

Чтобы ИИ мог выполнять поставленные задачи, он должен быть предварительно обучен на реальных или схожих к ним задачах. Для этого используется два метода:

- машинное обучение;
- глубокое обучение.

Машинное обучение — способ обучения системы или машины программирования и строгого контроля.

Машинное обучение делится три типа:

- контролируемое — обучение на наборах размеченных данных с очевидными закономерностями;
- неконтролируемое — обучение на наборах неразмеченных данных без явных закономерностей;
- обучение с подкреплением — последовательное обучение на размеченных и неразмеченных наборах данных.

Для машинного обучения достаточно небольшого объема данных. Процесс обучения разделяется на небольшие этапы, результаты каждого из которых объединяются в блок выходных данных. Признаки должны быть точно определены и созданы пользователем. Выходные данные представляются в виде числа: оценки или классификации.

Глубокое обучение — разновидность машинного обучения с использованием нейронных сетей. Процесс обучения разделен на несколько этапов и имеет структуру с несколькими вводными, скрытыми и выходными слоями.

Модель учится не на исходном наборе данных, а на результатах обучения, полученных на предыдущем этапе — то есть каждый завершенный цикл использует в качестве учителя.

Глубокое обучение отличается от машинного алгоритмом — часть процесса скрыта и не имеет очевидной логики, а пользователь получает результат, который может быть представлен в любом формате: текст, аудио, число.

Для глубокого обучения нужен большой объем размеченных исходных данных и мощное вычислительное оборудование.

Классная работа

Домашняя работа

Практическая работа к теме 6

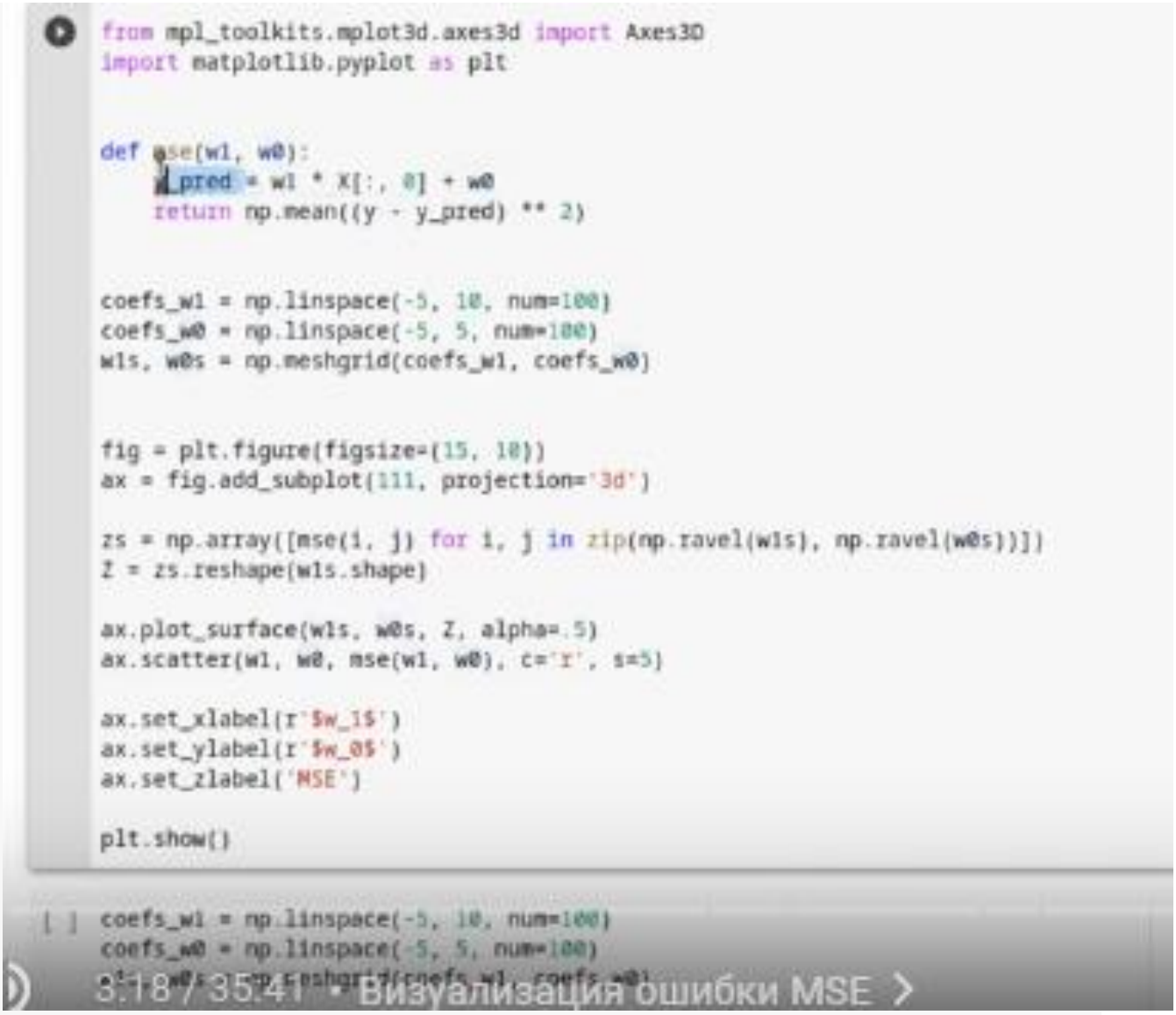
https://youtu.be/dokH3xc26iY?si=kG9tXdBiTRxHkl6K

Пример 1

```
from keras.layers import dense
from keras.models import Sequential

model=Sequential([
   Dense(1,input_shape(1,),activation='relu')
])
model.summary()
```

Пример 2 Нейрон для умноженеия на 3

```
import numpy as np
X=np.array([[1],[2],[3],[4],[5],[6],[7],[8]])
Y=np.array([1,6,9,12,15,18,21,24])
from keras.layers import dense
from keras.models import Sequential

model=Sequential([
   Dense(1,input_shape(1,),activation='linear')
])
model.summary()
model.predict(x[:1])
print(w1*x[:1]+w0)
```



Coefs_w1=np.lincpace(-5,10,num=100)
From mpl_toolkits.mplot3d.axes3d import Axes3D
Import matplotlib.pyplot as plt
Def use(w1,w0):
    Y_prep*w1*x[:,0]+w0
    Return np.mean((y-y_prep)**2)
Koef_w1=linspase(-5,10, num=100)
Wls,w0s=np.meshgrid(coefs_w1,coefs_w0)
Fig=plt.figure(figsize=(15,10))
Ax=fig.add_subplot(111,projection=’3d’)
Zs= np.array([mse(I,j) for I,j in zip(np.ravel(wls),np.ravel(wos))])
Z=zs.reshape(wls.shape)
Ax.plot_surface(wls,wos,Z,alpha=.5)
Ax.scatter(w1,w0,nse(w1,w0), c=’r’,s=5)
Ax.set_xlabel(r’w_1’)
Ax.set_ylabel(r’w_0’)
Ax.set_zlabel(‘MSE’)
Plt.show()

```
coefs_w0 = np.linspace(-5, 5, num=100)
w1s, w0s = np.meshgrid(coefs_w1, coefs_w0)

zs = np.array([round(mse(i, j)) for i, j in zip(np.ravel(w1s), np.ravel(w0s))])
Z = zs.reshape(w1s.shape)

fig = plt.figure(figsize = (10,7))
plt.imshow(Z, extent=[-5,10, -5,5], origin = 'lower', cmap = 'jet', alpha = 1)
plt.colorbar()
plt.scatter(w1, w0, c='r', s=15, label='start weights')


plt.xlabel('w1', fontsize=11)
plt.ylabel('w0', fontsize=11)

plt.legend(loc="upper right");
```

▾ Вручную

Реализуем две функции:

1. mserror - функция среднеквадратичной ошибки

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - y_pred_i)^2 = \frac{1}{n}\sum_{i=0}^{n}(y_i - (w_1 \cdot X_i + w_0))^2 = \frac{1}{n}\sum_{i=0}^{n}(y_i - w_1 \cdot X_i - w_0)^2$$

2. gr_mserror - градиент функции MSE. Распишем его отдельно для весов:

w_1: $\frac{\partial MSE}{\partial w_1} = \frac{1 \cdot 2}{n}\sum(y_i - y_pred_i) \cdot -X$

w_0: $\frac{\partial MSE}{\partial w_0} = \frac{1 \cdot 2}{n}\sum(y_i - y_pred_i) \cdot -1$

```
# функция, определяющая среднеквадратичную ошибку
def mserror(X, w1, w0, y):
    y_pred = w1 * X[:, 0] + w0
    return np.sum((y - y_pred) ** 2) / len(y_pred)

# функция градиента
def gr_mserror(X, w1, w0, y):
    y_pred = w1 * X + w0
    return {'grad_w1': 2/len(y)*np.sum((y - y_pred) * (-X)),
            'grad_w0': 2/len(y)*np.sum((y - y_pred)) * (-1)}
```

Перед тем, как считать градиенты и перед тем, как запускать градиентный спуск, давайте посчитаем ошибку

```
#Функция, определяющая среднеквадратическую ошибку
Def mserror(x,w1,w0,y):
        Y_pred=w1*x[:0]+w0
        Return np.sum((y-y_pred)**2)/len(y_pred)
#Функция градиента
Def gr_mserror(x,w1,w0,y):
        Y_pred=w1*x+w0
        Return{‘grad_w1’::2/len(y0)*np.sum((y-y_pred)*(-x)),
        ‘pred_w0’:2/len(y)*np((y-y_pred))*(-1)}
```

```
Print(model.predict(x[:1])
Print(w1*x[:1]+w0)
From keras.activations.import linear
Print(linear(w1*x[:1])+w0))
```

```
[14] model.predict(X[:1])

     array([[0.03501177]], dtype=float32)

[ ] w1 * X[:1] + w0

[ ] from keras.activations import linear
    linear(w1 * X[:1] + w0)
```

```
# функция, определяющая среднеквадратичную ошибку
def mserror(X, w1, w0, y):
    y_pred = w1 * X[:, 0] + w0
    return np.sum((y - y_pred) ** 2) / len(y_pred)

# функция градиента
def gr_mserror(X, w1, w0, y):
    y_pred = w1 * X + w0
    return {'grad_w1': 2/len(y)*np.sum((y - y_pred) * (-X)),
            'grad_w0': 2/len(y)*np.sum((y - y_pred)) * (-1)}
```

```
# функция, определяющая среднеквадратическую ошибку
Def mserror(x,w1,w0,y):
Y_pred=w1*x[:,0]*w0
Return np.sum((y-y_pred)**2)/len(y_pred)
#функция градиента
Def gr_mseerror(x,w1,w0,y):
        Y_pred=w1*x+w0
Return{‘grad_w1’:2/len(y)*np.sum((y-y_pred)*(-x)),’grad_w0’:2/len(y)*np((y-y_pred))*(-1))
```

Пример 3

```
Import numpy as np
Input matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow import keras
from tensorflow.keras.layers import Dense
#обучающая выборка
```

```
c=np.array([-40,-10,0,15,22,38])
f=np.array([-40,14,32,46,59,72,100])
model=keras.Sequenial()
model.add(Dense(units=1, input_shape(1,), activation='linear'))
model.compile(loss='mean_squared_errort',optimizer=keras.optimizers.adam(0.1)
)
history = model.fit(c,f,epochs=500,verbose=False)
print('обучение завершено')
print(model.predict([100]))
print(model.get_weights())
pltplot(log.history(['loss'])
plt.grid(true)
plt.show()
```

Тема 2.5. Обучение искусственной нейронной сети. Метод обратного распространения ошибки

Выбор модели

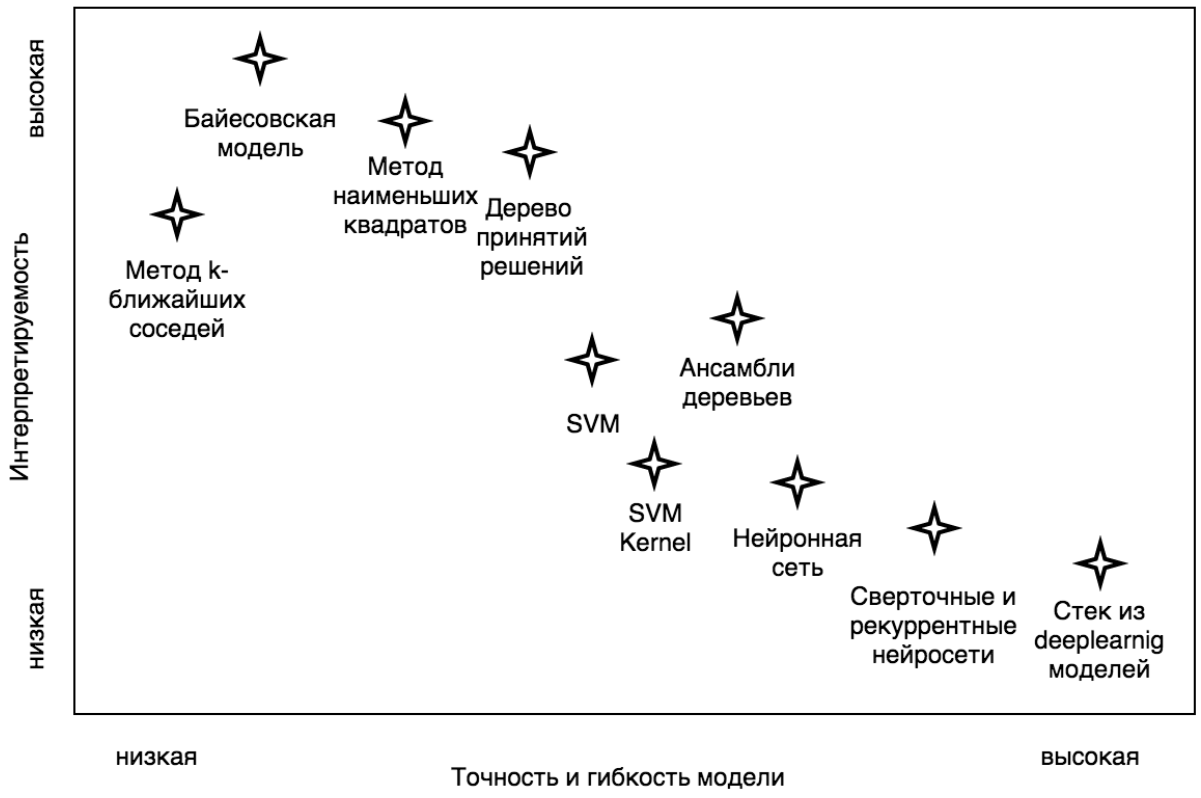


Рис. 1 — Зависимость гибкости алгоритма машинного обучения и интерпретируемости полученной модели

Тема 2.6. Применение нейронных сетей для решения задачи классификации

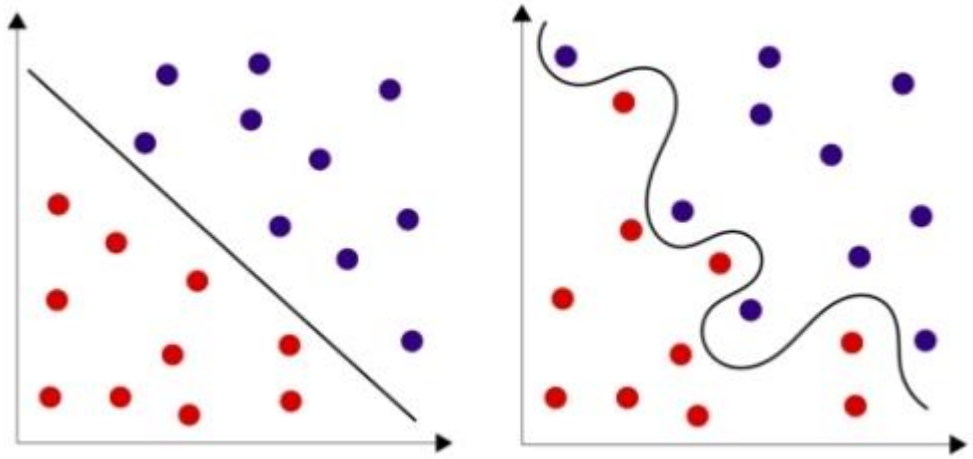


Рисунок 3

Тема 2.7. Применение нейронных сетей для решения задачи регрессии

Тема 2.8. Анализ качества обучения нейронной сети

Тема 2.9 Контрольная работа 2

МОДУЛЬ 3. КОМПЬЮТЕРНОЕ ЗРЕНИЕ

Компьютерное зрение (CV)

находят, отслеживают, классифицируют, идентифицируют объекты, извлекают данные из изображений, анализируют полученную информацию

Применяется для распознавания объектов видео аналитики, описания содержания изображений и видео, распознавания жестов и рукописного ввода, интеллектуальной обработки изображений.

Тема 3.1. Оцифровка изображений

ЗАГРУЗКА ИЗОБРАЖЕНИЯ И ПОЛУЧЕНИЕ ИНФОРМАЦИИ О НЕМ.

Цель работы: ознакомление с основными методами загрузки изображений и получение общей информации о нем с использованием внешней библиотеки PIL.

Подготовка к выполнению заданий.

На компьютере должен быть выход в интернет.

Поместите в текущую папку набор цветных изображений.

Поместите не в текущую папку набор других цветных изображений.

Установите внешнюю библиотеку PIL (Pillow).

Задание 1. Загрузка изображения.

Подключите модуль Image из библиотеки PIL. Для этого напишите и запустите на выполнение команду:

from PIL **import** Image

Откройте файл, расположенный в текущем рабочем каталоге. Для этого напишите и запустите на выполнение команду:

img = Image.open("img1.jpg**")**

выводим изображение на экран:

img.show()

Откройте этот же файл, расположенный в текущем рабочем каталоге, в бинарном режиме. Для этого напишите и запустите на выполнение команду:

img = Image.open("Tulips.jpg") img = img.convert("L")

а в более ранних, чем Release 7.2.0.dev0 Pillow выполняем команду

img = Image.open("img1.jpg", "rb") # выводим изображение на экран:

img.show()

Вместо указания пути к файлу можно передать файловый объект, открытый в бинарном режиме. Для этого напишите и запустите на выполнение команды:

Открываем файл в бинарном режиме

f = open("img2.gif**", "rb")**

Передаем объект файла

img = Image.open(f)

выводим переданное изображение на экран:

img.show()

f.close() # Закрываем файл

Загрузка изображения из строки. Модуль StringIO . Сначала откройте файл в бинарном режиме:

f = open("img2.gif**", "rb")**

Сохраните изображение в переменной:

i = f.read()

f.close() # Закрываем файл # Подключаем модуль StringIO:

import StringIO

Передаем объект:

img = Image.open(StringIO.StringIO(i))

выводим изображение на экран:

img.show()

Задание 2. Получение информации об изображении

Получите полную информацию об открытом изображении. Для этого выполните следующее:

```
>>> img = Image.open("foto.jpg")
>>> img.size, img.format, img.mode
((800, 600), 'JPEG', 'RGB')
>>> img.info
{'jfif': 258, 'jfif_unit': 0, 'adobe': 100, 'progression': 1,
'jfif_version': (1, 2), 'adobe_transform': 100,
'jfif_density': (100, 100)}
```

Получите информацию о цветовой модели изображения.

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
```

Получите информацию о размере изображения.

img.size

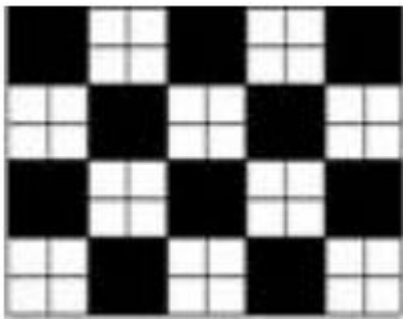
В консоль выведется размер повернутого на 90 градусовизображения:

(600, 800)

Получите координаты прямоугольной области, в которую вписывается всё изображение **img .**

img.getbbox() (0, 0, 800, 600)

***Подсчитать информационный объём изображения, необходимый для хранения чёрно-белого изображения вида* (в битах)**



Задание 3. Ручное кодирование изображений

Таблица 3

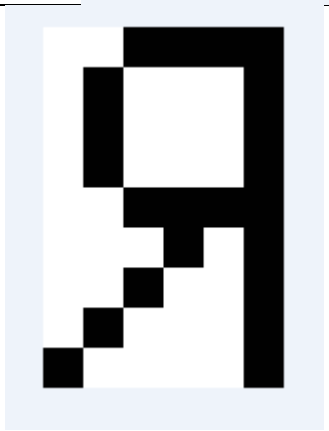
						0	**0**	**0**	**0**	**0**
						0	**1**	**1**	**1**	**0**
						0	**1**	**0**	**1**	**0**
						0	**1**	**1**	**1**	**0**
						0	**1**	**0**	**1**	**0**
						0	**1**	**0**	**1**	**0**
						0	**0**	**0**	**0**	**0**

Таблица 4

										0	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
										0	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
										0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
										1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
										0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
										0	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**
										0	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
										0	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**
										0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
										0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**

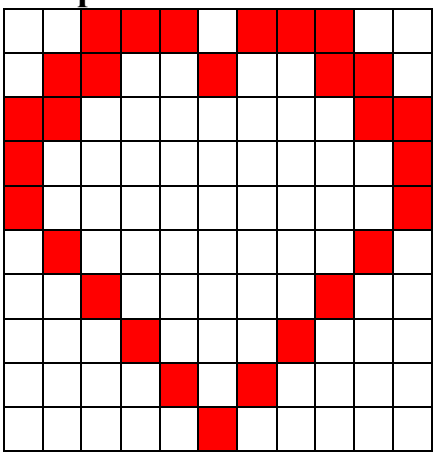
Задание 4. Матричная запись изображений

```
import numpy as np
from PIL import Image
labels=[[0,0,0,1,1,1, 0,0,0],
        [0,0,1,0,0,0,1,0, 0],
         [0,1,0,0,0,0,0,0,1,0],
         [1,1,1,1,1,1,1,1,1],
         [0,1,0,0,0, 0, 0,1, 0],
         [0,1,0,1,1,1,0,1,0],
         [0,1,0,1,0,1,0,1,0],
         [0,1,0,1,1,1,0,1,0],
          [0,1,0,0,0,0,0,1,0],
         [1,1,1,1,1,1,1,1,1]]
binary_transform = np.array(labels).astype(np.uint8)
binary_transform[binary_transform <= 0] = 255

img = Image.fromarray(binary_transform, 'L')
img.save('domik.png')
```

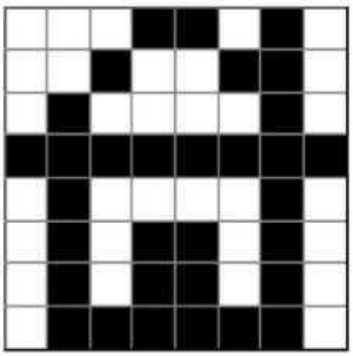


```
import numpy as np
from PIL import Image
labels=[[0,0,1,1,1,1],
      [0,1,0,0,0,1],
       [0,1,0,0,0,1],
       [0,1,0,0,0,1],
       [0,0,1,1,1,1],
       [0,0,0,1,0,1],
       [0,0,1,0,0,1],
       [0,1,0,0,0,1],
[1,0,0,0,0,1]]
binary_transform =
np.array(labels).astype(np.uint8)
binary_transform[binary_transform <= 0] = 255

img = Image.fromarray(binary_transform, 'L')
img.save('image.png')
```

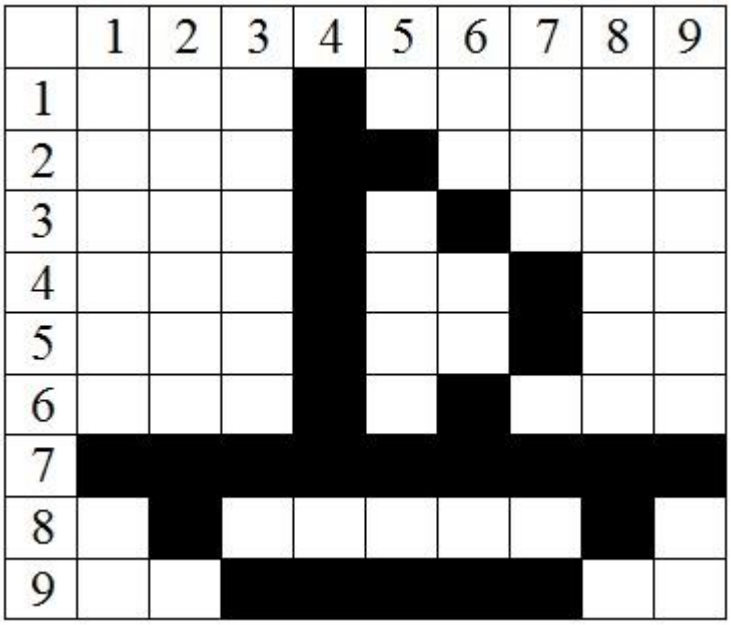
Задачи 3-5 для самостоятельного решения **Матричная запись изображений**

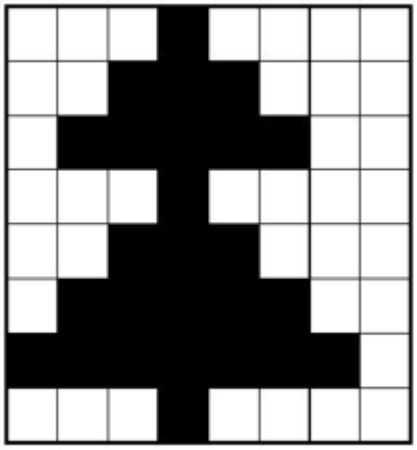


Раздаточный материал по теме матричная запись изображений на Python



0	0	0	1	1	0	1	0
0	0	1	0	0	1	1	0
0	1	0	0	0	0	1	0
1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	0
0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0





0 0 0 1 0 0 0 0
0 0 1 1 1 0 0 0
0 1 1 1 1 1 0 0
0 0 0 1 0 0 0 0
0 0 1 1 1 0 0 0
0 1 1 1 1 1 0 0
1 1 1 1 1 1 1 0
0 0 0 1 0 0 0 0

ЦВЕТОВЫЕ МОДЕЛИ.
СОЗДАНИЕ И СОХРАНЕНИЕ ИЗОБРАЖЕНИЯ.

Цель работы: ознакомление со способами создания и сохранения изображения, а так же с цветовыми моделями в Python с использованием внешней библиотеки PIL.

Подготовка к выполнению заданий.

1. На компьютере должен быть выход в интернет.
2. Поместите в текущую папку набор цветных изображений.
3. Поместите не в текущую папку набор других цветных изображений.
4. Установите внешнюю библиотеку PIL.

Задание 1. Получение и изменение цвета пиксела.

1. Получите и измените цвет пиксела методом **load()**.

Сначала получите информацию о цветовой модели открытого изображения:

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
```

Теперь получите цвет пиксела:

obj = img.load()
obj[25, 45] # Получаем цвет пикселя
(122, 86, 62)
Задаем цвет пикселя (красный)
obj[25, 45] = (255, 0, 0)
img.show() # Просматриваем изображение (цвет пикселя будет изменен на obj[25, 45], заданный выше.

2. Получите и измените цвет пиксела методами **gel()** и putpixel().

# Получаем цвет пикселя
img.getpixel((25, 45))
(255, 0, 0)
Изменяем цвет пикселя **img.putpixel((25, 45), (0, 0, 255))** # Получаем цвет пикселя **img.getpixel((25, 45))**
(0, 0, 255)
img.show() # Просматриваем изображение

Задание 2. Переход от одной цветовой модели изображения к другой

1. Преобразуйте изображение из режима **RGB в RGBA, используя split() и merge .**

split() – возвращает каналы изображения в виде кортежа.

merge(<Режим>, <Каналы>) – собирает изображение из каналов (операция, обратная **split).**

2. Преобразуйте изображение из режима **RGB в RGBA,**

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
>>> R, G, B = img.split()
>>> mask = Image.new("L", img.size, 128)
>>> img2 = Image.merge("RGBA", (R, G, B, mask) )
>>> img2.mode
'RGBA'
>>> img2.show()
```

используя convert().

img = Image.open("foto.jpg") img.mode

'RGB'

img2 = img.convert("RGBA") img2.mode

'RGBA'

img2.show()

#прочитать изображение и конвертировать его в градациях серого:

pil_im = Image.open('empire.jpg').convert('L')

3. Преобразуйте изображение **RGB** в формат **P** , указав смешивание цветов и адаптивную палитру в 128 цветов.

Задание 3. Сохранение изображения.

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
>>> img2 = img.convert("P", None,
Image.FLOYDSTEINBERG, Image.ADAPTIVE, 128)
>>> img2.mode
'P'
```

Сохраните изображение в текущую папку.

Сохранение изображения в текущую папку в формате **JPEG**: **img.save("tmp.jpg")**

в формате BMP:

img.save("tmp.bmp", "BMP")

f = open("tmp2.bmp", "wb") # Передаем файловый объект **img.save(f, "BMP")**

f.close()

Задание 4. Создание нового изображения

1. Создайте черный квадрат.

img = Image.new("RGB", (100, 100)) img.show()# Черный квадрат

2. Создайте красный квадрат.

img = Image.new("RGB", (100, 100), (255, 0, 0))

img.show() # Красный квадрат

3. Создайте Зеленый квадрат.

img = Image.new("RGB", (100, 100), "green") img.show() # Зеленый квадрат

4. Создайте красный квадрат.

img = Image.new("RGB", (100, 100), "#f00")

img.show() # тоже красный квадрат

5. Создайте белый квадрат.

img = Image.new("RGB", (100, 100), "white") img.show() # Белый квадрат

6. Создайте серый квадрат.

img = Image.new("RGB", (320, 240), "silver") img.show() # Серый квадрат

7. Создайте цветной прямоугольник.

img = Image.new("RGB", (320, 240), "rgb(205,**100,200)")**

img.show() # цветной прямоугольник

8. Создайте цветной прямоугольник (Каналы в процентах).

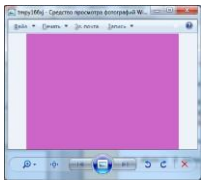
img = Image.new("RGB", (320, 240), "rgb(10%, **100%,40%)")**

img.show() # цветной прямоугольник. Каналы в процентах

9. Создайте цветной прямоугольник заданного цвета. Затем поменяйте этот цвет.

img = Image.new("RGB", (640, 480), "rgb(205,**100,200)")**

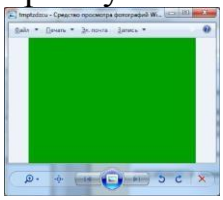
img.show() # сиреневый прямоугольник



for x in xrange(640):

for y in xrange(480): img.putpixel((x,y),(0,160,0))

img.save("okno.png", "PNG") img.show() # зеленый прямоугольник



10. Создайте прямоугольник заданного цвета (функциональная раскраска).

img = Image.new("RGB", (640, 480), "rgb(205,**100,200)")**

img.show() # сиреневый прямоугольник

for x in xrange(640): for y in xrange(480):

img.putpixel((x,y),
x/3,(x+y)/6,y/3)) img.save("okno.png", "PNG")
img.show() # прямоугольник с функциональной раскраской

Тема 3.2. Сверточные нейронные сети

Сверточные нейронные сети (Convolutional Neural Networks, CNN) — особые типы нейронных сетей, которые помогают компьютерам видеть и понимать изображения и видео.

Такие сети имеют несколько слоев, называемых сверточными. Они позволяют CNN изучать сложные особенности и делать более точные предсказания о содержимом визуальных материалов.

Сверточные нейросети применяются в том числе для:

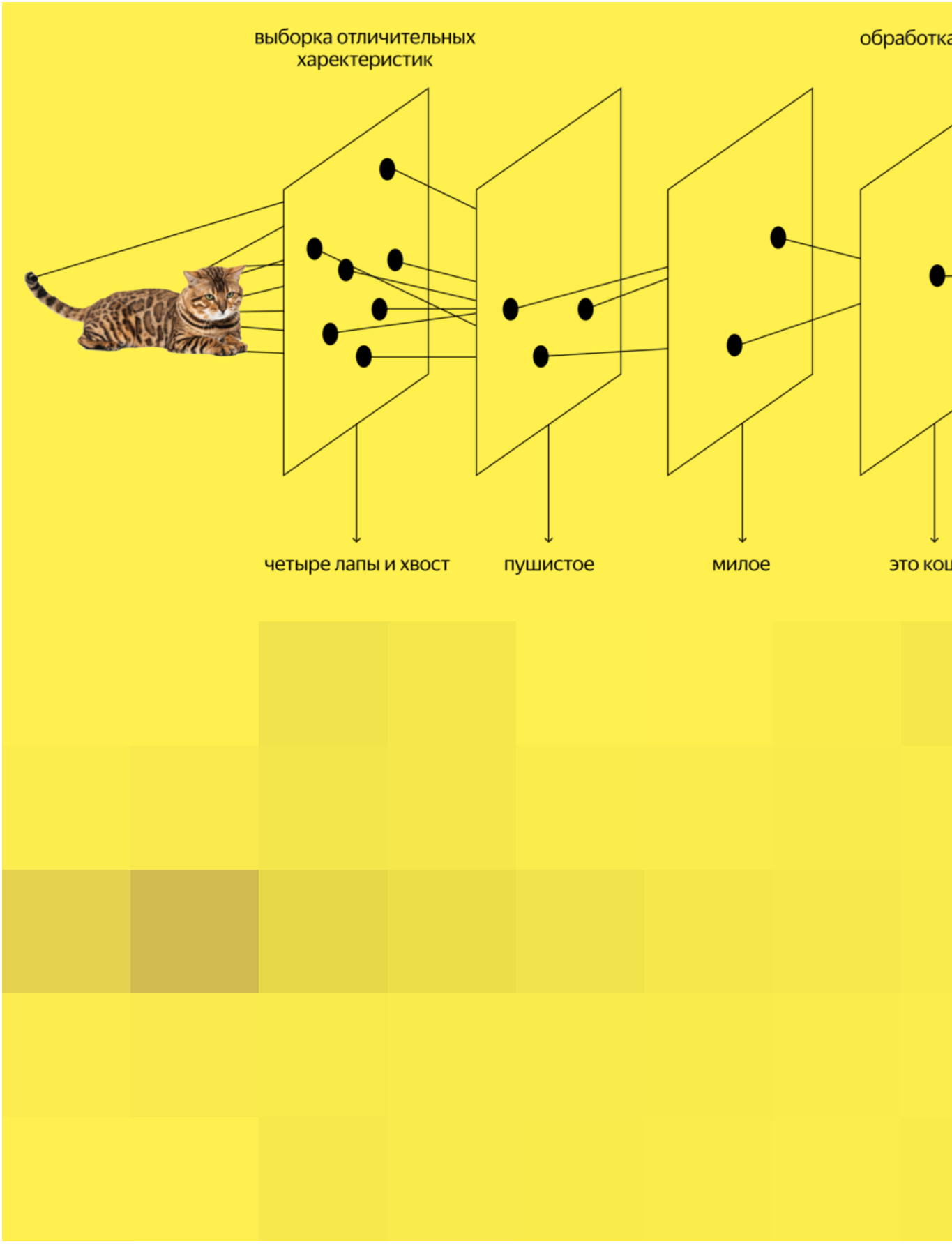
- распознавания лиц;
- автопилотирования;
- медицинского прототипирования;
- обработки естественного языка.

Классификация изображений.

Наборы данных

1. MNIST – набор изображений рукописных цифр (10 классов) [http://yann.lecun.com/exdb/mnist]
2. Fashion MNIST – изображения одежды разного фасона (10 классов) [https://github.com/zalandoresearch/fashion-mnist]
3. Cifar-10 / Cifar-100 – изображения объектов естественного мира (10 / 100 классов) [http://www.cs.utoronto.ca/~kriz/cifar.html]
4. Caltech-101 / Caltech-256 – изображения объектов естественного мира (101 / 256 классов) [http://www.vision.caltech.edu/Image_Datasets/Caltech101] / [http://www.vision.caltech.edu/Image_Datasets/Caltech256]
5. ImageNet – набор изображений, принадлежащих 1000 категориям естественного мира [http://www.image-net.org]

Чаще всего с помощью свёрточных нейросетей можно решать две задачи: распознавание и классификацию. Например, с помощью нейросети на изображении кошки можно определить её окрас — по большому фрагменту изображения, цвет глаз — по более маленьким фрагментам или отличить кошку от собаки.



Нейронная сеть распознаёт кошку

AlexNet – первая глубокая свёрточная нейронная сеть, которая стала основой для современных архитектур. Разработчики сети выиграли конкурс по

классификации изображений LSVRC-2012 на наборе данных ImageNet.

Есть несколько датасетов, или наборов данных, на которых новички могут с помощью свёрточных нейросетей тренироваться решать задачи по классификации. MNIST — это база, с которой обычно начинают изучение свёрточных нейронных сетей. Инструмент можно найти в наборе Google Colab и попробовать за полчаса научить нейросеть распознавать рукописные цифры.

Ещё один вариант — ImageNet. В этом датасете 1000 разных классов, а кроме кошек и собак, есть, например, вишня, дерево или фотоаппарат. На курсе «Специалист по Data Science» студенты работают с конкретным датасетом для проекта по компьютерному зрению. Например, строят и обучают модель, которая на основе массива данных будет определять возраст человека по фото.

Тема 3.3. Распознавание объектов на изображении

Тема 3.4. Предварительно обученные нейронные сети

Фильтры для изображений
https://blog.aspose.com/ru/imaging/apply-median-wiener-filter-to-images-in-python/

Библиотека Python для применения фильтров медианных и винеровских изображений
Чтобы применить медианный и винеровский фильтры к изображениям, мы будем использовать Aspose.Imaging for Python — мощную библиотеку обработки изображений, позволяющую без труда манипулировать изображениями. Чтобы использовать библиотеку, вы можете либо скачать, либо установить ее с помощью следующей команды.
> pip install aspose-imaging-python-net
Применение медианного фильтра к изображению в Python
Медианный фильтр является широко используемым методом шумоподавления, в котором используется метод нелинейной цифровой фильтрации. Ниже приведены шаги по применению медианного фильтра к изображению в Python.
Сначала загрузите изображение с помощью метода Image.load().
Затем приведите изображение к типу RasterImage.
Создайте экземпляр класса MedianFilterOptions и инициализируйте его размером прямоугольника.
Примените медианный фильтр с помощью метода RasterImage.filter(Rectangle, MedianFilterOptions).
Наконец, сохраните отфильтрованное изображение с помощью метода RasterImage.save().

В следующем примере кода показано, как применить медианный фильтр к изображению в Python.

```
import aspose.pycore as aspycore
from aspose.imaging import Image, RasterImage
from aspose.imaging.imagefilters.filteroptions import MedianFilterOptions
import os


if 'TEMPLATE_DIR' in os.environ:
	templates_folder = os.environ['TEMPLATE_DIR']
else:
	templates_folder = r"C:\Users\USER\Downloads\templates"

delete_output = 'SAVE_OUTPUT' not in os.environ
data_dir = templates_folder
# Загрузите зашумленное изображение
with Image.load(os.path.join(data_dir, "template.gif")) as image:
	# Вставьте изображение в RasterImage
	if aspycore.is_assignable(image, RasterImage):
		raster_image = aspycore.as_of(image, RasterImage)
		# Создайте экземпляр класса MedianFilterOptions и установите размер, примените фильтр MedianFilterOptions к объекту RasterImage и сохраните полученное изображение.
		options = MedianFilterOptions(4)
		raster_image.filter(image.bounds, options)
		image.save(os.path.join(data_dir, "result.gif"))

if delete_output:
	os.remove(os.path.join(data_dir, "result.gif"))
```

Ниже представлено изображение до и после применения медианного фильтра.

Примените фильтр Винера Гаусса к изображению в Python

Винер Гаусса — еще один широко используемый метод повышения четкости и уменьшения шума на изображениях. Давайте посмотрим на шаги, необходимые для применения фильтра Винера Гаусса к изображению в Python.

Сначала загрузите изображение с помощью метода Image.load().

Затем приведите изображение к типу RasterImage.

Создайте экземпляр класса GaussWienerFilterOptions и инициализируйте его размером радиуса и значением Smooth.

(Необязательно) Чтобы получить изображение в градациях серого, установите для свойства GaussWienerFilterOptions.grayscale значение true.

Примените фильтр Винера Гаусса, используя метод RasterImage.filter(Rectangle, GaussWienerFilterOptions).

Наконец, сохраните полученное изображение с помощью метода RasterImage.save().
В следующем примере кода показано, как применить фильтр Винера Гаусса к изображению в Python.

```
import aspose.pycore as aspycore
from aspose.imaging import Image, RasterImage
from aspose.imaging.imagefilters.filteroptions import GaussWienerFilterOptions
import os


if 'TEMPLATE_DIR' in os.environ:
	templates_folder = os.environ['TEMPLATE_DIR']
else:
	templates_folder = r"C:\Users\USER\Downloads\templates"

delete_output = 'SAVE_OUTPUT' not in os.environ
data_dir = templates_folder
# Загрузите изображение
with Image.load(os.path.join(data_dir, "template.jpg")) as image:
	# Вставьте изображение в RasterImage
	if aspycore.is_assignable(image, RasterImage):
		raster_image = aspycore.as_of(image, RasterImage)
		# Создайте экземпляр класса GaussWienerFilterOptions и установите размер радиуса и значение сглаживания.
		options = GaussWienerFilterOptions(12, 3)
		options.grayscale = True
		# Примените фильтр MedianFilterOptions к объекту RasterImage и сохраните полученное изображение.
		raster_image.filter(image.bounds, options)
		image.save(os.path.join(data_dir, "result.jpg"))

if delete_output:
	os.remove(os.path.join(data_dir, "result.jpg"))
```

Ниже приведено изображение до и после применения фильтра Винера Гаусса с опцией оттенков серого.

Фильтр Motion Wiener для изображения в Python

Винеровский фильтр движения используется для уменьшения размытия или ухудшения качества, вызванного размытием изображения в движении. Такое размытие происходит из-за относительного движения между камерой и объектом. Ниже приведены шаги по применению фильтра Винера движения к изображению в Python.
Сначала загрузите изображение с помощью метода Image.load().
Затем приведите изображение к типу RasterImage.

Создайте экземпляр класса MotionWienerFilterOptions и инициализируйте его длиной, значением сглаживания и углом.
Примените фильтр Motion Wiener с помощью метода RasterImage.filter(Rectangle, MotionWienerFilterOptions).
Наконец, сохраните полученное изображение с помощью метода RasterImage.save().
В следующем примере кода показано, как применить фильтр Винера движения к изображению в Python.

```
import aspose.pycore as aspycore
from aspose.imaging import Image, RasterImage
from aspose.imaging.imagefilters.filteroptions import MotionWienerFilterOptions
import os


if 'TEMPLATE_DIR' in os.environ:
	templates_folder = os.environ['TEMPLATE_DIR']
else:
	templates_folder = r"C:\Users\USER\Downloads\templates"

delete_output = 'SAVE_OUTPUT' not in os.environ
data_dir = templates_folder
# Загрузите изображение
with Image.load(os.path.join(data_dir, "template.gif")) as image:
	# Вставьте изображение в RasterImage
	if aspycore.is_assignable(image, RasterImage):
		raster_image = aspycore.as_of(image, RasterImage)
		# Создайте экземпляр класса MotionWienerFilterOptions и установите длину, значение сглаживания и угол.
		options = MotionWienerFilterOptions(50, 9, 90)
		options.grayscale = True
		# Примените фильтр MedianFilterOptions к объекту RasterImage и сохраните полученное изображение.
		raster_image.filter(image.bounds, options)
		image.save(os.path.join(data_dir, "result.gif"))

if delete_output:
	os.remove(os.path.join(data_dir, "result.gif"))
```

Бесплатная библиотека фильтров изображений Python
Вы можете получить бесплатную временную лицензию и применять медианный и винеровский фильтры к изображениям без ограничений оценки.
Бесплатное онлайн-приложение для редактирования изображений
Используйте наш бесплатный веб-инструмент для редактирования изображений, чтобы редактировать свои изображения в Интернете. Этот редактор изображений использует Aspose.Imaging for Python и не требует создания учетной записи.

Заключение

В этой статье вы узнали, как применять медианный фильтр и фильтр Винера к изображениям в Python. Шаги и примеры кода продемонстрировали, как применять различные типы фильтров для программного устранения шума в изображениях. Кроме того, мы рассмотрели, как уменьшить шум движущихся объектов на изображении с помощью фильтра Винера движения. Наконец, мы предоставили вам совершенно бесплатное онлайн-приложение для редактирования изображений, и вы можете использовать его без регистрации.

Вы можете узнать больше о библиотеке обработки изображений Python, используя документацию. Кроме того, вы можете поделиться с нами своими вопросами через наш форум.

Смотрите также

Изменить цвета пикселей изображений в Python

Объединение нескольких изображений в Python

Преобразование изображения в PDF в Python

Тема 3.5. Перенос обучения в нейронных сетях

Перенос обучения (Transfer Learning) — это метод в машинном обучении, который заключается в использовании знаний, полученных при решении одной задачи, для решения другой, схожей задачи.

Тема 3.6. Распознавание предметов одежды. Построение архитектуры нейронной сети и ее обучение.

Тема 3.7. Контрольная работа 3

МОДУЛЬ 4. АНАЛИЗ ЕСТЕСТВЕННОГО ЯЗЫКА

Тема 4.1. Нейронные сети для задач обработки естественного языка

Среди нейронных сетей для задач обработки естественного языка популярны три основных:

1) рекуррентные нейронные сети;
2) рекурсивные нейронные сети;
3) свёрточные нейронные сети.

Для анализа естественного языка в Python можно использовать следующие библиотеки:

1. NLTK (Natural Language Toolkit).
2. TextBlob.
3. spaCy.
4. Gensim.
5. TfidfVectorizer (из библиотеки scikit-learn).

Также можно применять различные методы анализа текста, такие как:

- анализ частоты слов;
- извлечение ключевых слов и фраз;
- определение тональности текста (позитивный, негативный, нейтральный);
- анализ коллокаций и групп слов;
- создание облака тегов;
- анализ тематики текста.

Синтаксический **анализ**, также известный как **анализ** зависимостей, является важной частью обработки **естественного языка**. Он фокусируется на выявлении синтаксических отношений между словами в предложении. Для визуализации этих отношений используется дерево зависимостей, которое является графическим представлением структуры предложения.

Шаг 1. Пайплайн – разбиение текста на отдельные предложения.

Шаг 2. Токенизация, или выделение слов

Шаг 3. Определение частей речи

Шаг 4. Лемматизация

Лемматизация – нахождение основной формы (леммы) каждого слова в предложении.

Шаг 5. Определение стоп-слов

Шаг 6. Парсинг зависимостей

Шаг 6б. Поиск групп существительных

Шаг 7. Распознавание именованных сущностей (Named Entity Recognition, NER)

Большинство NER-моделей распознают следующие типы объектов:

имена людей;

названия компаний;

географические обозначения (и физические, и политические);

продукты;

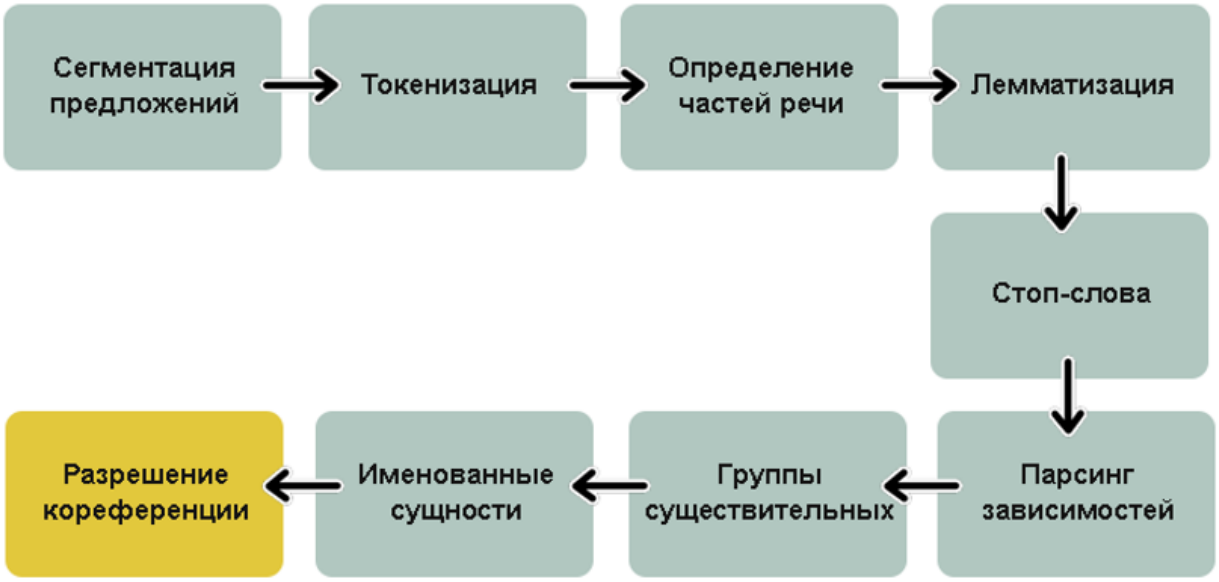
даты и время;

денежные суммы;

события.

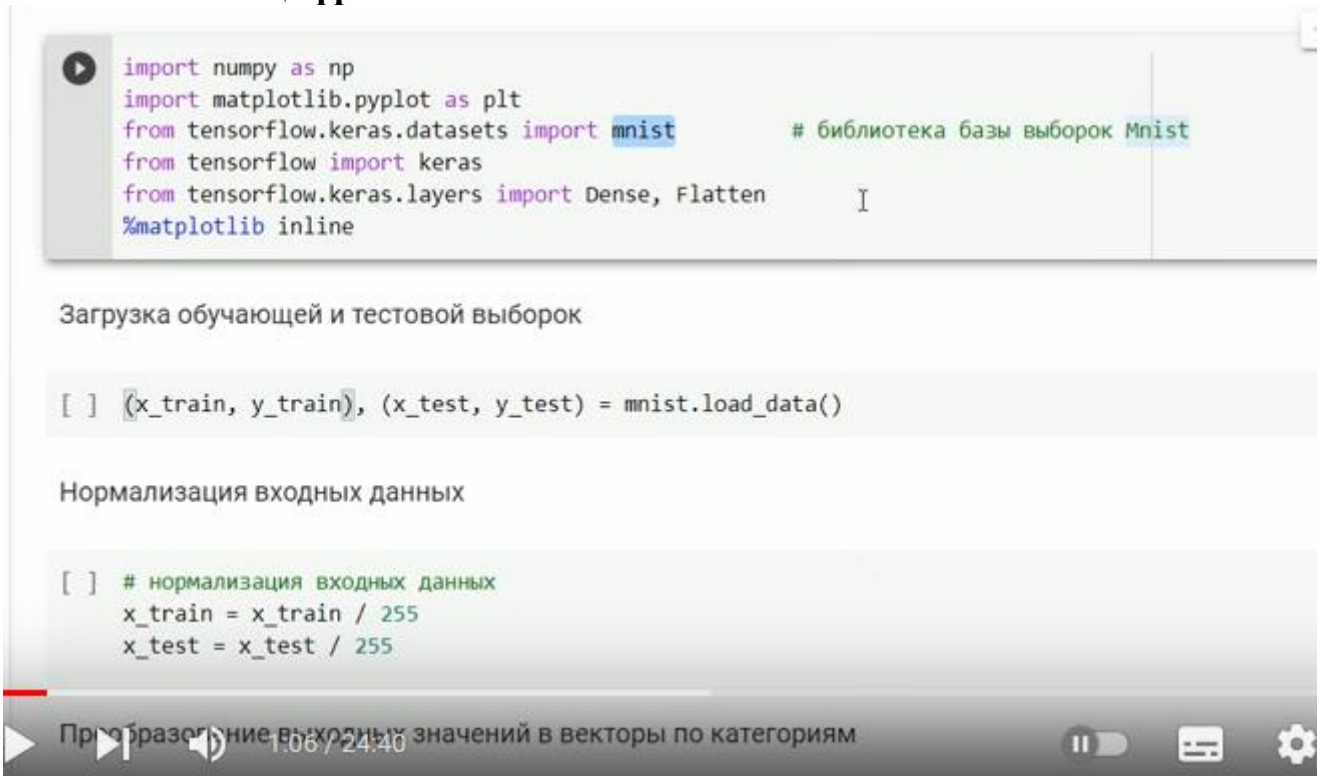
Так как эти модели позволяют легко извлекать из сплошного текста структурированные данные, они очень активно используются в разных областях. Это один из самых простых способов извлечения выгоды из NLP-конвейера.

Шаг 8. Разрешение кореференции



Естественный язык (NLP) Речевые технологии
тексты: распознают, автоматически переводят
речь: распознают, генерируют

Распознавание цифр библиотека МНИСТ



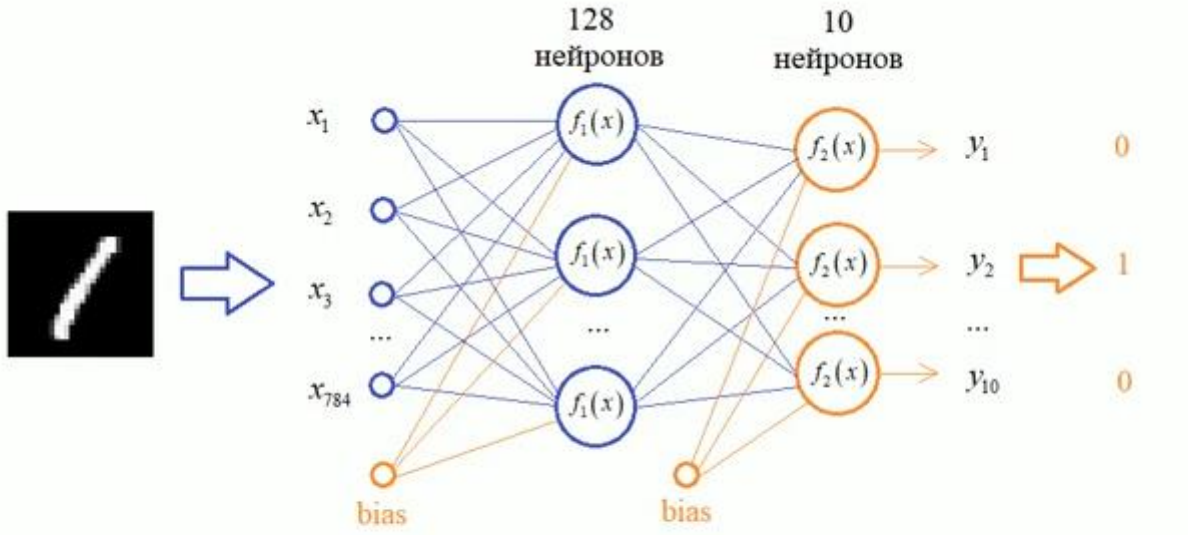
- где взять обучающую выборку
- какую структуру НС выбрать

MNIST – (сокращение от «Modified National Institute of Standards and Technology») – база данных образцов рукописного написания цифр

60 000 изображений в обучающей выборке и 10 000 – в тестовой

(x_train, y_train), (x_test, y_test) = mnist.load_data()

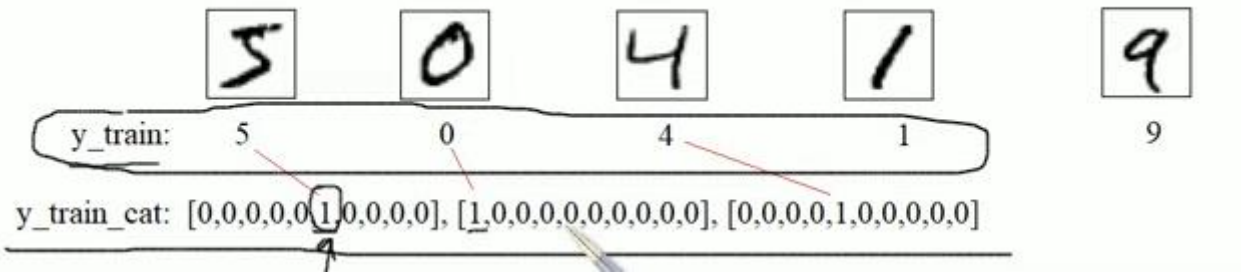
- x_train – изображения цифр обучающей выборки;
- y_train – вектор соответствующих значений цифр (например, если на i-м изображении нарисована 5, то y_train[i] = 5);
- x_test – изображения цифр тестовой выборки;
- y_test – вектор соответствующих значений цифр для тестовой выборки.

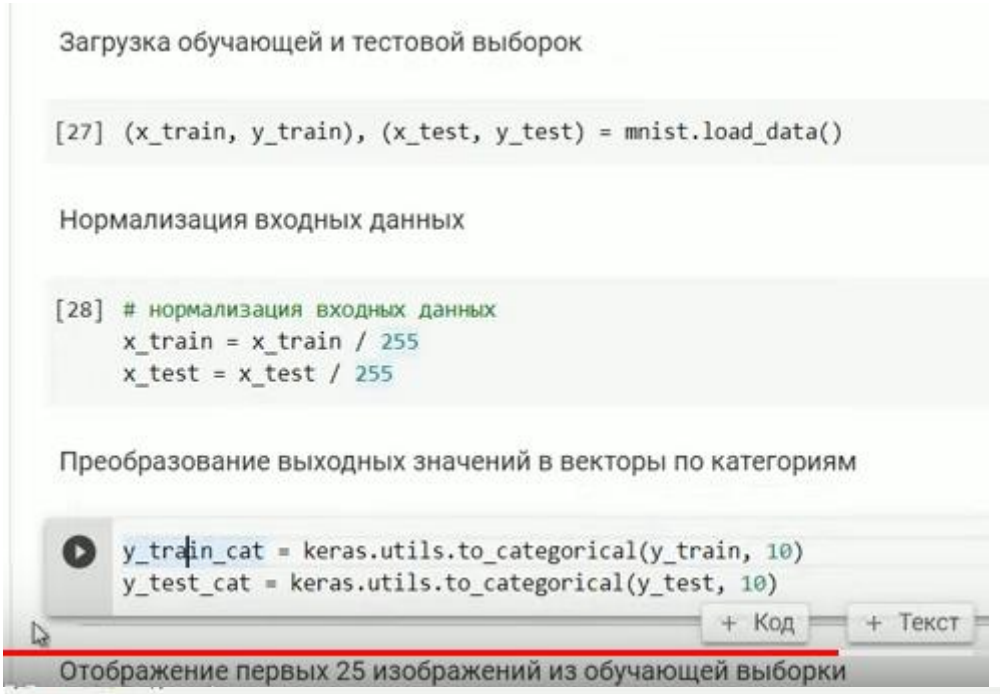


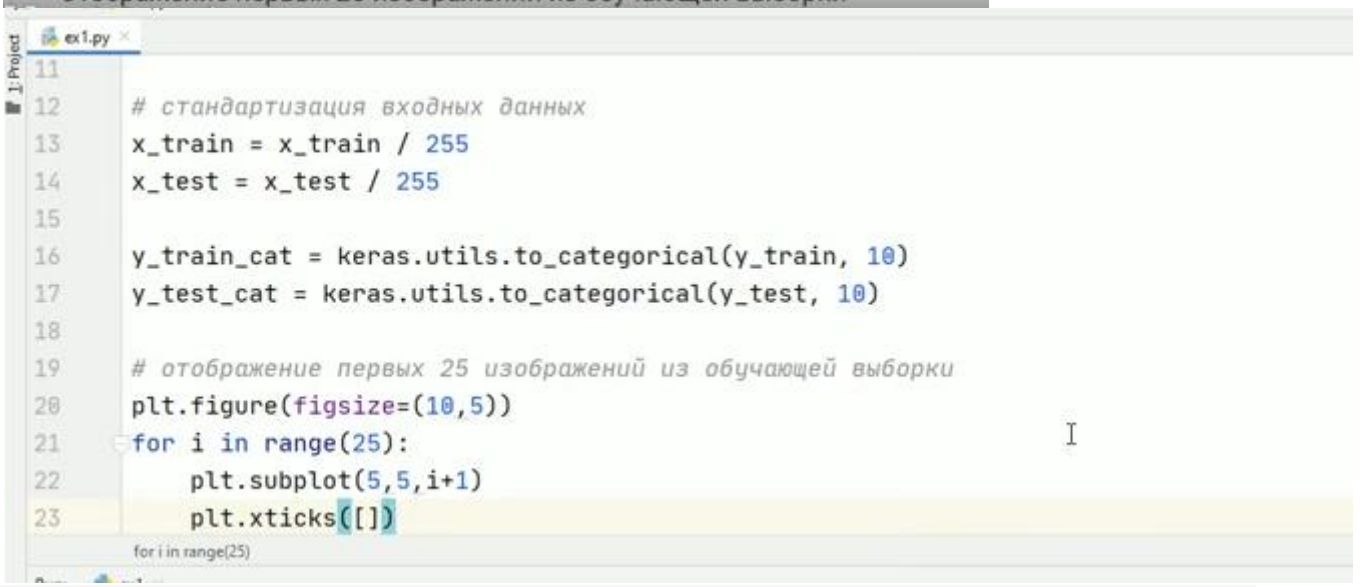
Flatten(input_shape=(28, 28, 1))

Dense(128, activation='relu')

Dense(10, activation='softmax')

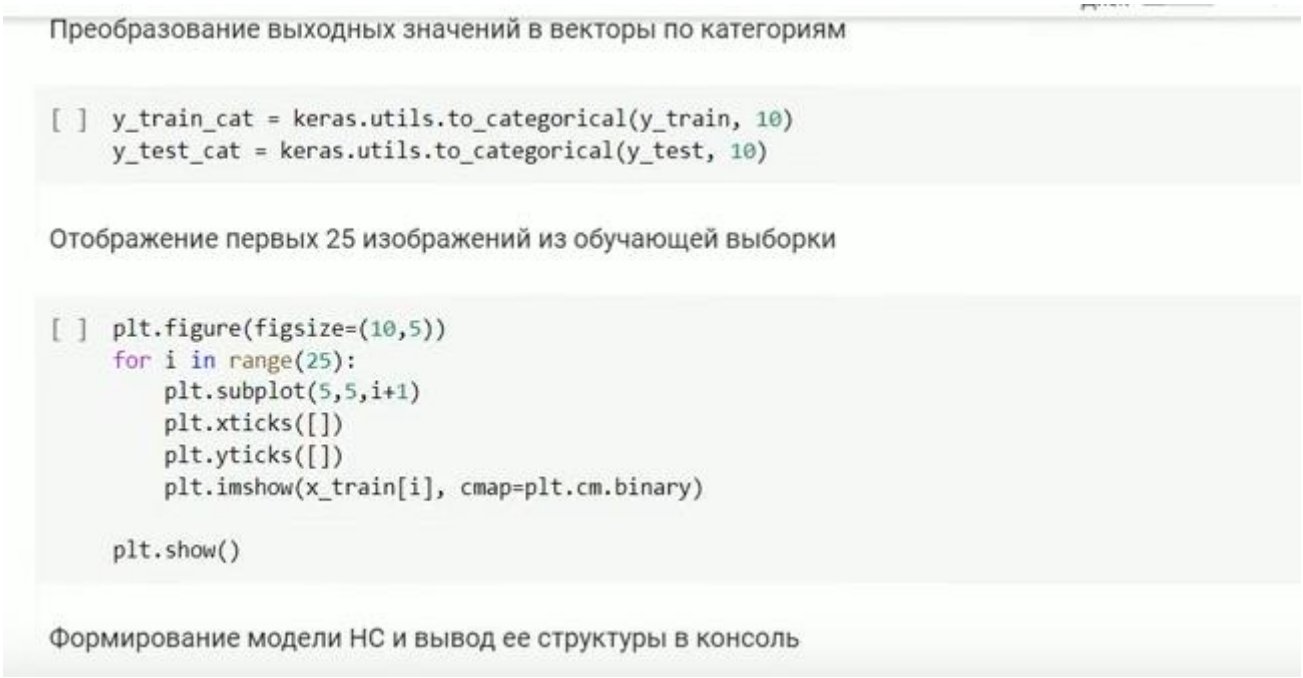




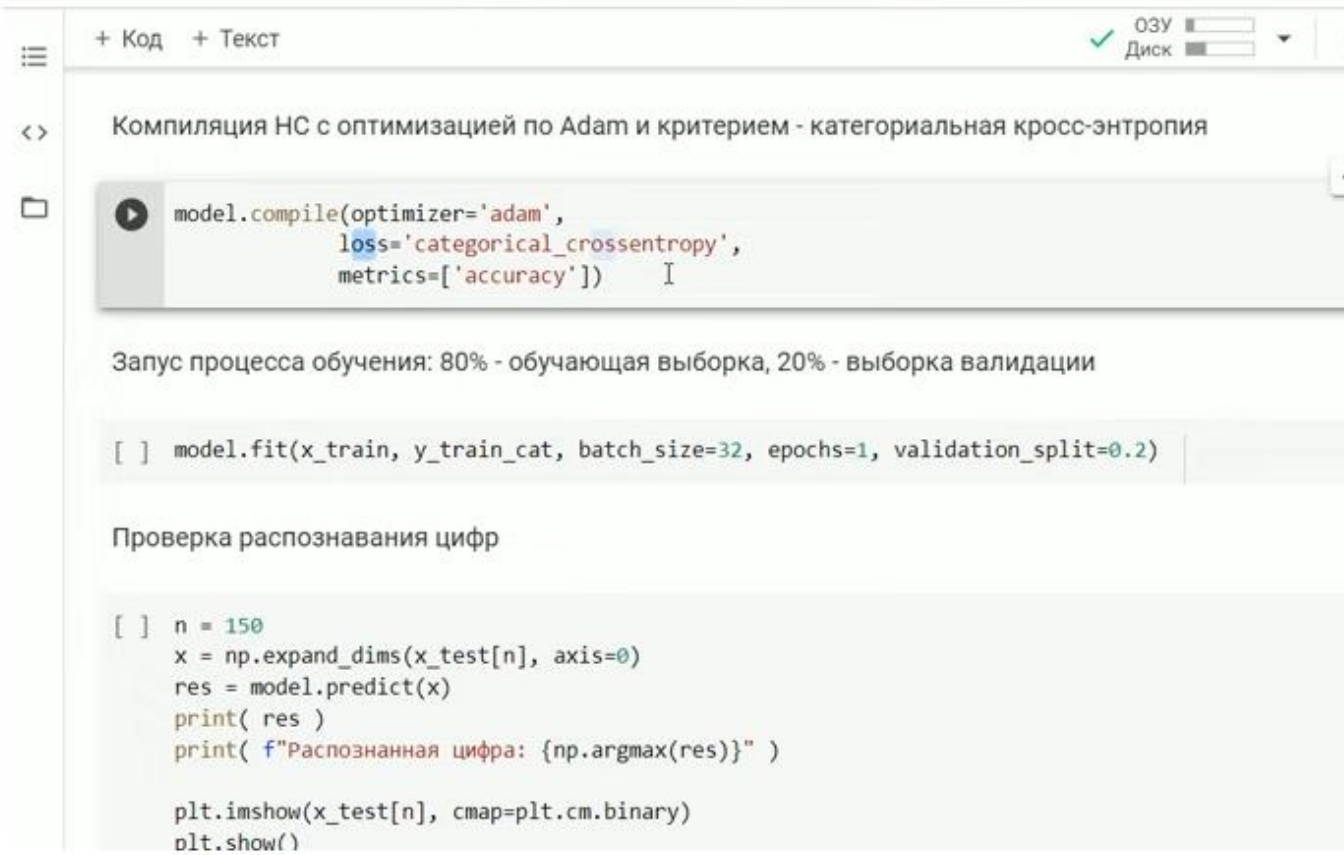



```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist  # библиотекабазы выборок mnist
from tensorflow import keras
from tensorflow.keras.layers import Dense, Flatten
%matplotlib inline
# загрузка обучающей и тестовой выборки
(x_train,y_train),(x_test,y_test)=mnist.load_data()
#нормализация входных данных
x_train=x_train/255
y_train=y_train/255
#стандартизация входных данных
x_train=x_train/255
x_test=x_test/255
y_train_cat=keras.utils.to_categorical(y_train,10)
```

```
y_test_cat=keras.utils.to_categorical(y_test,10)
#отображение первых 25 изображений из обучающей выборки
plt.figure(figsize=(10,5))
for i in range(25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.imshow(x_train[i],cmap=plt.cm.binary)
plt.show()
model=keras.Sequential  ([
Flatten(input_shape(28,28,1)),
Dense(124,activation='relu'),
Dense(10,actiwation='softmax')
])
Print(model.sumary())
```




```
# преобразование выходных значений в векторы по категориям
Y_train_cat=keras.utils.to_categorical(y_train,10)
Y_test_cat=keras.utils.to_categorical(y_test,10)
jnj
# Формирование модели нейронной сети и вывод ее структуры в консоль
model =keras.Sequential([
Flatten(input_shape(28,28,1)),
Dense(128,activation=’relu’),
Dense(10, activation=’softmax’)
])
Print(model.summary())# вывод структуры нейронной сети в консоль
```



model.compile(optimizer=’adam’,
loss=’categorical_crossentropy’,
metrics=[‘accuracy’])
model.fit(x_train,y_train,bath_size=32,epohs=1,validation_split=0.2)
n=150
x=np.expand_dims(x_test[n],axis=0)
res=model.predict(x)
print(res)
print(f’распознанная цифра:(np.argmax(res)’)

Тема 4.2. Одномерные сверточные нейронные сети
Тема 4.3. Рекуррентные нейронные сети для задач обработки естественного языка

ЧАСТЬ 2 РЕШЕНИЕ ЗАДАЧ КОМПЬЮТЕРНОГО ЗРЕНИЯ

- где взять обучающую выборку
- какую структуру НС выбрать

MNIST – (сокращение от «Modified National Institute of Standards and Technology») – база данных образцов рукописного написания цифр

60 000 изображений в обучающей выборке и 10 000 – в тестовой

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

- x_train – изображения цифр обучающей выборки;
- y_train – вектор соответствующих значений цифр (например, если на i-м изображении нарисована 5, то y_train[i] = 5);
- x_test – изображения цифр тестовой выборки;
- y_test – вектор соответствующих значений цифр для тестовой выборки.

```python
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow import keras
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

tf.random.set_seed(1)

input = keras.Input(shape=(32, 32, 3))
x = layers.Conv2D(32, 3, activation='relu')(input)
x = layers.MaxPooling2D(2, padding='same')(x)
x = layers.Conv2D(64, 3, activation='relu')(x)
x = layers.MaxPooling2D(2, padding='same')(x)
x = layers.Flatten()(x)
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.5)(x)
output = layers.Dense(10, activation='softmax')(x)

model = keras.Model(inputs=input, outputs=output)
model.summary()
```

CIFAR-10

нейросеть: 96,5%

человек: 94%

```
from tensorflow import keras

from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical


model = keras.Sequential()
model.add(Dense(128, activation='relu', name="layer1"))
model.add(Dense(10, activation='softmax', name="layer2"))

x = tf.random.uniform((1, 20), 0, 1)
y = model(x)

print( model.layers[0].weights )
```

```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow import keras
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

class TfConv2D(tf.Module):
    def __init__(self, kernel=(3, 3), channels=1, strides=(2, 2), padding='SAME', activate="relu"):
        super().__init__()
        self.kernel = kernel
        self.channels = channels
        self.strides = strides
        self.padding = padding
        self.activate = activate
        self.fl_init = False

    def __call__(self, x):
        if not self.fl_init:
            self.w = tf.random.truncated_normal((*self.kernel, x.shape[-1], self.channels), stddev=0.
```

```
from tensorflow.keras.utils import to_categorical


model = keras.Sequential([
#    Input(shape=(28, )),
    Dense(128, activation='relu', input_shape=(784,), name="hidden_1"),
    Dense(10, activation='softmax', name="output")
])


(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train / 255
x_test = x_test / 255

x_train = tf.reshape(tf.cast(x_train, tf.float32), [-1, 28*28])
x_test = tf.reshape(tf.cast(x_test, tf.float32), [-1, 28*28])

y_train = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)
```

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=32, epochs=5)
```

```
model_ex = keras.Model(inputs=model.inputs,
                       outputs=[layer.output for layer in model.layers])
```

```
x = tf.expand_dims(x_test[0], axis=0)
y = model_ex(x)
y2 = model(x)
print(y, y2, sep="\n\n")
```

```
model_ex = keras.Sequential([
    model,
    Dense(10, activation="tanh")
])

model.trainable = False

model_ex.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model_ex.fit(x_train, y_train, batch_size=32, epochs=3)
```

9. Statsmodels

Когда я сначала изучил R, проведение статистических тестов и исследование статистических данных казались мне самым простым в R и я избегал Python для статистического анализа до тех пор, пока я не изучил Statsmodels в Python.

Когда использовать? Statsmodels - это универсальный пакет Python, который обеспечивает простые вычисления для описательной статистики и оценки и формирования статистических моделей.

Что можно делать с помощью Statsmodels?

1. Линейная регрессия;
2. Корреляция;
3. Метод наименьшего квадрата (OLS);
4. Анализ выживания;
5. Обобщенные линейные модели и байесовская модель;
6. Однофакторный и двухфакторный анализ, проверка гипотез (в основном, что может сделать R!).

10. Plotly

Plotly - это типичная графическая библиотека для Python. Пользователи могут импортировать, копировать, вставлять или передавать данные, которые должны быть проанализированы и визуализированы. Plotly предлагает изолированную версию Python (где вы можете запустить Python,

ограниченный в своих возможностях). Теперь осталось понять, что значит ограниченная версия, но я точно знаю, что Plotly облегчает задачу!
Когда использовать? Вы можете использовать Plotly, если хотите создавать и отображать фигуры, обновлять фигуры, наводить курсор на текст для получения подробной информации. Plotly также имеет дополнительную функцию отправки данных на облачные серверы. Это интересно!

Что можно делать с помощью Plotly?

Библиотека графиков Plotly имеет широкий спектр графиков, которые вы можете построить:

1. ***Основные диаграммы:*** линейные, круговые, точечные, пузырьковые, Ганта, санбёрст, древовидные, санкей, графики с областями;
2. ***Статистические стили и стили Seaborn***: ошибки, гистограммы, диаграммы Facet и Trellis, деревообразные графики, графики-скрипки, линии тренда;
3. ***Научные карты:*** контур, троичный сюжет, логарифмический график, поля векторов, ковровый график (Carpet plot), радарчарт, тепловые карты Роза ветров и Полярный сюжет;
4. Финансовые графики;
5. Карты;
6. Субплоты;
7. Трансформации;
8. Взаимодействие Jupyter Widgets.

Plotly это *типичная* библиотека графиков. Подумайте о визуализации и Plotly сделает это!

Итак, мы изучили путеводитель по топ 10 библиотекам Python для науки о данных, а теперь рассмотрим наши 4 бонусные библиотеки!

1. SpaCy

SpaCy - это библиотека с открытым исходным кодом, используемая для продвинутого NLP для Python и Cython (язык программирования Python, обеспечивающий ощущение и производительность в стиле C с кодом Python, а также синтаксис на основе C).

2. Bokeh

Bokeh - это библиотека Python, которую я бы назвал интерактивной визуализацией данных. С такими инструментами, как Tableau, QlikView или PowerBI, зачем нам Bokeh? Во-первых, Bokeh позволяет очень быстро строить сложные статистические графики с помощью простых команд. Он поддерживает вывод HTML, блокнота или сервера. Во-вторых, можно интегрировать визуализацию Bokeh в приложения Flask и Django или визуализации, написанные в других библиотеках, таких как matplotlib, seaborn, ggplot.

3. Gensim

Gensim - это то, что, по моему мнению, сильно отличается от того, что мы встречали до этого. Он автоматически извлекает семантические темы из документов без труда и с высокой эффективностью. Алгоритмы Gensim не

контролируются, это означает, что никакой человеческий ввод не требуется - просто текстовые документы, и затем выполняется извлечение.
Gensim – библиотека обработки естественного языка предназначения для «Тематического моделирования».
С его помощью можно обрабатывать тексты, работать с векторными моделями слов (такими как Word2Vec, FastText и т. д.) и создавать тематические модели текстов.
https://webdevblog.ru/gensim-rukovodstvo-dlya-nachinajushhih/

4. NLTK

NLTK (Natural Language Toolkit) в основном работает с человеческим языком, а не с компьютерным, чтобы применять обработку естественного языка (NLP). Он содержит библиотеки обработки текста, с помощью которых вы можете выполнять токенизацию, парсинг, классификацию, выделение, тегирование и семантическое обоснование данных. На основе функционала этой библиотеки может показаться, что она повторяется, но каждая библиотека в Python была написана для повышения некоторой эффективности.

Диаграмма рассеяния — Scatter Plot

Диаграмма рассеяния — это диаграмма, которая отображает точки на основе двух измерений набора данных.

Pytorch
Import numpy as np
From sklearn.datasets import make_blobs

```
import numpy as np
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# генерируем 1200 точек, распределенных вокруг двух центров: (-2,0.5) и (3,-0.5)
X, y = make_blobs(n_samples=1200, centers=[[-2,0.5],[3,-0.5]], cluster_std=1, random_state=42)

# отрисовываем точки на плоскости, покрасив их в два цвета в зависимости от класса
colors = ("red", "green")
colored_y = np.zeros(y.size, dtype=str)

for i, cl in enumerate([0,1]):
    colored_y[y.ravel() == cl] = str(colors[i])

plt.figure(figsize=(10,7))
plt.scatter(X[:, 0], X[:, 1], c=colored_y)
plt.show()
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=200)
```

```python
import torch.nn as nn

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.fc1 = nn.Linear(2, 2)
        self.act1 = nn.Sigmoid() # можно заменить на nn.Tanh / nn.ReLU / ...

        self.fc2 = nn.Linear(2, 1)
        self.act2 = nn.Sigmoid()
```

```python
def forward(self, x):
    """
    params:
        x (torch.Tensor): данные, поданные на вход сети
    """

    x = self.fc1(x)
    x = self.act1(x)

    x = self.fc2(x)
    x = self.act2(x)
    return x
```

```python
import torch.nn as nn

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.fc1 = nn.Linear(2, 5)
        self.act1 = nn.Sigmoid() # можно заменить на nn.Tanh / nn.Rel

        self.fc2 = nn.Linear(5, 1)
        self.act2 = nn.Sigmoid()


    def forward(self, x):
        """
        params:
            x (torch.Tensor): данные, поданные на вход сети
        """

        x = self.fc1(x)
        x = self.act1(x)

        x = self.fc2(x)
        x = self.act2(x)
```

```python
def forward(self, x):
    """
    params:
        x (torch.Tensor): данные, поданные на вход сети
    """

    x = self.fc1(x)
    x = self.act1(x)

    x = self.fc2(x)
    x = self.act2(x)
    return x
```

https://colab.research.google.com/drive/1kPOJIH14RC644xgBDfmbU5Rq-iP5sKU-?usp=sharing#scrollTo=xnYVjcnJC4TY

Добавьте в проект все необходимые модули:

mnist – модуль для загрузки набора данных рукописных цифр, который вы используете при обучении нейронной сети;

Sequential – модуль для создания последовательной модели нейронной сети;
Dense – линейный (полносвязный) слой. Из таких слоев будет создана ваша нейросеть;
utils – модуль с полезными инструментами для подготовки данных;
plt – модуль рисования графиков.

```
from tensorflow.keras.datasets import mnist      # Библиотека с базой рукописных цифр
from tensorflow.keras.models import Sequential  # Подключение класса создания модели Sequential
from tensorflow.keras.layers import Dense       # Подключение класса Dense - полносвязный слой
from tensorflow.keras import utils              # Утилиты для подготовки данных
import numpy as np                              # Работа с массивами
import matplotlib.pyplot as plt                 # Отрисовка изображений

# Отрисовка изображений в ячейках ноутбука
%matplotlib inline
```

набор данных для задачи, которую вы собираетесь решать. У вас это набор картинок, на которых изображены рукописные цифры от **0** до **9**. Следующей строкой кода вы скачаете эти данные:

- **x_train_org**, **y_train_org** – изображения для обучения нейронной сети;
- **x_test_org**, **y_test_org** – изображения для тестирования нейронной сети.

Метрики
новые метрики, позволяющие разграничить некоторое множество букв в отдельный класс.
Метрики
Метрика Хэмминга – метрика которая показывает, как сильно объекты не похожи между собой.

метрики (Отражения горизонтального и вертикального, преобладания веса горизонтального и вертикального).
Данную метрику часто используют при кодировании информации и передаче данных. Например, после сеанса передачи на выходе имеется следующая последовательность бит (1001001), также нам известно, что должна прийти другая последовательность бит (1000101). Мы вычисляем метрику путем сравнения частей последовательности с соответствующими местами из другой последовательности. Таким образом метрика Хэмминга в нашем случае равна 2. Так как объекты отличаются в двух позициях. 2- это степень непохожести, чем больше, тем хуже в нашем случае.
Следовательно, чтобы определить какая буква изображена нужно найти ее метрику со всеми готовыми шаблонами. И тот шаблон, чья метрика окажется наиболее близкой к 0 будет ответом.

Задача №1

К Вам обратился Заказчик - «Автосалон XXX». Для решения производственных задач компании необходимо разработать модель определения марки автомобиля по фото.

Вам необходимо подготовить пилотный проект, в рамках которого требуется создать и обучить нейронную сеть, которая будет классифицировать изображение автомобиля по трем категориям:

- Феррари
- Рено
- Мерседес

Используя материалы занятия и датасет, предоставленный Заказчиком (можно скачать с помощью кода ниже), создайте и обучите модель нейронной сети для классификации марок авто. Минимальная точность нейронной сети в рамках разрабатываемого пилотного проекта должна составлять 60% на проверочной выборке.

Степик кодирование изображений
https://stepik.org/lesson/765436/step/2

https://pythonru.com/primery/kak-ispolzovat-modul-datetime-v-python
https://practicum.yandex.ru/blog/svertochnye-neyronnye-seti/

Учебное издание

Коренюгина Людмила Михайловна

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ(PYTHON)

Методические указания по подготовке к практическим занятиям, в том числе в интерактивной форме, и самостоятельному изучению дисциплины для

Составитель: Л.М.Коренюгина
Рецензент:
Редактор:

Оригинал–макет и верстка: Л.М.Коренюгина

Подписано в печать Сдано в производство

Формат 14,85х21,0 Бумага потребительская.
Печать трафаретная. Усл. печ. л. 5,5
Изд. № Заказ №
Тираж экз. завод – экз.

Редакционно–издательский центр МБОУ СОШ 63.
660059, Красноярск, ул. Вавилова, 49 «Б», тел. +7 (391) 201–53–62 e–mail: sch63@mail.ru
МБОУ ДО ЦДО "Аэрокосмическая школа имени Героя Социалистического Труда Гупалова В.К.",
660123, Красноярский край, г. Красноярск, ул. Иркутская, 2 , тел. +7 (391) 262-95-60e–mail: udo_aeroschool@mailkrsk.ru