

Московский Автомобильно-Дорожный Государственный Университет

(МАДИ)

Заочный Факультет

Преподаватель: Макаренко Любовь Федоровна

Группа 2ЗбАСУс1

Выполнил: Трифонов Евгений Александрович

Курсовая работа по ООП

Реализовать работу сети магазинов

(человек, покупка, продавец, магазин).

Содержание

[Введение 3](#_Toc61897402)

[Python 4](#_Toc61897403)

[Используемые библиотеки 6](#_Toc61897404)

[Библиотека numpy 6](#_Toc61897405)

[Библиотека pandas 7](#_Toc61897406)

[Библиотека re 7](#_Toc61897407)

[Управление аутентификацией пользователя 8](#_Toc61897408)

[Обработка входа пользователя 8](#_Toc61897409)

[Обработка выхода пользователя из системы 11](#_Toc61897410)

[Создание модели 13](#_Toc61897411)

[Отправляем сообщение в чат 14](#_Toc61897412)

[Отображение сообщений чата 15](#_Toc61897413)

[Вывод 18](#_Toc61897414)

[Список литературы 19](#_Toc61897415)

# Введение

Необходимо написать программу в стиле ООП, удовлетворяющую следующим условиям: в программе должны быть классы и объекты, принадлежащие разным классам; один объект с помощью метода своего класса должен так или иначе изменять данные другого объекта. Желательно в программе продемонстрировать такие свойства ООП как наследование и полиморфизм.

# Python



Python (в русском языке распространено название пито́н или, более правильно, па́йтон) — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой набор полезных функций.

Python поддерживает структурное, обобщенное, объектно-ориентированное, функциональное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализация интерпретатора для JVM с возможностью компиляции, CLR, LLVM, другие независимые реализации. Проект PyPy использует JIT-компиляцию, которая значительно увеличивает скорость выполнения Python-программ.

Python — активно развивающийся язык программирования, новые версии с добавлением/изменением языковых свойств выходят примерно раз в два с половиной года. Язык не подвергался официальной стандартизации, роль стандарта де-факто выполняет CPython, разрабатываемый под контролем автора языка. В настоящий момент Python занимает второе место в рейтинге TIOBE с показателем 12,12 %. Аналитики отмечают, что это самый высокий балл Python за все время его присутствия в рейтинге.

# Используемые библиотеки

В ходе написания данного проекта, были использованы библиотеки pandas, numpy, re.

## Библиотека numpy



NumPy — библиотека с открытым исходным кодом для языка программирования Python. Возможности:

* поддержка многомерных массивов (включая матрицы);
* поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами

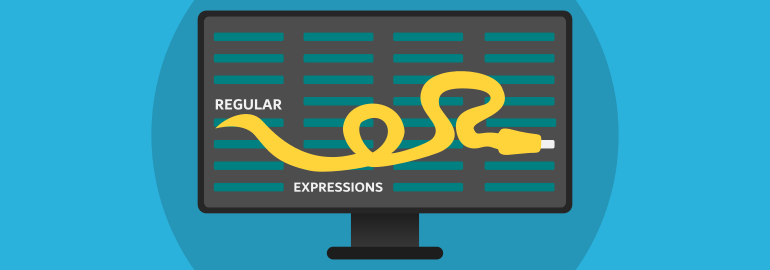
Математические алгоритмы, реализованные на интерпретируемых языках (например, Python), часто работают гораздо медленнее тех же алгоритмов, реализованных на компилируемых языках (например, Фортран, Си, Java). Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами. В результате любой алгоритм, который может быть выражен в виде последовательности операций над массивами (матрицами) и реализованный с использованием NumPy, работает так же быстро, как эквивалентный код, выполняемый в MATLAB.

## Библиотека pandas



Pandas — программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временны́ми рядами. Название библиотеки происходит от эконометрического термина «панельные данные», используемого для описания многомерных структурированных наборов информации. pandas распространяется под новой лицензией BSD.

## Библиотека re



Re – это библиотека регулярных вырождений. Говоря простым языком, регулярное выражение — это последовательность символов, используемая для поиска и замены текста в строке или файле. Как уже было упомянуто, их поддерживает множество языков общего назначения: Python, Perl, R. Так что изучение регулярных выражений рано или поздно пригодится.

# Ход работы

Для начала создадим контекстное меню для пользователя и установим функционал программы :

* print("по команде “se” осуществим поиск в базе")
* print("по команде “ad” добавим магазин в базу")
* print("по команде “de” удалим магазин из базы")
* print("по команде “all” посмотрим всю Базу данных ")
* print("по команде “by” Оформим заказ")
* print("по команде “end” программа закроется ")

Следуя данным пунктам, разобьем нашу работу на несколько этапов:

1. Поиск в БД
2. Добавление в БД
3. Удаление в БД
4. Просмотр БД
5. Оформление заказа

Опираясь на основные принципы ООП, будем использовать классы. Для этого создадим их (рисунок 1.)

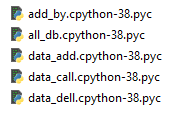


Рисунок 1 – созданные классы.

Классы – это объекты, которые позволяют группировать структуры данных и процедуры в одном месте. Например, представьте, что вы пишете код для организации инвентаря в магазине одежды. Вы можете создать класс, который берет каждый предмет одежды в магазине и хранит ключевые параметры, такие как тип одежды, ее цвет и размер.

# Класс data\_call

import pandas as pd

import numpy as np

data = pd.read\_csv('Contacts\_sample.csv', delimiter=';') # записываем из файла в DataFrame

class data\_call:

'ищем магазин по введеному имени'

# Constructor

def \_\_init\_\_ (self,link\_app):

# Класс data\_call имеет 1 атрибут 'link\_app'

self.link\_app = link\_app

# Метод

def search\_data(self):

sort\_data = data.query("LInk == @self.link\_app")

print (sort\_data)

print("")

В данном классе мы принимаем значение от пользователя и ищем его в базе с помощью библиотеки pandas. В данной библиотеки есть метод query, который работает с SQL.

# Класс data\_dell

import pandas as pd

import numpy as np

data = pd.read\_csv('Contacts\_sample.csv', delimiter=';') # записываем из файла в DataFrame

from data\_call import data\_call

# Клас data\_dell расширен из класса data\_call.

class data\_dell(data\_call):

'удаляем магазин по введеному имени'

def \_\_init\_\_ (self,link\_app,id\_sites):

# Вызывается constructor родительского класса (data\_call)

# чтобы прикрепить значение к атрибуту 'link\_app' родительского класса

super().\_\_init\_\_(link\_app)

self.id\_sites = id\_sites

# Переопределить (override) метод с одинаковым названием родительского класса.

def deleted\_data (self):

sort\_data = data.query("ID == @self.id\_sites and LInk == @self.link\_app")

print("будет удалена строка: ")

print (sort\_data)

data = data.drop(np.where(data['ID'] == self.id\_sites )[0])

print("data")

Python позволяет вам создать расширенный класс из одного или многих других классов. Этот класс называется производный класс (derived class) или просто подкласс.

Подкласс унаследует атрибуты, методы, и другие члены из родительского класса. Он так же может переопределять (override) методы родительского класса. Если подкласс не определяет свой конструктор, он унаследует конструктор родительского класса по умолчанию.

В отличие от Java, CSharp и некоторых других языков, Python допускает множественное наследование. Класс может быть расширен из одного или нескольких родительских классов.

В Python, конструктор (constructor) используется для создания объекта и прикрепляет значение атрибутам (attribute).

Конструктор подклассов всегда вызывается конструктором родительского класса, чтобы инициализировать значение для атрибутов родительского класса, затем он прикрепляет значение этим атрибутам.

data\_dell является подклассом, унаследованным от класса data\_call, он тоже имеет атрибуты (attribute) этого класса.

# Класс – data\_add

Python позволяет множественное наследование, что означает, что вы можете создать расширенный класс из двух или более других классов. Родительские классы могут иметь одинаковые атрибуты (attribute) или методы .... Подкласс будет приоритетно наследовать атрибуты, методы, ... первого класса в списке наследования.

import pandas as pd

import numpy as np

data = pd.read\_csv('Contacts\_sample.csv', delimiter=';') # записываем из файла в DataFrame

class data\_add:

'добавляем магазин'

# Constructor

def \_\_init\_\_ (self,link\_app,Contact\_app, tupe\_app):

self.link\_app = link\_app # передаем название

self.Contact\_app = Contact\_app # передаем контакты

self.tupe\_app = tupe\_app # тип контакты ( мыло или телефон )

count\_row = data.shape[0] + 1 # весь список и добавляем в конец

count\_id = data['ID'].max()+ 1 # добавляем последний ID

count\_sitesid = data['SiteID'].max()+ 1 # последнее id сайта

data.loc[count\_row] = [count\_id, count\_sitesid, self.link\_app, self.Contact\_app, self.tupe\_app ]

sort\_data = data.query("ID == @count\_id") #

print ("")

print ("добавлена запись")

print (sort\_data)

print ("")

# Метод

#полиморфизм

def greeting(self):

print (data)

Данный класс так – же построен на методе наследования. Его вызов в программе происходит по средством вызова функции класса и передачи в нее значения от пользователя.

link\_app = input ("введите ссылку на магазин для удаления: «)

print ("")

data\_c = data\_call(link\_app)

data\_c.search\_data()

id\_sites = input ("Выберите ID для удаления: «)

data\_d = data\_dell(link\_app,id\_sites)

data\_d.deleted\_data()

# Класс – all\_db

import pandas as pd

import numpy as np

data = pd.read\_csv('Contacts\_sample.csv', delimiter=';') # записываем из файла в DataFrame

#полиморфизм

class get\_bd():

# Метод

#полиморфизм

def greeting(self):

print ("База данных имеет вид")

print (data)

Полиморфизм с функцией

Здесь я создаю два класс, all\_db и пользуюсь готовым классом data\_add. У обоих классов есть метод greeting (). Оба выводят базу данных. Создадим два соответствующих объекта из двух классов выше и вызовим действия этих двух объектов в одной функции (функция intro).

В пункте просмотра всей базы:

#полиморфизм

def intro(BD):

BD.greeting()

flora = get\_bd()

intro(flora)

В пункте добавление:

#полиморфизм

def intro(BD):

BD.greeting()

intro(data\_a)

# Класс – add\_by

import pandas as pd

import numpy as np

from data\_call import data\_call

dict = {'ФИО':["Трифонов Е.А.", "Горский И.А.", "Трифонов Е.А.", "Гросул А.В."],

'Товар': ["Телефон Iphone 11 pro max", "Телевизор Samsung PlasaTV 55\* ", "Наушники AirPods pro max", "Кофемашинка Bosh 1732p"],

'Ссылка на магазин': ["www.mvideo.ru", "www.enter.ru", "www.insales.ru", "www.ozon.ru"],

'Контакты':["89777737304", "gosky.ilya@gmail.com", "trifon799@gmail.com", "89151597812]"]}

dff = pd.DataFrame(dict)

data = pd.read\_csv('Contacts\_sample.csv', delimiter=';') # записываем из файла в DataFrame

# Клас add\_by расширен из класса data\_call

class add\_by(data\_call):

'удаляем магазин по введеному имени'

def \_\_init\_\_ (self,link\_app,id\_sites,FIO,tovar):

# Вызывается constructor родительского класса (data\_call)

# чтобы прикрепить значение к атрибуту 'link\_app' родительского класса

super().\_\_init\_\_(link\_app)

self.id\_sites = id\_sites

self.FIO = FIO

self.tovar = tovar

# Переопределить (override) метод с одинаковым названием родительского класса.

def by\_pisition (self):

sort\_data = data.query("ID == @self.id\_sites and LInk == @self.link\_app")

print("вы покупаете у данного магазина")

print (sort\_data)

print("")

segment = int (self.id\_sites)

contact\_bd = data.loc[data['ID'] == segment, 'Contact'].item()

LInk\_bd = data.loc[data['ID'] == segment, 'LInk'].item()

count\_row = dff.shape[0] + 1 # весь список и добавляем в конец

print("Будет добавлен пользователь",self.FIO,",который хочет купить",self.tovar,",у магазина ",LInk\_bd,". Контакты магазина ",contact\_bd)

print("")

dff.loc[count\_row] = [self.FIO, self.tovar, LInk\_bd, contact\_bd]

print (dff)

print("")

В данном классе добавляем записи в базу данных используя уже знакомые нам алгоритмы действия.

# Вывод

В ходе написания данного проекта, было разработано программное обеспечение (далее ПО), в стиле ООП, удовлетворяющую следующим условиям: в программе должны быть классы и объекты, принадлежащие разным классам; один объект с помощью метода своего класса должен так или иначе изменять данные другого объекта. В программе продемонстрированы такие свойства ООП как наследование и полиморфизм.

Благодаря простоте и гибкости языка Python, его можно рекомендовать пользователям (математикам, физикам, экономистам и т.д.) не являющимся программистами, но использующими вычислительную технику и программирование в своей работе.

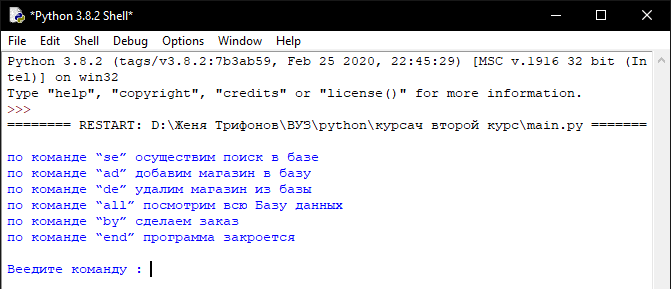


Рисунок 2 – главное меню



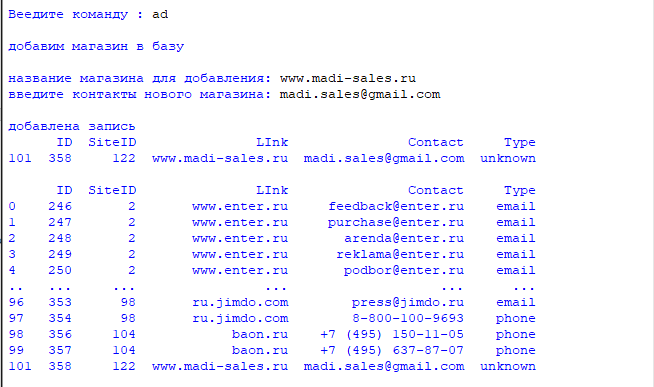
Рисунок 3 – поиск в базе 

Рисунок 4 – добавление записей

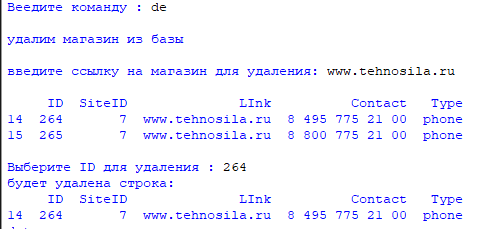


Рисунок 5 – удаление записей



Рисунок 6 – просмотр всей базы.

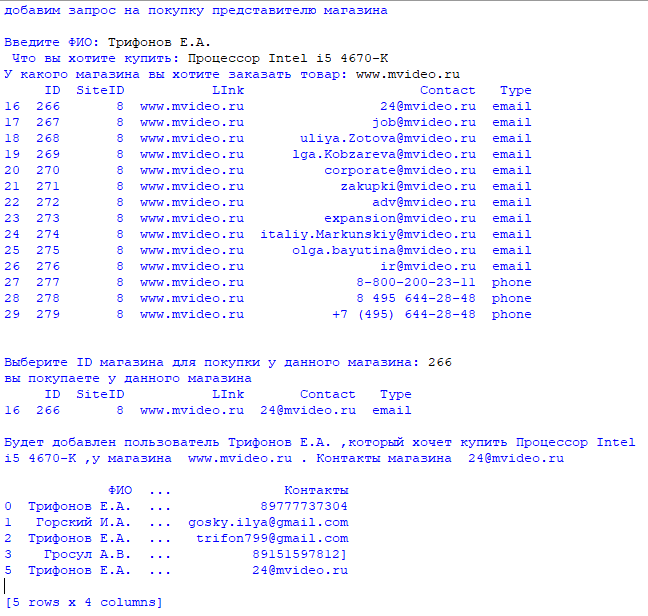


Рисунок 7 – покупка у представителя магазина

Программы на Python разрабатываются в среднем в полтора-два (а порой и в два-три) раза быстрее, нежели на компилируемых языках (С, С++, Pascal). Поэтому, язык может представлять не малый интерес и для профессиональных программистов, разрабатывающих приложения, не критичные к скорости выполнения, а также программы, использующие сложные структуры данных. В частности, Python хорошо зарекомендовал себя при разработке программ работы с графами, генерации деревьев.

Все поставленные цели выпилены, программа написана и функционирует.

# Список литературы

1. Марк Лутц - Изучаем Python (4-е издание) / Марк Лутц, – Москва, 2019 – 592с;
2. Марк Саммерфилд - Программирование на Python 3. Подробное руководство / Марк Саммерфилд– Москва, 2020 – 670с;
3. Научная электронная библиотека Python [Электронный ресурс].– Режим доступа: <https://www.python.org/>, свободный – (06.06.2020)