

DIRTY PYTHON 1.0

STONE 13TH PRESENTS

ЛЕКЦИЯ 9



DIRTY
PYTHON

ЧТО У НАС СЕГОДНЯ?

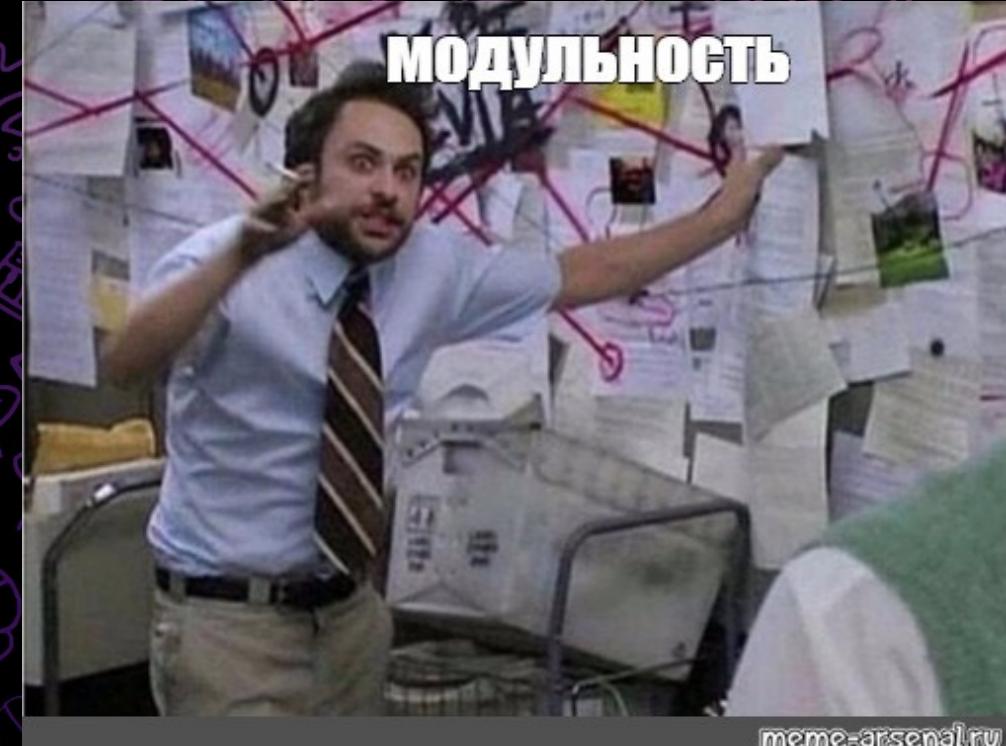
Модульность

Что это вообще такое?

Как организовать?

Какие такие пакеты?

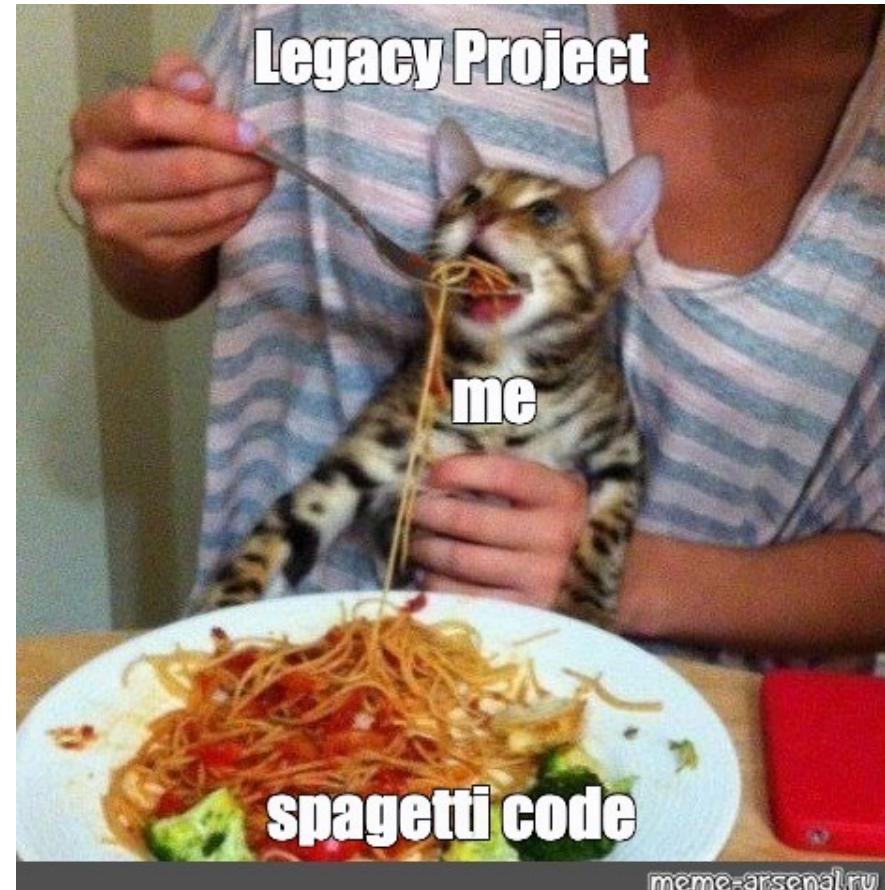
Телефонный справочник



memes-arsenal.ru

СНАЧАЛА ПРО SPAGHETTI CODE

- Когда мы решаем простенькие задачки, то вполне разумно умещать их в одном файле
- Но если у нас проект, который предполагает несколько сотен, тысяч, а то и десятков тысяч строк кода, согласитесь в одном файле работать будет мягко говоря неудобно.
- Кстати, именно такой код и называют лапша-код или spaghetti code. Разобраться в нем сложно даже самому автору, не то что тем, кто будет читать его после



А ТЕПЕРЬ ПРО МОДУЛЬНОСТЬ

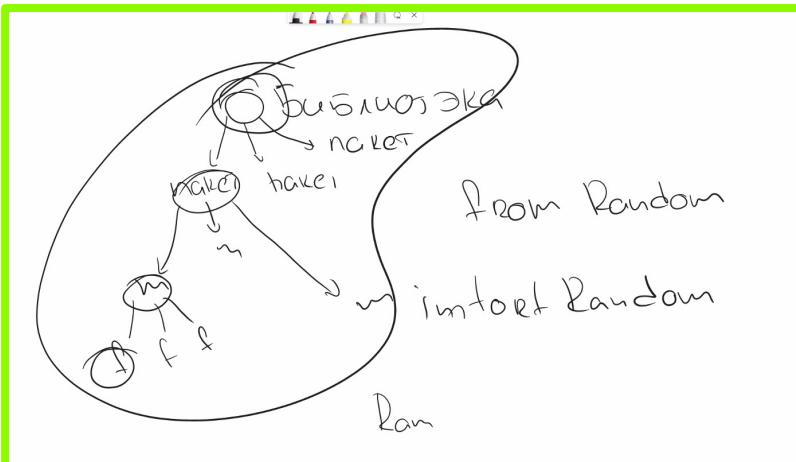
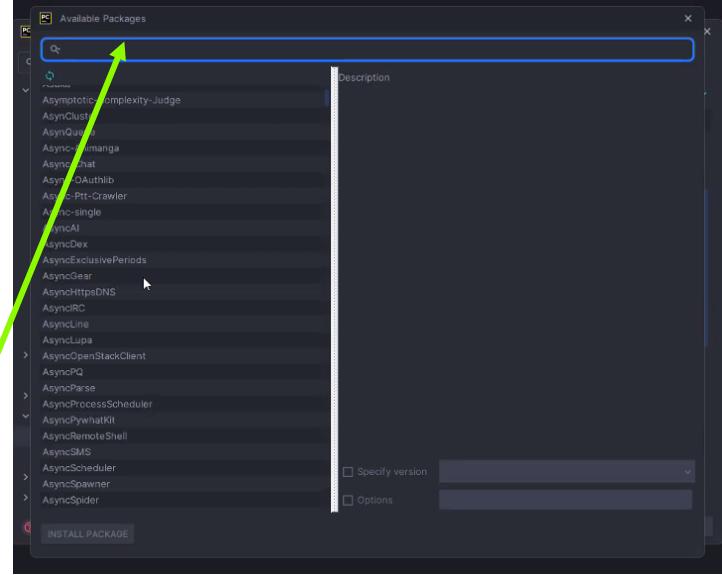


- Модульность в языках программирования — принцип, согласно которому логически связанные между собой подпрограммы, переменные и т. д. группируются в отдельные файлы (модули).
- Группировать модули надо согласно продуманной логики, а если над проектом работает несколько людей и логика у каждого своя? Для этого и есть патерны.
- Поскольку в программировании часто встречаются похожие задачи, то существуют и шаблоны для их решения — паттерны (их около 40 шт)

ПАКЕТЫ И БИБЛИОТЕКИ



- Если файлов становится действительно много их удобно группировать в директории, а в python даже есть специальные package или пакеты, которые позволяют облегчать указание пути к файлам и их содержимому.
- Кстати, многочисленные библиотеки Python, которых вагон и тележечка и постоянно появляются новые, устроены по принципу пакетов.
- Когда мы импортируем библиотеку, мы можем указать не всю ее, а только раздел, или даже подраздел раздела, или даже конкретную нужную функцию из подраздела раздела библиотеки...



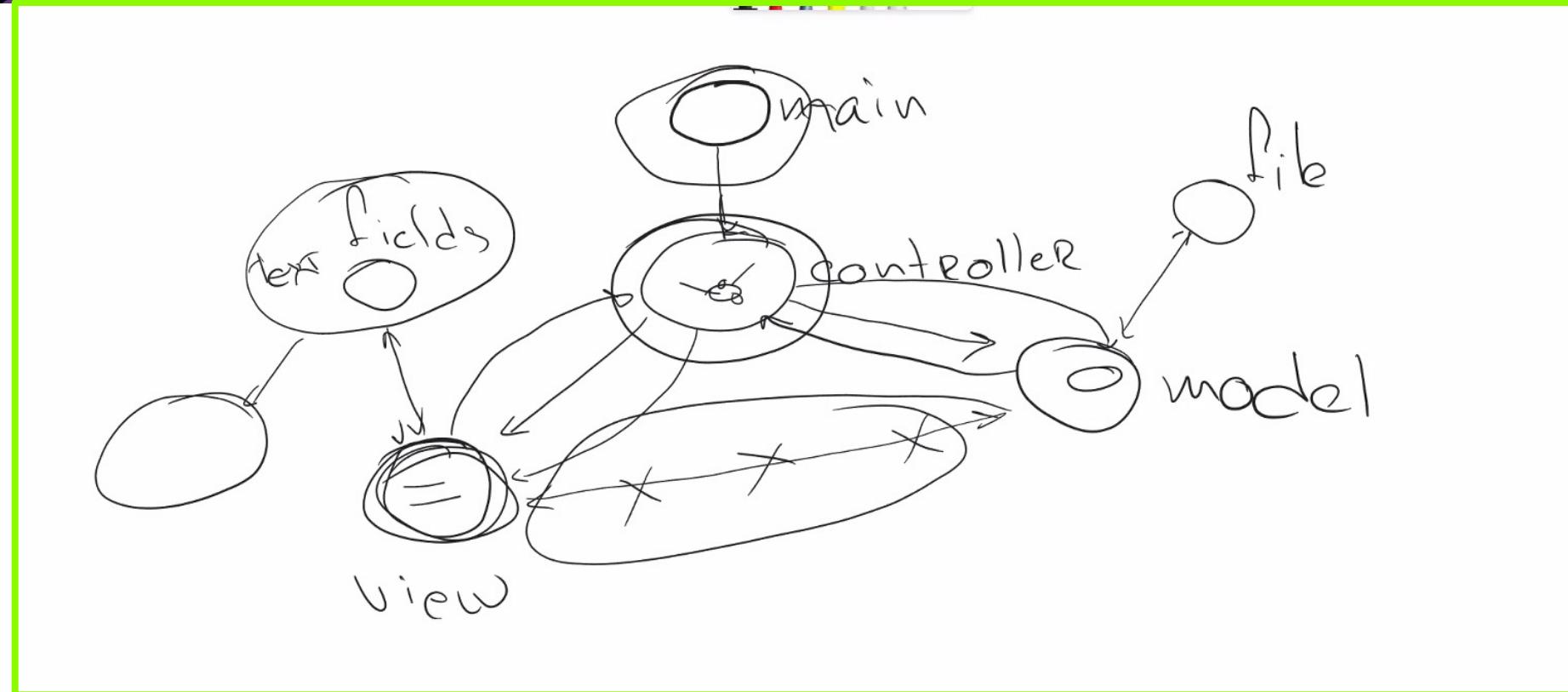


MVC

- Итак, перед началом работы продумывают, какие цели и задачи стоят в проекте, какие функции и как надо реализовать. И как организовать сам проект – то есть его модульность
- Мы уже говорили про паттерны и шаблоны. Сегодня на примере телефонного справочника разберем, что за зверь MVC
- Обязательными частями такой системы является:
- Файл `main` – из которого собственно запускается все программа
- Файл или группа файлов `model` – содержит внутренние функции: обработка информации, обращение к базам данных и т.д.
- Файл `view` – все принты и инпуты (то есть выдача и получение информации будут здесь)
- И файл `controller`, который свяжет между собой `model` и `view`, и через него будет работать весь проект
- Дополнительно могут быть любые блоки – например с базой данных, файлами, логгерами, картинками, текстами и т.д.



СХЕМА MVC

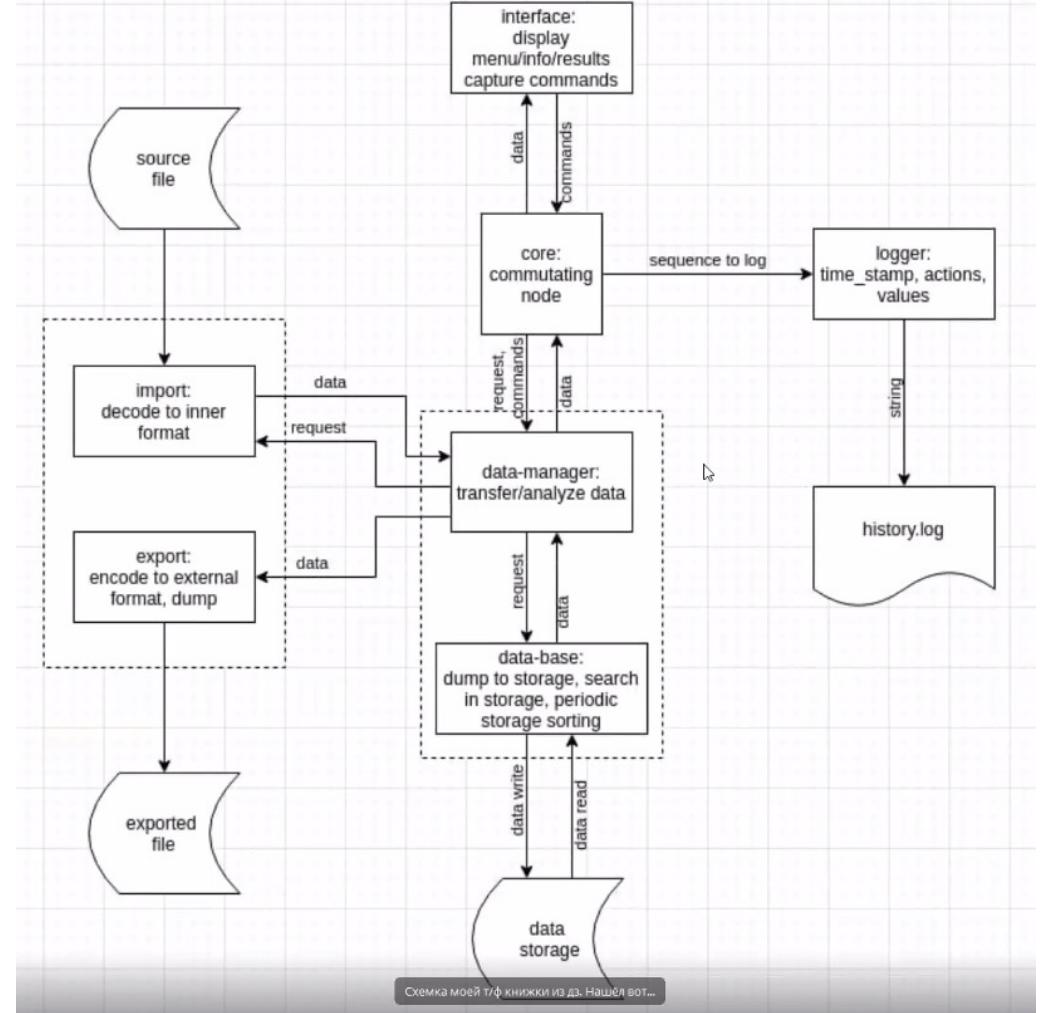


- Как видите view и model работают только через контролер: это позволяет избежать зацикленности, когда мы ссылаемся из файла 1 на файл 2, оттуда на файл 3, а файл 3 снова на файл 1 – в итоге получается путаница, когда не понятно, что откуда брать

ТЕЛЕФОННЫЙ СПРАВОЧНИК



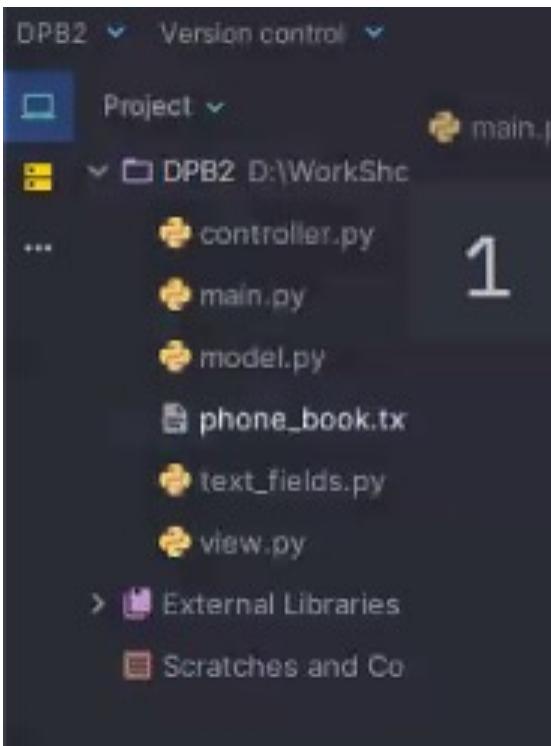
- Андрей нарисовал вот такую схему...
 - Мы же давайте подумаем, что должен уметь делать наш справочник...
 - Открывать файл с записями, показывать его, вносить новые записи, изменять старые, удалять данные, осуществлять поиск, сохранять изменения... Все это и запишем в главное меню
 - Теперь, когда мы знаем, что хотим от справочника осталось только написать все функции)))



НАЧИНАЕМ?



- Начнем с того, что создадим в проекте все модули которые нам понадобятся:
 - ✓ main.py
 - ✓ controller.py
 - ✓ model.py
 - ✓ view.py
 - ✓ phone_book.txt – собственно сама телефонная книга, с которой работаем
 - ✓ text_field.py – файлик в котором удобно хранить строковые переменные (например, для вывода тех или иных сообщений в терминал)



АДАЛЬШЕ СМОТРИМ КОД



- Для начала запишем несколько контактов в саму телефонную книгу
- Помним, что нам потом еще извлекать информацию, поэтому сразу продумайте через какой символ будете сплитовать
- В `text_field` создаем переменные, куда будем записывать весь текст
- В `main` запустим наш контроллер
- В контроллере – запускаем функцию `старт`, которая и будет обеспечивать все взаимодействия между юзером и файлом через `view` и `model` ...

```
Панфилов Кирилл:89094512021:Семинары GB
* Роман Лепихов:8890988098:Молодец
Andrey Беляев:9900:Друг Стоуна
Диана Мороз:00-00-000:Пиздатые конспекты
```

```
menu = '''Главное меню:
1. Открыть файл
2. Сохранить файл
3. Просмотреть все контакты
4. Добавить контакт
5. Найти контакт
6. Изменить контакт
7. Удалить контакт
8. Выход'''
```

```
import controller

if __name__ == '__main__':
    controller.start()
```

СМОТРИМ КОД



В файле view пропишем функции, которые покажут нам главное меню, примут информацию, какой пункт выбрал юзер, используя тексты из text_field

И тут же будут все остальные функции, которые предполагают ввод и вывод информации

```
main.py controller.py view.py x model.py text_fields.py phone_book.txt
1 import text_fields
2
3 def main_menu() -> int:
4     print(text_fields.menu)
5
```

```
4 def input_choice():
5     while True:
6         number = input(text_fields.input_choice)
7         if number.isdigit() and 0 < int(number) < 9:
8             return int(number)
9         else:
10            print(text_fields.wrong_choice)
11
```

СМОТРИМ КОД



В файле view пропишем функции, которые покажут нам главное меню, примут информацию, какой пункт выбрал юзер, используя тексты из text_field

И тут же будут все остальные функции, которые предполагают ввод и вывод информации

```
main.py controller.py view.py x model.py text_fields.py phone_book.txt
1 import text_fields
2
3 def main_menu() -> int:
4     print(text_fields.menu)
5
```

```
4 def input_choice():
5     while True:
6         number = input(text_fields.input_choice)
7         if number.isdigit() and 0 < int(number) < 9:
8             return int(number)
9         else:
10            print(text_fields.wrong_choice)
11
```

СМОТРИМ КОД



В файле controller прописываем функцию, которая содержит бесконечный цикл: запускает функцию main_menu, а далее обрабатывает выбор пользователя

Здесь отлично подойдет match case

Осталось только заполнить все случаи))

Например, организуем в model открытие нашего файла

А во view организуем показ контактов

```
def start():
    while True:
        choice = view.main_menu()
        match choice:
            case 1:
                model.open_file()
            case 2:
                pass
            case 3:
                pb = model.phone_book
                view.show_contact(pb, text_fields.no_phone_book)
            case 4:
                pass
            case 5:
```

```
def show_contact(book: list[dict[str, str]], message: str):
    if book:
        for i, contact in enumerate(book, 1):
            print(f'{i:<3} | {contact["name"]:<20} | {contact["phone"]:<20} '
                  f'| {contact["comment"]:<20}')
    else:
        print(message)
```

```
5 def open_file():
6     with open(PATH, 'r', encoding='UTF-8') as file:
7         data = file.readlines()
8         for contact in data:
9             contact = contact.strip().split(':')
10            contact = {'name': contact[0], 'phone': contact[1], 'comment': contact[2]}
11            phone_book.append(contact)
12
```

И ЧТО ДАЛЬШЕ?

- ЗАПОЛНИТЬ ВЕСЬ ФУНКЦИОНАЛ
- ПОСТАРАЙТЕСЬ, ЧТОБЫ ВАШИ ФУНКЦИИ БЫЛИ МАКСИМАЛЬНО УНИВЕРСАЛЬНЫ, НО ПРИ ЭТОМ ДЕЛАЛИ ЧТО-ТО ОДНО (АТОМАРНЫ)
- ДА, НА МАЛЕНЬКИХ ПРОЕКТАХ МОДУЛЬНОСТЬ МОЖЕТ ПОКАЗАТЬСЯ СИЗИФОВЫМ ТРУДОМ, НО ПОВЕРЬТЕ, ВЫ ОЦЕНИТЕ, КОГДА ЗАХОТИТЕ НЕМНОГО ИЗМЕНИТЬ ПРОЕКТ И ВАМ НЕ ПРИДЕТСЯ ПЕРЕЛОПАЧИВАТЬ ВЕСЬ МНОГОСТРОЧНЫЙ КОД, ЧТОБЫ НАЙТИ ВЗАИМОСВЯЗИ



D1RTY
РУТНОН