Министерство образования Московской области

Государственное образовательное учреждение высшего образования Московской области

«Государственный гуманитарно-технологический университет»

**Ликино-Дулевский политехнический колледж – филиал ГГТУ**

**Отчёт по учебной практике**

МДК 01.01 «Разработка программных модулей»

Вариант 1 «Информация о Сотрудниках»

**Выполнил(а):**

Абуталыблы Ильяс\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студентка группы ИСП.20.А\_\_\_\_\_\_\_\_\_\_\_\_

09.02.07 Информационные системы и программирование)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

очной формы обучения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель:**

Гжегожевский С.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2022 г.

Оглавление

[**1.** **Разработка программных модулей в среде Python** 3](#_Toc121350702)

[**1.1.** **Подготовка и настройка среды. Начало работы** 3](#_Toc121350703)

[**1.1.1.** **Установка интерпретатора Python** 3](#_Toc121350704)

[**1.1.2.** **Установки и настройка IDE** 5](#_Toc121350705)

[**1.1.3.** **Особенности задания имен переменных и правила организации кода в Python** …………………………………………………………………………………….12](#_Toc121350706)

[**1.2.** **Библиотека для анализа данных Pandas** 14](#_Toc121350707)

[**1.2.1.** **Описание библиотеки Pandas** 14](#_Toc121350708)

[**1.2.2.** **Практическая часть** 14](#_Toc121350709)

[**1.3.** **Создание пользовательской модели классов** 17](#_Toc121350710)

[**1.3.1.** **Основные работы с пользовательскими классами в Python** 17](#_Toc121350711)

[**1.3.2.** **Практическая часть** 18](#_Toc121350712)

[**1.4.** **Построение интерфейса приложения** 20](#_Toc121350713)

[**1.4.1.** **Библиотека PyQt5** 20](#_Toc121350714)

[**1.4.2.** **Практическая часть** 20](#_Toc121350715)

[**1.5.** **Написание локального веб-сервера при помощи модуля flask** 23](#_Toc121350716)

[**1.5.1.** **Введение во flask** 23](#_Toc121350717)

[**1.5.2.** **Написание веб-сервера для своего варианта** 24](#_Toc121350718)

[**1.6.** **Юнит тестирование Pytest** 27](#_Toc121350719)

[**1.6.1.** **Юнит-тестирование в Python** 27](#_Toc121350720)

[**1.6.2.** **Написание юнит-тестов для своей предметной области** 27](#_Toc121350721)

[**Вывод по разделу** 27](#_Toc121350722)

[**Приложения** 29](#_Toc121350723)

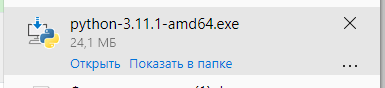
1. **Разработка программных модулей в среде Python**
   1. **Подготовка и настройка среды. Начало работы**
      1. **Установка интерпретатора Python**

**Установка:**

1. Заходим на официальный сайт и скачиваем предлагаемую версию.



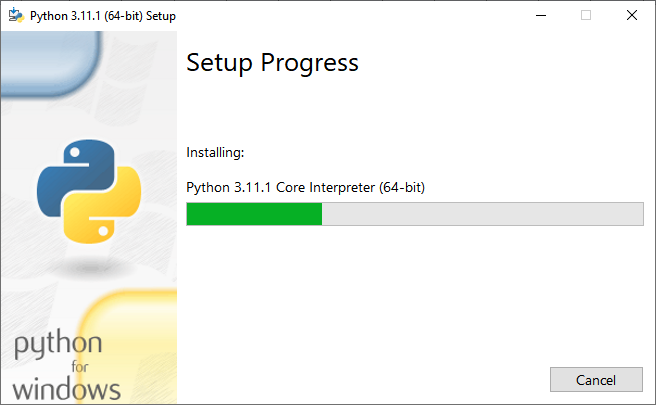
1. Запускаем установленный файл.



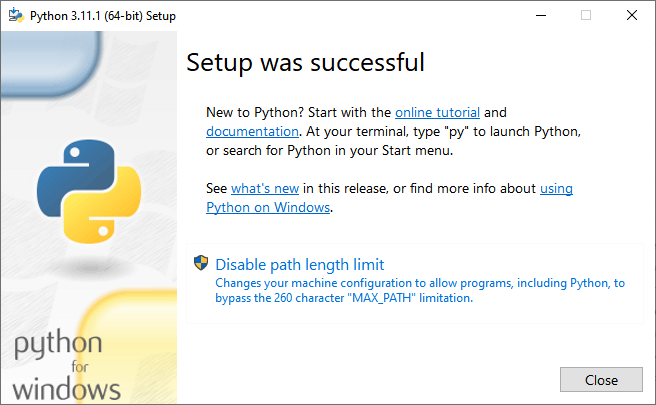
1. Нажимаем «Установить сейчас».



1. Загрузка начнётся автоматически.

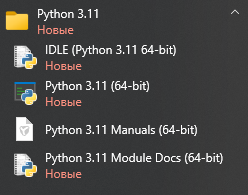


1. После установки будет выведено сообщение об его установке и предложения о знакомстве с некоторой документацией.



**Содержание установочного пакета:**

Установочный пакет включает в себя: IDLE, pip и документацию, также он создаёт ярлыки и ассоциации файлов.



**Куда устанавливается:**

В моём случае файл установился в:

C:\Users\Админ\AppData\Local\Programs\Python\Python39

**Какую версию можно устанавливать:**

На данный момент можно установить версию 3.9.1

**Версия, используемая мной:**

Версия, используемая на моём ПК является 3.10.6

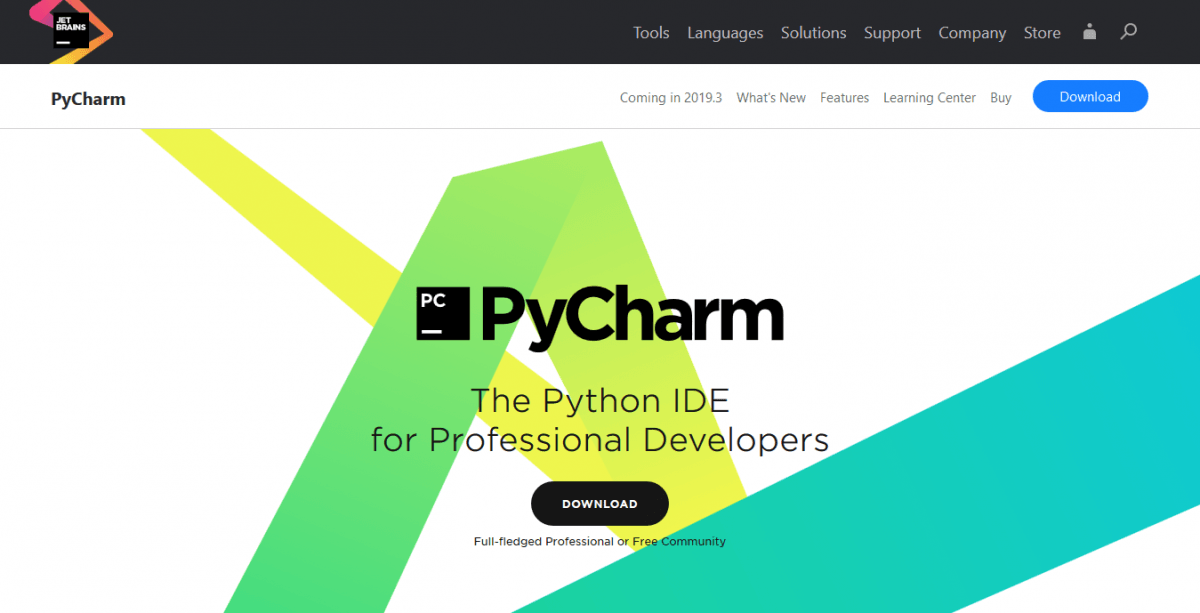
* + 1. **Установки и настройка IDE**

IDE, используемое мной является PyCharm.

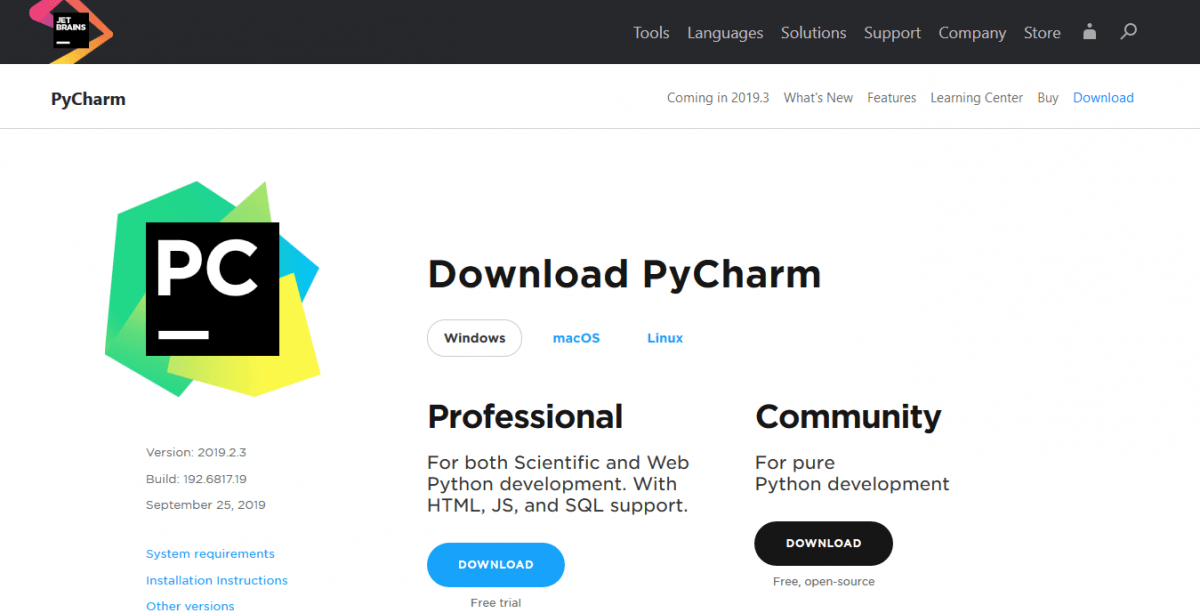
JetBrains PyCharm — это интегрированная среда разработки (IDE), используемая для написания программ на Python. В ней есть инструменты для анализа кода, графический отладчик, встроенное модульное тестирование и поддержка веб-разработки с Django.

**Установка PyCharm:**

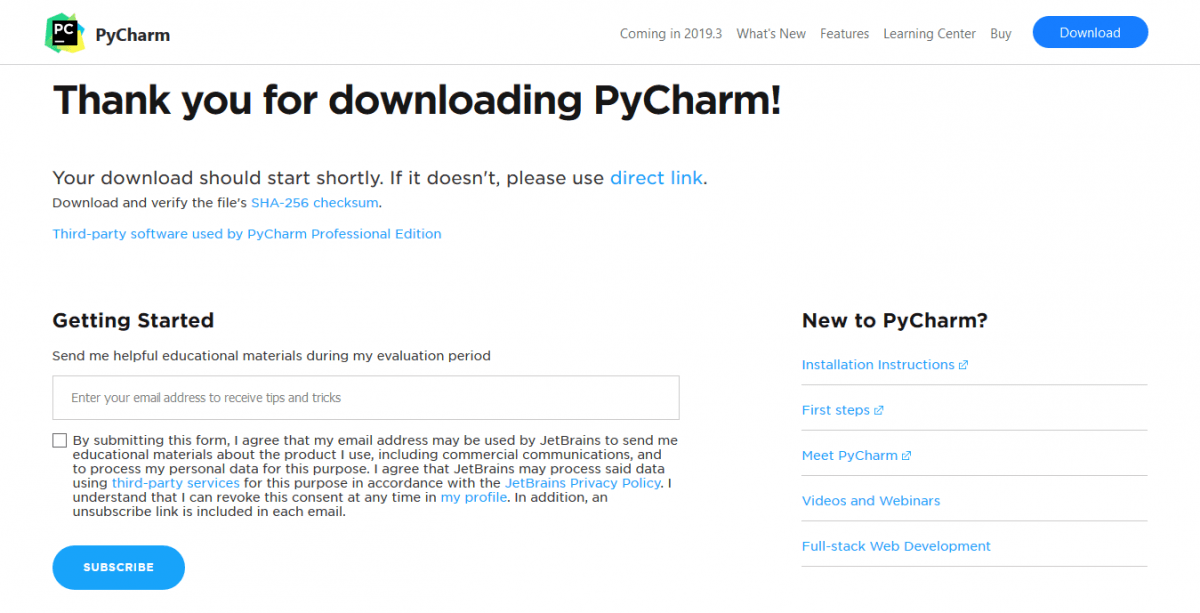
1. Перейдите на сайт JetBrains [**https://www.jetbrains.com/pycharm/**](https://www.jetbrains.com/pycharm/). PyCharm — это проект JetBrains.



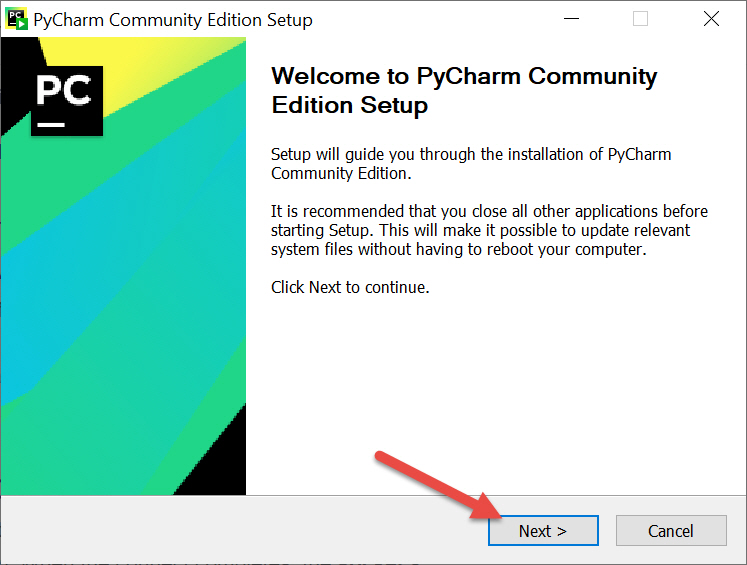
1. Нажмите «Download». Произойдет перенаправление на страницу загрузки PyCharm.



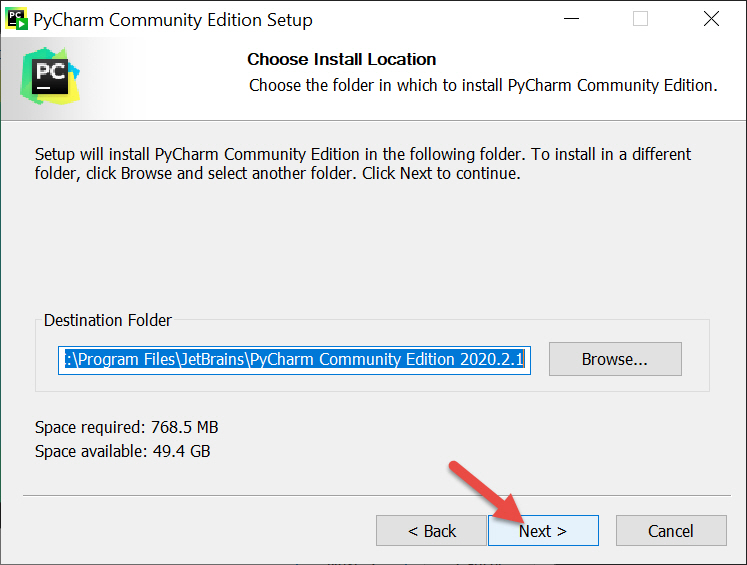
1. Дальше необходимо выбрать версию PyCharm: **Community** или **Professional**. Professional — это платная версия с полным набором функций. Она идеально подходит для профессиональной разработки. Community бесплатная. Ею можно пользоваться благодаря набору базовых возможностей. После нажатия на кнопку «Загрузить» скачивание начнется автоматически.



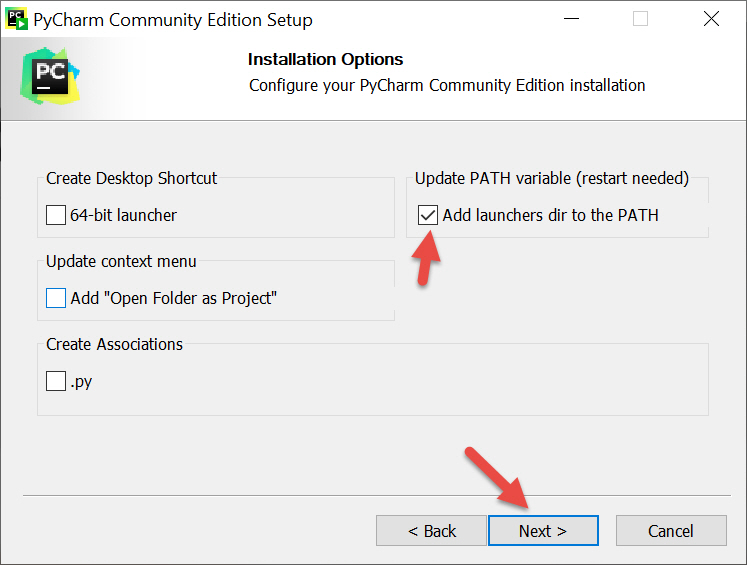
1. Теперь необходимо запустить установку. Нажмите «Next».



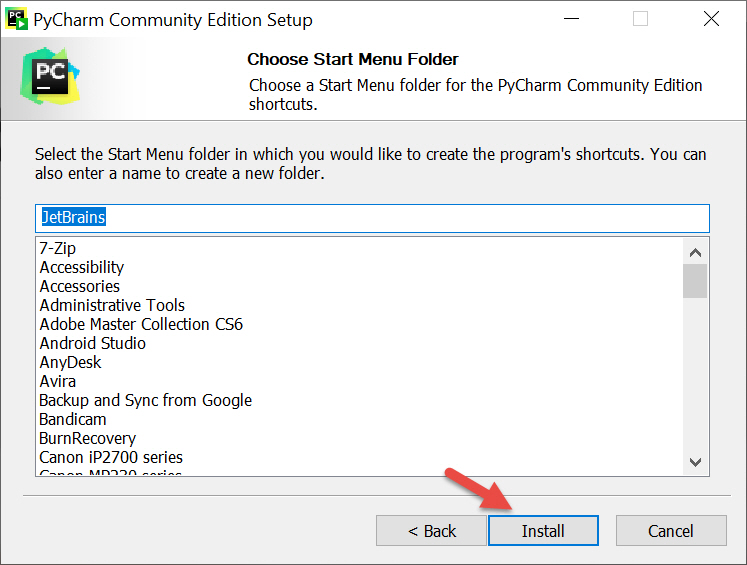
1. Оставьте папку установки по умолчанию. Это папка C:\Program files(x86)\JetBrains\PyCharm Community Edition 2019.2. Нажмите «Next».



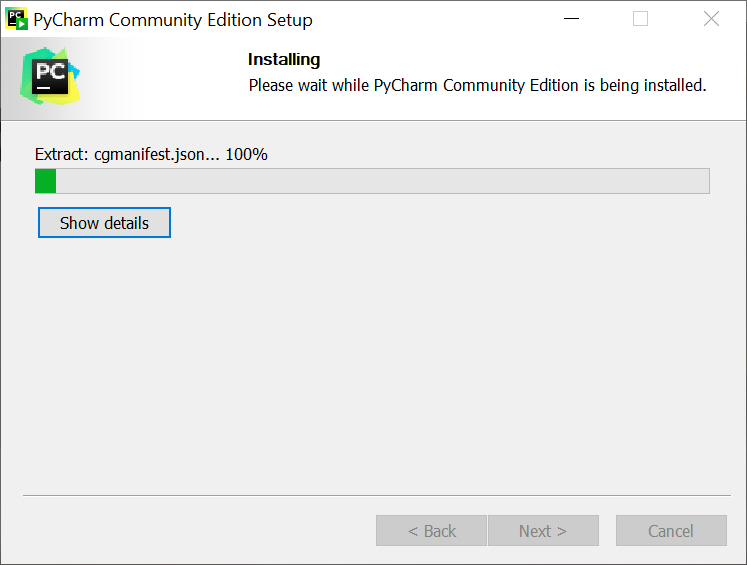
1. Поставьте галочки напротив тех пунктов, что вам необходимы. Нажмите «Next».



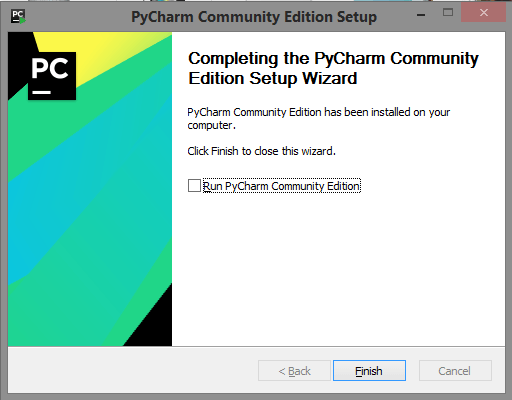
1. Далее установщик попросит указать имя для отображения в стартовом меню. По умолчанию это JetBrains. Нажмите «Install».



1. Установка начнется автоматически. Дождитесь завершения.



1. После завершения установки поставьте галочку напротив «Run PyCharm Community Edition» для запуска, а затем «Finish».



**Настройка PyCharm:**

Среду IDE можно настроить так, чтобы она соответствовала потребностям и была удобна для пользователя. Для этого необходимо зайдите в меню **File / Settings**, чтобы просмотреть список доступных параметров настройки.

|  |
| --- |
| [File / Settings PyCharm](https://4.bp.blogspot.com/-q60uNwukW1Q/WbE4eFKpRDI/AAAAAAAAAHw/899liYDdpCsuAsK4SRgtSi4nPKe83UudACLcBGAs/s1600/2017-09-07_15-11-57.png) |
| File / Settings PyCharm |
| [Settings PyCharm](https://1.bp.blogspot.com/-hI8ciJPzAXM/WbE47GzUuOI/AAAAAAAAAH0/NhAea_2f7iUN8Z2Ojy4o_C2nA1uJSq5FQCLcBGAs/s1600/2017-09-07_15-15-10.png) |
| Settings PyCharm |

**Вот, например, некоторые из них:**

**Внешний вид IDE**

Первое, что нужно подстроить – это общий «внешний вид». Зайдите в меню **File / Settings /Appearance and Behavior /Appearance**, чтобы выбрать Тему IDE: тема по умолчанию, или Dracula, если вы предпочитаете более тёмную настройку:

|  |
| --- |
| [PyCharm File / Settings / Appearance and Behavior /Appearance](https://2.bp.blogspot.com/-TnzgEqOU2uY/WbE6-ErUZFI/AAAAAAAAAIA/pdPp_uLFkCUp6VUOAJnuYiHNQWyRS4bwwCLcBGAs/s1600/2017-09-07_15-23-05.png) |
| File / Settings / Appearance and Behavior /Appearance |

**Редактор**

Множество вкладок, доступных в меню **File / Settings / Editor** помогут настроить каждый аспект поведения редактора. Здесь доступно множество опций, начиная с общих настроек. Для настройки цвета для каждого доступного языка и варианта использования, для вкладок и настроек свертки кода, для поведения завершения кода и.т.д

|  |
| --- |
| [PyCharm File / Settings / Editor](https://1.bp.blogspot.com/-LCD5pgAovRg/WbE8jwi0myI/AAAAAAAAAIM/VAgVbI9I2MUeiGkiaPU27ODnaESDl3BqQCLcBGAs/s1600/2017-09-07_15-27-58.png) |
| File / Settings / Editor |

**Стиль кода**

Стиль кода может быть определён для каждого языка **File / Settings / Editor / Code Style**. Также можете создать и сохранить свой собственный стиль кода.

|  |
| --- |
| [PyCharm File / Settings / Editor / Code Style](https://3.bp.blogspot.com/-8zQo3y6zcwg/WbE-4zrfaEI/AAAAAAAAAIY/Pq41eR5y5X4rzMB-enkDz8svFKyli96CgCLcBGAs/s1600/2017-09-07_15-40-38.png) |
| File / Settings / Editor / Code Style |

**Раскладка**

PyCharm использует подход, ориентированный на клавиатуру, что означает, что почти все действия, доступные в среде IDE, сопоставляются с комбинациями клавиш.  
Горячие клавиши – одна из ваших личных привычек. PyCharm предоставляет стандартную раскладку (выберите **Help / Keymap Reference** в главном меню), делая кодирование действительно продуктивным и удобным. Однако её всегда можно изменить **File / Settings / Keymap**

|  |
| --- |
| [Help / Keymap Reference](https://1.bp.blogspot.com/-_EL69NDOCRo/WbFA8hvrFNI/AAAAAAAAAIk/qQLNzOlD0xASXzkJqp5qjQcOvd2HYIV1QCLcBGAs/s1600/2017-09-07_15-47-43.png) |

Help / Keymap Reference

* + 1. **Особенности задания имен переменных и правила организации кода в Python**

**Особенности задания имён переменных:**

Переменные в Python не требуют объявления типа переменной (так как Python – язык с динамической типизацией) и являются ссылками на область памяти. Правила именования переменных:

* имя переменной может состоять только из букв, цифр и знака подчёркивания
* имя не может начинаться с цифры
* имя не может содержать специальных символов @, $, %

Также имена переменных не должны пересекаться с названиями операторов и модулей или же других зарезервированных слов. В Python есть рекомендации по именованию функций, классов и переменных:

* имена переменных обычно пишутся или полностью большими или полностью маленькими буквами (например DB\_NAME, db\_name);
* имена функций задаются маленькими буквами, с подчёркиваниями между словами (например, get\_names);
* имена классов задаются словами с заглавными буквами без пробелов, это так называемый CamelCase (например, CiscoSwitch).

**Правила организации кода в Python:**

**Концептуальные требования:**

* Система классов программы должна быть осмысленной и отвечать парадигме объектно-ориентированного программирования.
* Методы не должны занимать более 25 строк. Если метод занимает большее количество строк, значит он выполняет несколько независимых операций и требует разделения на более мелкие методы.
* Категорически запрещается дублирование одинакового кода!
* Запрещается собственная реализация средств, имеющих аналоги в стандартной библиотеке Python. Во всех случаях, когда возможно использование библиотечных классов Python, они должны быть использованы.
* Рекомендуется использовать шаблоны проектирования во всех случаях, когда это представляется уместным.
* Не допускается выполнять оптимизацию кода, ухудшающую его качество, если это не является действительно необходимым. В случае применения оптимизации, ее реализация должна быть скрыта внутри класса/модуля.
* Все возможные ошибочные ситуации должны корректно обрабатываться. Сообщения об ошибках должны быть максимально информативными.
* Сравнение с None должно осуществляться только с помощью операции is:

if x is None and z is not None:

pass

**Внешний вид кода:**

* Используйте 4 пробела на один уровень отступа. Символы табуляции запрещены
* Ограничьте максимальную длину строки 90 символами
* В случае если строка больше 90 символов, осуществите перенос на новую строку с двойным отступом относительно текущего отступа
* Отделяйте функции верхнего уровня и определения классов двумя пустыми строками. Определения методов внутри класса отделяйте одной пустой строкой
* Не рекомендуется размещение нескольких операторов в одной строке, кроме случаев, когда это не ухудшает удобочитаемости программы
* При импорте имён из модулей с помощью инструкции from запрещается использовать форму со звёздочкой
  1. **Библиотека для анализа данных Pandas**
     1. **Описание библиотеки Pandas**

**О pandas**

**История разработки**

В 2008 году в AQR Capital Management началась разработка pandas. К концу 2009 года она была с открытым исходным кодом, и сегодня ее активно поддерживает сообщество единомышленников по всему миру, которые вкладывают свое драгоценное время и энергию в создание pandas с открытым исходным кодом. Спасибо всем нашим участникам.

С 2015 года pandas является проектом, спонсируемым NumFOCUS. Это поможет обеспечить успех разработки pandas как проекта с открытым исходным кодом мирового уровня.

**Временная шкала**

2008: Начато развитие pandas

2009: pandas становится открытым исходным кодом

2012: Опубликовано первое издание Python для анализа данных

2015: pandas становится проектом, спонсируемым NumFOCUS

2018: Первый очный спринт для разработчиков ядра

**Основные моменты библиотеки**

* Быстрый и эффективный объект DataFrame для обработки данных со встроенной индексацией;
* Инструменты для чтения и записи данных между структурами данных в памяти и различными форматами: CSV и текстовые файлы, Microsoft Excel, базы данных SQL и быстрый формат HDF5;
* Интеллектуальное выравнивание данных и интегрированная обработка недостающих данных: автоматическое выравнивание на основе меток при вычислениях и простое преобразование беспорядочных данных в упорядоченную форму.;
* Гибкое изменение формы и поворот наборов данных;
* Интеллектуальное нарезание на основе меток, причудливое индексирование и подмножество больших наборов данных;
* Столбцы можно вставлять и удалять из структур данных для изменения размера;
* Агрегирование или преобразование данных с помощью мощного механизма группирования, позволяющего выполнять операции разделения, применения и объединения наборов данных;
* Высокопроизводительное объединение и объединение наборов данных;
* Иерархическое индексирование по осям обеспечивает интуитивно понятный способ работы с многомерными данными в низкоразмерной структуре данных;
* Временные ряды - функциональность: генерация диапазона дат и преобразование частоты, статистика движущихся окон, сдвиг даты и задержка. Даже создавайте временные смещения для конкретного домена и объединяйте временные ряды без потери данных.;
* Высоко оптимизированный по производительности, с критическими путями кода, написанными на Cython или C.
* Python с pandas используется в самых разных академических и коммерческих областях, включая финансы, нейробиологию, экономику, статистику, рекламу, веб-аналитику и многое другое.

**Миссия**

Pandas стремится стать фундаментальным высокоуровневым строительным блоком для практического анализа данных в реальном мире на Python. Кроме того, у него есть более широкая цель - стать самым мощным и гибким инструментом анализа / обработки данных с открытым исходным кодом, доступным на любом языке.

Информация взята с официального сайта: <https://pandas.pydata.org/about/>

* + 1. **Практическая часть**

**Задание:**

Вам необходимо написать программу, которая будет хранить информацию в csv файле и реализовать приведенный ниже функционал вашего варианта. Для выполнения необходимо использовать библиотеку pandas.

Вариант 15:

Необходимо написать программу, которая будет хранить информацию о расписании поездов (например, такую как: название пункта назначения и отправления, номер поезда, время отправления, время в пути, станции на которых делается остановка (вложенный список)).

Реализуйте следующий функционал:

* Добавление новой записи (номер поезда должно быть уникальным).
* Удаление записи по номеру поезда (учитывать, что такой номер может отсутствовать в словаре).
* Вывести все записи по введённому пункту назначения
* Вывести все записи о поездах, которые отправляются в указанный интервал времени.
* Вывести все станции остановки указанного поезда по его номеру
* Вывести количество поездов идущих из указанной станции
* Вывести среднее время в пути всех поездов с указанной станции отправления
* Любая ваша функция.

Информация о 10 записях уже должны быть занесена в программу (из кода).

При реализации данного функционала необходимо реализовать диалог с пользователем (например, каждой функции можно присвоить номер, когда пользователь его введет, функция будет выполнена).

**Примеры выполнения нескольких функций:**

1. Функция открытия файла

def file\_open():

try:

data = pandas.read\_csv('data.csv', delimiter=";")

except Exception as e:

print(e)

print('Файл открыт. Записей:', len(data.index))

return data



1. Функция добавления

def insert(fio, male, age, tel, mailru, jobtitle, stagee):

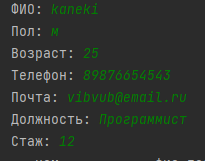
global csv\_file

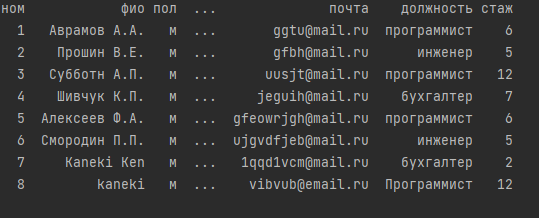
df\_insert = pandas.DataFrame([(max(csv\_file['ном'] + 1), fio, male,age, tel, mailru, jobtitle, stagee)],

columns=('ном', 'фио', 'пол', 'возраст', 'телефон', 'почта', 'должность', 'стаж'))

df2 = pandas.concat([csv\_file, df\_insert])

return df2





1. Вывести средний возраст

def avg\_age():

print("Средний возраст:", csv\_file["возраст"].mean())



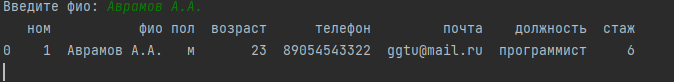
1. Поиск по фамилии

def find\_by\_FIO(val, col\_name):

global csv\_file

fio\_find = csv\_file[csv\_file[col\_name].isin([val])]

print(fio\_find)



**Программный код:**

Приложения 1. «Скрипт выполнения задания Pandas»

* 1. **Создание пользовательской модели классов**
     1. **Основные работы с пользовательскими классами в Python**

**Классы**

Классы предоставляют средства объединения данных и функциональности вместе. Создание нового класса создает объект нового типа, позволяя создавать новые экземпляры этого типа. К каждому экземпляру класса могут быть прикреплены атрибуты для поддержания его состояния. Экземпляры класса также могут иметь методы (определенные его классом) для изменения его состояния.

По сравнению с другими языками программирования, механизм классов Python добавляет классы с минимальным количеством нового синтаксиса и семантики. Это смесь механизмов классов, найденных в C ++ и Modula-3. Классы Python предоставляют все стандартные функции объектно-ориентированного программирования: механизм наследования классов допускает несколько базовых классов, производный класс может переопределять любые методы своего базового класса или классов, а метод может вызывать метод базового класса с тем же именем. Объекты могут содержать произвольные объёмы и типы данных. Как и в случае с модулями, классы обладают динамической природой Python: они создаются во время выполнения и могут быть изменены после создания.

В терминологии C ++ обычно члены класса (включая элементы данных) являются общедоступными (за исключением частных переменных, см. Ниже), а все функции-члены являются виртуальными. Как и в Modula-3, нет сокращений для ссылки на члены объекта из его методов: функция метода объявляется с явным первым аргументом, представляющим объект, который неявно предоставляется вызовом. Как и в Smalltalk, классы сами по себе являются объектами. Это обеспечивает семантику для импорта и переименования. В отличие от C ++ и Modula-3, встроенные типы могут использоваться в качестве базовых классов для расширения пользователем. Также как и в C ++, большинство встроенных операторов со специальным синтаксисом (арифметические операторы, индексы и т. Д.) Могут быть переопределены для экземпляров класса.

* + 1. **Практическая часть**

**Задание:**

На основе вашего варианта из задания со словарями вам необходимо разработать набор собственных классов.

Классы:

* Employee(сотрудник)–номер, ФИО, пол, возраст, почта, должность, стаж;
* Departments(отделы) – название, номер, имя начальника;
* Kids(дети) – ФИО, возраст, пол;
* Car(автомобиль)-Марка, модель, номер;

Методы:

* Вывести сотрудника;
* Вывести общий стаж;
* Увеличение возраста ребенка;
* Изменение стоимости автомобиля;
* Добавление сотрудника в отдел

Дополните модель собственными классами, свойствами и методами при необходимости.

**Пример работы с созданными классами:**

1. Перехват функции print, когда она преобразует свое значение в строку

def \_\_str\_\_(self):

return f"Имя ребенка: {self.name} ,Возраст ребенка: {self.age} ,Пол ребенка: {self.male}"



1. Добавляем сотрудника в департамент

def append(self, emp):

if type(emp) != Employee:

raise TypeError("Атрибут содержит тип {} вместо Employee".format(type(emp)))

self.employees.append(emp)



1. Сравнить сотрудников по стажу

# Здесь и ниже операции сравнения >, >=, <, <=, ==, !=

def \_\_eq\_\_(self, other):

return self.stage == other.stage

def \_\_lt\_\_(self, other):

return self.stage > other.stage

def \_\_le\_\_(self, other):

if self.\_\_eq\_\_(other):

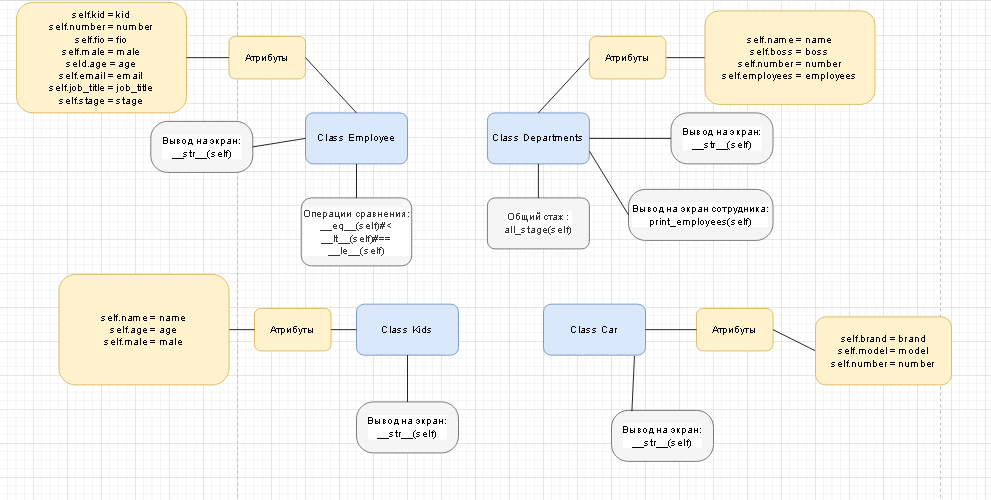
return True

if self.\_\_lt\_\_(other):

return True



**Карта классов:**

****

**Программный код:**

Приложения 2. «Скрипт выполнения задания с классами»

* 1. **Построение интерфейса приложения**
     1. **Библиотека PyQt5**

PyQt5 является одним из наиболее часто используемых модулей для создания GUI приложений в Python, и это связанно с его простотой, о которой вы узнаете далее.

Отличительная особенность, которая вдохновляет разработчиков пользоваться PyQt5 – это PyQt5 Designer, благодаря которому можно создавать сложные GUI приложения достаточно быстро. Вам нужно только перетаскивать свои виджеты для создания собственной формы.

* + 1. **Практическая часть**

**Задание:**

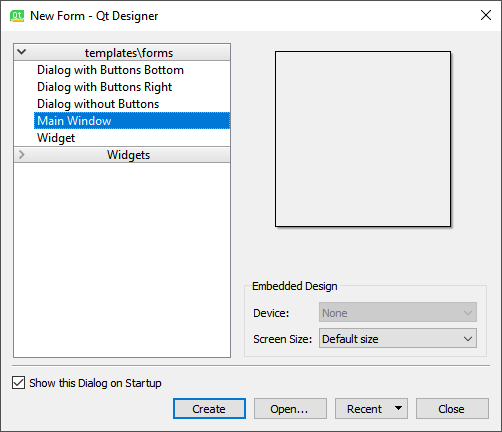
Разработайте форму для вашего варианта (предметной области). На форме отобразите все поля ввода. Данные при нажатии на кнопку добавить должны попадать в таблицу или в многострочное поле. При нажатии на кн. Сохранить реализовать сохранение данных в файл (txt или csv). При нажатии на кн. Открыть предусмотреть загрузку данных из файла на форму.

**Процесс создания интерфейса в qtDesigner:**

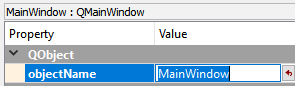
1. Установите/ запустите qtDesigner.



1. Выберите тип окна Main Window:



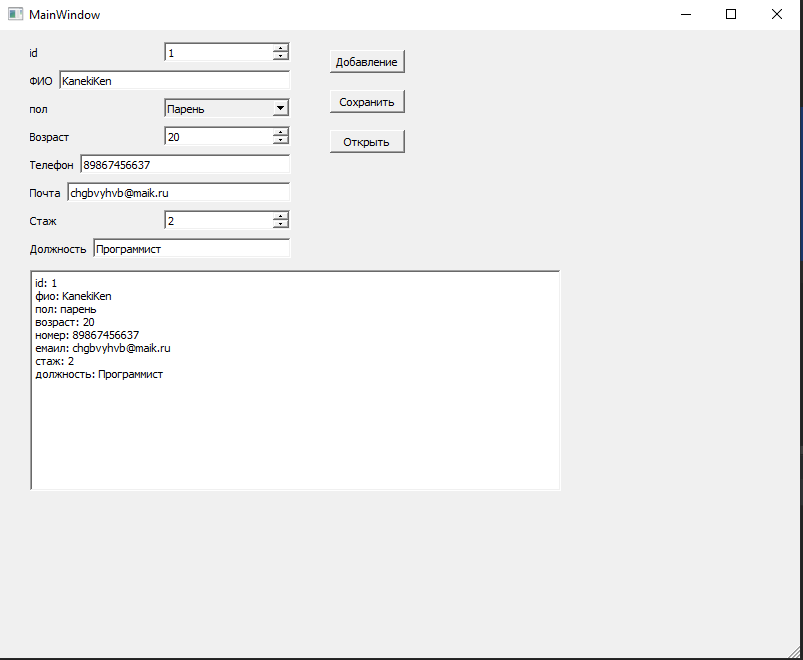
Укажите заголовок для вашего проекта:



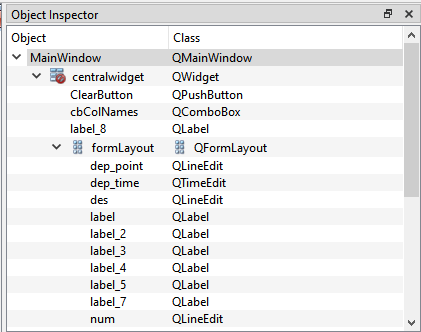
1. Разработайте форму для вашего варианта (предметной области).

На форме отобразите все поля ввода. Данные при нажатии на кнопку добавить должны попадать в таблицу или в многострочное поле. При нажатии на кн. Сохранить реализовать сохранение данных в файл (txt или csv). При нажатии на кн. Открыть предусмотреть загрузку данных из файла на форму.

Интерфейс:

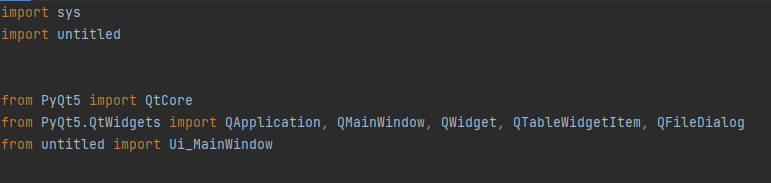


Измените имена объектов в соответствии с их назначением:

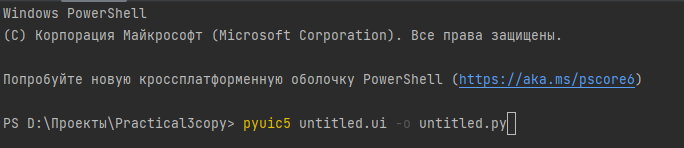


**Процесс разработки функционала по вашему варианту:**

1. Создайте новый проект. Файл main будет главным вашего проекта.



1. Создайте на основе вашего файла интерфейса библиотеку main\_form.py Команда в терминале: pyuic5 main\_form.ui -o main\_form.py



1. Подключите созданный файл к вашему проекту. Он должен находиться в той же директории:

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget

class MyWidget(QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.pbAdd.clicked.connect(self.Add)

self.pbSave.clicked.connect(self.Save)

self.pbOpen.clicked.connect(self.Open)

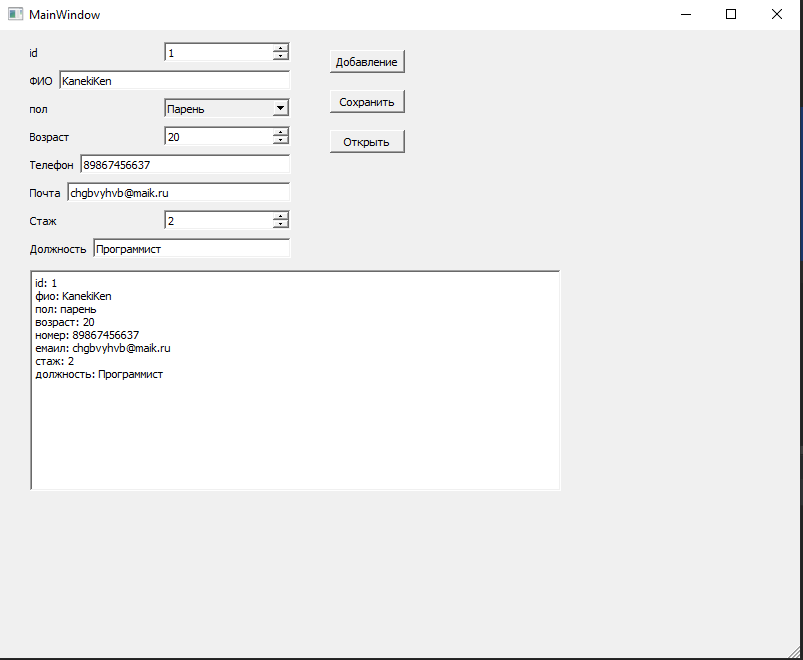
if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = MyWidget()

ex.show()

sys.exit(app.exec\_())



**Программный код:**

Приложение 3. «Скрипт выполнения задания с QtDesigner»

* 1. **Написание локального веб-сервера при помощи модуля flask**
     1. **Введение во flask**

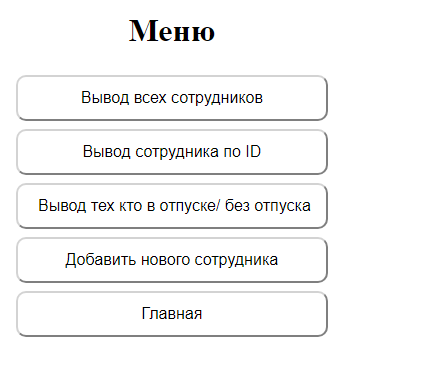
Flask — это упрощенная платформа Python для веб-приложений, которая обеспечивает основные возможности маршрутизации URL-адресов и визуализации страниц.

Flask называют "микро"- платформой, так как она не предоставляет напрямую такие функции, как проверка форм, абстракция базы данных, проверка подлинности и т. д. Эти функции предоставляются специальными пакетами Python, называемыми расширениями Flask. Расширения легко интегрируются с Flask и отображаются так, как будто являются частью самой среды Flask. Например, Flask не предоставляет модуль шаблонов страницы. Использование шаблонов обеспечивается расширениями, такими как Jinja и Jade, как демонстрируется в этом учебнике.

Flask используется для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков - минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

* + 1. **Написание веб-сервера для своего варианта**

**Реализуемые функции:**

****

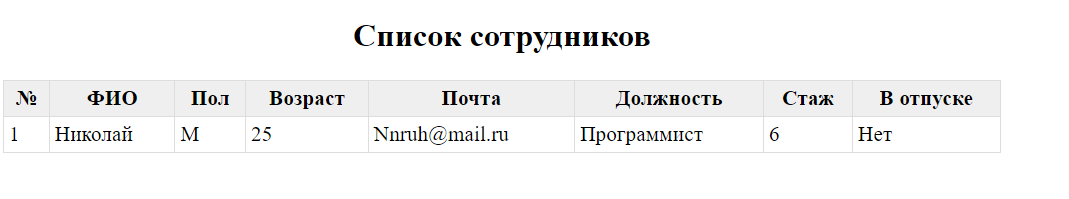
1. **Вывод всех поездов**

Выводит в таблицу список всех сотрудников из файла json.

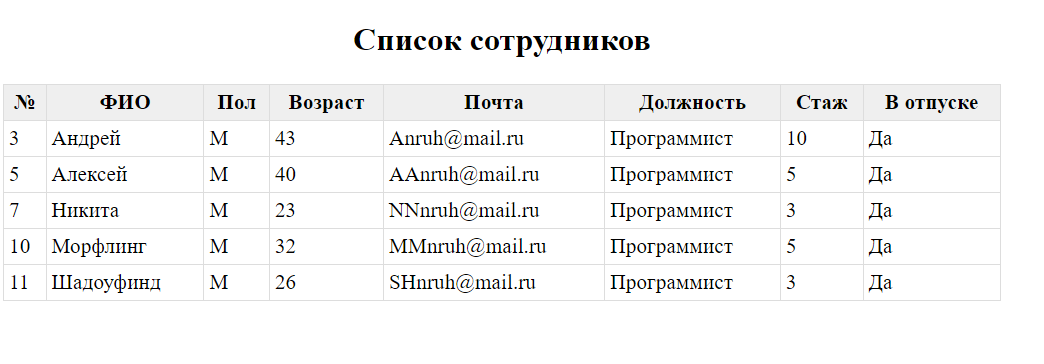
****

1. **Вывод сотрудников по ID**

Выводит в таблице поезд, ID которого было введено в диалоговом окне.

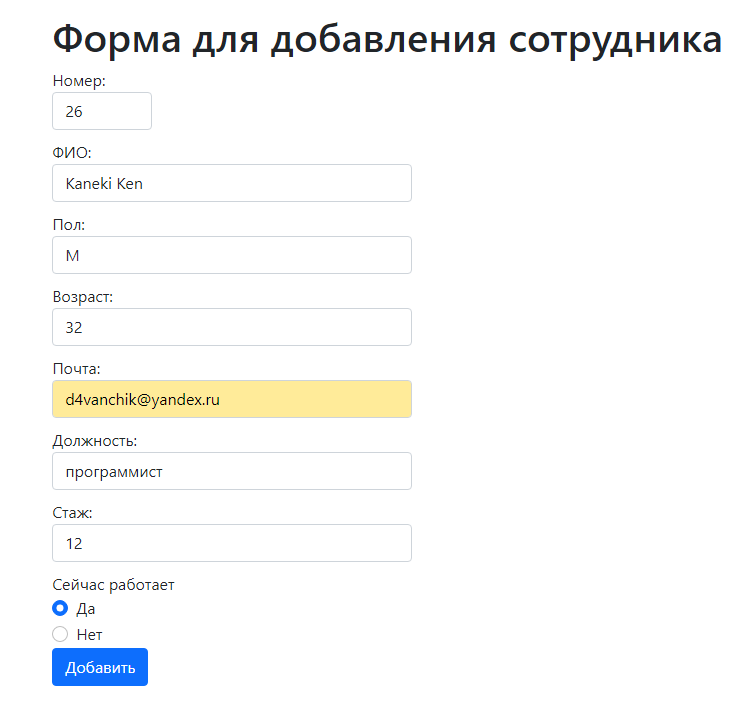
****

1. **Вывод тех кто в отпуске**

****

1. **Добавить нового сотрудника**

Переводит на форму добавления сотрудника, после заполнения полей по кнопке «Добавить» произойдёт добавление сотрудника.

****

1. **Главная**

Переносит на главную форму

****

**Вид вывода информации:**

Информация выводится в таблице, данные которые получает из файла json.

**Программный код:**

Приложение 4. «Скрипт выполнения задания с flask»

* 1. **Юнит тестирование Pytest**
     1. **Юнит-тестирование в Python**

Юнит-тестирование – тестирование программных модулей приложения с помощью программных процедур.

Библиотека pytest – это основанная на Python среда тестирования, которая используется для написания и выполнения тестовых кодов. В настоящее время службы REST pytest в основном используются для тестирования API, хотя можно использовать pytest для написания простых и сложных тестов, т.е. мы можем писать коды для тестирования API, БД, пользовательского интерфейса и т.д.

Пример теста:

class Rectangle():

def \_\_init\_\_(self, a, b):

if type(a) != int:

raise TypeError('Expected int, got {}'.format(type(a)))

if type(b) != int:

raise TypeError('Expected int, got {}'.format(type(b)))

self.a = a

self.b = b

def get\_area(self):

return self.a \* self.b

def get\_perimeter(self):

return self.a \* 2 + self.b \* 2

* + 1. **Написание юнит-тестов для своей предметной области**

Программный код:

Приложение 5. «Скрипт выполнения задания с Unit тестами»

# Вывод по разделу

В результате выполнения практических заданий по разработке программных модулей на языке Python была проведена работа с такими модулями как: Pandas, Classes, PyQt5, Flask, Pytest.

В пункте 1.1 была описана среда разработки и особенности языка Python.

В пункте 1.2 был создан скрипт по работе с csv файлами и библиотекой Pandas моего варианта.

В пункте 1.3 был создан скрипт по работе с классами для моего варианта.

В пункте 1.4 был создан скрипт по работе с библиотекой PyQt5 для моего варианта.

В пункте 1.5 был создан скрипт по работе с библиотекой Flask для моего варианта.

В пункте 1.6 был создан скрипт по работе с Unit-тестами для моего варианта.

# 

# Приложения

**Приложения 1. «Скрипт выполнения задания Pandas»**

import pandas

csv\_file = pandas.read\_csv('data.csv', delimiter=";")

menu = {

'1': 'Открыть файл',

'2': 'Добавить',

'3': 'Удалить',

'4': 'Поиск сотрудников по ФИО',

'5': 'Вывести средний возраст',

'6': 'Найти по дожности',

'7': 'Вывести год рождения для каждого сотрудника',

'8': 'Вывести средний стаж',

'9': 'Вывести данные о всех сотрудниках',

'10': 'Вывести общий стаж асех работников',

'11': 'Сохранить в файл',

'0': '<-Меню',

'exit': 'Выход'

}

def file\_open():

try:

data = pandas.read\_csv('data.csv', delimiter=";")

except Exception as e:

print(e)

print('Файл открыт. Записей:', len(data.index))

return data

def insert(fio, male, age, tel, mailru, jobtitle, stagee):

global csv\_file

df\_insert = pandas.DataFrame([(max(csv\_file['ном'] + 1), fio, male,age, tel, mailru, jobtitle, stagee)],

columns=('ном', 'фио', 'пол', 'возраст', 'телефон', 'почта', 'должность', 'стаж'))

df2 = pandas.concat([csv\_file, df\_insert])

return df2

def drop\_by\_arg(val, col\_name='фио'):

global csv\_file

if col\_name == 'ном' or col\_name == 'возраст':

val = int(val)

csv\_file = csv\_file.set\_index(col\_name)

csv\_file.drop(val, axis=0, inplace=True)

print(csv\_file)

def find\_by\_FIO(val, col\_name):

global csv\_file

fio\_find = csv\_file[csv\_file[col\_name].isin([val])]

print(fio\_find)

def avg\_age():

print("Средний возраст:", csv\_file["возраст"].mean())

def find\_by\_job\_title(val, col\_name):

global csv\_file

df = csv\_file[csv\_file[col\_name].isin([val])]

print(df)

def employee\_year\_of\_birth():

global csv\_file

a = [[csv\_file["фио"], csv\_file["возраст"]]]

print(a)

def avg\_stage():

aa = int(csv\_file["стаж"].mean())

print(f" Средний возраст в целочисленном: {aa}\n", "не в целоцисленном:", csv\_file["стаж"].mean())

def all\_emplyees():

global csv\_file

for i in csv\_file:

print(csv\_file[i])

def save():

try:

csv\_file.to\_csv('data.csv', index=False, sep=';')

except Exception as e:

print(e)

def stage\_sum():

print("общий стаж:", csv\_file["стаж"].sum(), "года")

print('\n'.join([f"{k} : {v}" for k, v in menu.items()]))

while True:

comand = input()

if comand == '1':

csv\_file = file\_open()

elif comand == '2':

csv\_file = insert(input('ФИО: '), input('Пол: '),

int(input('Возраст: ')), input('Телефон: '), input('Почта: '), input('Должность: '), input('Стаж: '))

print(csv\_file)

elif comand == '3':

col = input('Колонка: ')

val = input('Значение: ')

drop\_by\_arg(val, col\_name=col)

elif comand == "4":

val = input("Введите фио: ")

col = "фио"

find\_by\_FIO(val, col\_name=col)

elif comand == '5':

avg\_age()

elif comand == '6':

col = "должность"

val = input('Значение: ')

find\_by\_job\_title(val, col\_name=col)

elif comand == '7':

employee\_year\_of\_birth()

elif comand == '8':

avg\_stage()

elif comand == '9':

all\_emplyees()

elif comand == '10':

stage\_sum()

elif comand == '11':

save()

elif comand == '0':

print('\n'.join([f"{k} : {v}" for k, v in menu.items()]))

else:

break

**Приложения 2. «Скрипт выполнения задания с классами»**

class Employee:

def \_\_init\_\_(self, number, fio, male, age, email, job\_title, stage, kid=None, car=None):

if kid is None:

kid = []

self.kid = kid

if car is None:

car = []

self.car = car

self.number = number

self.fio = fio

self.male = male

self.age = age

self.email = email

self.job\_title = job\_title

self.stage = stage

def \_\_str\_\_(self):

return f": {self.number} ,{self.fio} ,{self.male} ,{self.age} ,{self.email} ,{self.job\_title}, {self.stage}, {self.kid}"

def \_\_eq\_\_(self, other):

return self.stage == other.stage

def \_\_lt\_\_(self, other):

return self.stage > other.stage

def \_\_le\_\_(self, other):

if self.\_\_eq\_\_(other):

return True

if self.\_\_lt\_\_(other):

return True

class Departments:

def \_\_init\_\_(self, name, number, boss=None, employees=None):

if employees is None:

employees = []

self.name = name

self.boss = boss

self.number = number

self.employees = employees

def append(self, emp):

if type(emp) != Employee:

raise TypeError("Атрибут содержит тип {} вместо Employee".format(type(emp)))

self.employees.append(emp)

def \_\_str\_\_(self):

return f"{self.\_\_class\_\_.\_\_name\_\_},{self.employees}"

def print\_employees(self):

return self.employees

def all\_stage(self):

a = 0

for emp in self.employees:

a += emp.stage

return a

class Kids:

def \_\_init\_\_(self, name, age, male):

self.name = name

self.age = age

self.male = male

def \_\_str\_\_(self):

return f"{self.name} ,{self.age} ,{self.male}"

def increase\_age(self, number):

self.age += number

class Car:

def \_\_init\_\_(self, brand ,model ,number, price):

self.brand = brand

self.model = model

self.number = number

self.price = price

def \_\_str\_\_(self):

return f"{self.brand} ,{self.model} ,{self.number}, {self.price}"

def increase\_price(self, summa):

self.price += summa

def \_\_eq\_\_(self, other):

return self.price == other.price

def \_\_lt\_\_(self, other):

return self.price > other.price

def \_\_le\_\_(self, other):

if self.\_\_eq\_\_(other):

return True

if self.\_\_lt\_\_(other):

return True

p = Employee(1, "Иван", "М", 18, "iabutalibli@inbox.ru", "Программист", 4)

p1 = Employee(2, "Бебрик", "М", 18, "iafsdghdfh@inbox.ru", "Инженер-Программист", 5)

p2 = Employee(3, "Kaneki", "М", 18, "ihdfhhdfhi@inbox.ru", "Программист-Инженер", 6)

p3 = Employee(4, "Олег", "М", 18, "ifewwgtywbli@inbox.ru", "Инженер", 2)

d = Departments("Архив", 1, "ПРОСТОБОСС", employees=[p, p1, p2, p3])

print(d.all\_stage())

k1 = Kids("Бебрик1", 10, "м")

k2 = Kids("Миша", 7, "м")

k3 = Kids("Оля", 9, "ж")

e1 = Employee(5, "Иван", "М", 18, "iabutalibli@inbox.ru", "Программист", 4, kid=k1)

print(e1)

c1 = Car("BMW", "X5", "Х217АВ", 1500000)

c2 = Car("BMW", "X3", "С439УК", 2000000)

c3 = Car("BMW", "X1", "Н768ОО", 6000000)

car11 = Car("MERCEDES", "A-CLASS", "Е620УХ", 8000000)

car22 = Car("MERCEDES", "B-CLASS", "Т191ТО", 3000000)

car33 = Car("MERCEDES", "C-CLASS", "А239НС", 2400000)

e2 = Employee(6, "Иван", "М", 18, "iabutalibli@inbox.ru", "Программист", 4, kid=k1, car=car11)

**Приложение 3. «Скрипт выполнения задания с QtDesigner»**

import sys

import untitled

from PyQt5 import QtCore

from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QTableWidgetItem, QFileDialog

from untitled import Ui\_MainWindow

class MyWidget(QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.pbAdd.clicked.connect(self.Add)

self.pbSave.clicked.connect(self.Save)

self.pbOpen.clicked.connect(self.Open)

def Add(self):

\_translate = QtCore.QCoreApplication.translate

try:

id = self.spinBox.text()

fio = self.lineEdit.text()

male = self.comboBox.currentIndex()

age = int(self.spinBox\_2.text())

phone = self.lineEdit\_2.text()

mailru = self.lineEdit\_3.text()

stage = int(self.spinBox\_3.text())

jobtitlt = self.lineEdit\_4.text()

male\_list = ['заглушка(не выбирать)','парень', 'девушка']

self.textBrowser.setText(\_translate("Form", f'id: {id} \nфио: {fio} \nпол: {male\_list[male]} \nвозраст: {age} \nномер: {phone} \nемаил: {mailru} \nстаж: {stage} \nдолжность: {jobtitlt}'))

except BaseException:

self.textBrowser.setText( \_translate("Form", 'Неверный формат ввода данных'))

def Save(self):

\_translate = QtCore.QCoreApplication.translate

try:

id = self.spinBox.text()

fio = self.lineEdit.text()

male = self.comboBox.currentIndex()

age = int(self.spinBox\_2.text())

phone = self.lineEdit\_2.text()

mailru = self.lineEdit\_3.text()

stage = int(self.spinBox\_3.text())

jobtitlt = self.lineEdit\_4.text()

male\_list = ['заглушка(не нажимать)','парень', 'девушка']

if id and male and fio and age and phone and mailru and stage and jobtitlt:

f = open(r'papka\\' + id + '.txt', 'w')

f.write(id + ' ' + str(fio) + ' ' + str(male) + ' ' + str(age) + ' ' + ' ' + str(phone) + ' ' + str(

mailru) + ' ' + str(stage) + ' ' + str(jobtitlt))

f.close()

self.textBrowser.setText(\_translate("Form", f'id: {id} \nфио: {fio} \nпол: {male\_list[male]} \nвозраст: {age} \nномер: {phone} \nемаил: {mailru} \nстаж: {stage} \nдолжность: {jobtitlt} \n'))

else:

self.textBrowser.setText(\_translate("Form", 'Неверный формат ввода данных'))

except BaseException:

self.textBrowser.setText(\_translate("Form", 'Неверный формат ввода данных'))

def Open(self):

try:

fname = QFileDialog.getOpenFileName(self, 'Выбрать документ .txt', 'papka', 'информация (\*.txt)')[

0]

\_translate = QtCore.QCoreApplication.translate

f = open(fname, 'r')

id, fio, male, age, phone, mailru, stage, jobtitle = map(str, f.readline()[:-1].split())

male\_list = ['парень', 'девушка']

self.textBrowser.setText(\_translate("Form",

f'id: {id} \nфио: {fio} \nпол: {male\_list[int(male)]} \nвозраст: {age} \nномер: {phone} \nемаил: {mailru} \nстаж: {stage} \nдолжность: {jobtitle}'))

f.close()

except BaseException:

self.textBrowser.setText(

\_translate("Form", 'Неверный файл')) # в случае ошибки выводим сообщение о некорркетном файле

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = MyWidget()

ex.show()

sys.exit(app.exec\_())**Приложение 4. «Скрипт выполнения задания с flask»**

Основной код main:

import json

from pathlib import Path

from flask import Flask, request, url\_for, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

@app.route('/index')

def start():

return render\_template('index.html')

# Выволд всех сотрудников

@app.route('/employees')

def employees():

with open("organization.json", "rt", encoding="utf-8") as f:

organization\_list = json.loads(f.read())

return render\_template('employees.html', org=organization\_list)

# Вывод сотрудника по ID

@app.route('/employee/<int:id>')

def employee(id=None):

with open("organization.json", "rt", encoding="utf-8") as f:

organization\_list = json.loads(f.read())

return render\_template('employee.html', org=organization\_list, id=id)

# Вывод тех кто в отпуске/ без отпуска

@app.route('/employees\_on\_leave/<on\_leave>')

def employees\_on\_leave(on\_leave="True"):

with open("organization.json", "rt", encoding="utf-8") as f:

organization\_list = json.loads(f.read())

return render\_template('employees\_on\_leave.html', org=organization\_list, on\_leave=on\_leave.capitalize())

# Добавление сотрудника

@app.route('/add\_employee', methods=['POST', 'GET'])

def add\_employee():

if request.method == 'GET':

return f'''<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"

integrity="sha384-giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">

<link rel="stylesheet" type="text/css" href="{url\_for('static', filename='/css/style.css')}" />

<title>Добавление</title>

</head>

<body>

<h1>Форма для добавления сотрудника</h1>

<div>

<form class="login\_form" method="post">

<label class="form-check-label" for="id">Номер:</label>

<input type="number" class="form-control" id="id" name="id">

<label class="form-check-label" for="id">ФИО:</label>

<input type="text" class="form-control" id="fio" placeholder="Введите ФИО" name="fio">

<label class="form-check-label" for="id">Пол:</label>

<input type="text" class="form-control" id="male" name="male">

<label class="form-check-label" for="id">Возраст:</label>

<input type="number" class="form-control" id="age" name="age">

<label class="form-check-label" for="id">Почта:</label>

<input type="text" class="form-control" id="email" name="email">

<label class="form-check-label" for="id">Должность:</label>

<input type="text" class="form-control" id="job\_title" name="job\_title">

<label class="form-check-label" for="id">Стаж:</label>

<input type="number" class="form-control" id="stage" name="stage">

<div class="form-group">

<label for="form-check">Сейчас работает</label>

<div class="form-check">

<input class="form-check-input" type="radio" name="on\_leave" id="True" value="True" checked>

<label class="form-check-label" for="yes">

Да

</label>

</div>

<div class="form-check">

<input class="form-check-input" type="radio" name="on\_leave" id="False" value="False">

<label class="form-check-label" for="no">

Нет

</label>

</div>

</div>

<button type="submit" class="btn btn-primary">Добавить</button>

</form>

</div>

</body>

</html>'''

elif request.method == 'POST':

print(request.form['id'])

print(request.form['fio'])

print(request.form['male'])

print(request.form['age'])

print(request.form['email'])

print(request.form['job\_title'])

print(request.form['stage'])

print(request.form['on\_leave'])

path = Path('organization.json')

data = json.loads(path.read\_text(encoding='utf-8'))

data['Employees'].append(

{'id': int(request.form['id']), 'FIO': request.form['fio'], 'male': request.form['male'],

'age': int(request.form['age']), 'email': request.form['email'], 'job\_title': request.form['job\_title'],

'stage': request.form['stage'], 'on\_leave': request.form['on\_leave']})

path.write\_text(json.dumps(data), encoding='utf-8')

return "Новый сотрудник добавлен."

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=8081, host='127.0.0.2')

**Приложение 5. «Скрипт выполнения задания с Unit тестами»**

from mainp import Employee, Departments, Kids, Car

import pytest

# Проверка на отрицательный оклад

def test\_wrong\_stage():

with pytest.raises(ValueError):

employee\_1 = Employee(1, "Kaneki", "М", 20, "fgghd@mail.ru", "программист", -2, kid=None, car=None)

def test\_wrong\_type\_Employee():

with pytest.raises(TypeError):

employee\_1 = Employee("1", "Kaneki", "М", 20, "fgghd@mail.ru", "программист", 2, kid=None, car=None)

def test\_employees\_lt():

employee\_1 = Employee(1, "Kaneki", "М", 20, "fgghd@mail.ru", "программист", 2, kid=None, car=None)

employee\_2 = Employee(2, "Kanekii", "М", 20, "fgghd@mail.ru", "программист", 2, kid=None, car=None)

assert (employee\_1 < employee\_2) == False

def test\_wrong\_type\_Department():

with pytest.raises(TypeError):

employee\_1 = Employee(1, "Kaneki", "М", 20, "fgghd@mail.ru", "программист", 2, kid=None, car=None)

dep\_1 = Departments("Закупок", "Смирнов И.В.")

dep\_1.append("Петров А.Н.")

def test\_append\_Department():

with pytest.raises(TypeError):

employee\_1 = Employee(1, "Kaneki", "М", 20, "fgghd@mail.ru", "программист", 2, kid=None, car=None)

dep\_1 = Departments("Закупок", "Смирнов И.В.")

count\_before = len(dep\_1.employees)

dep\_1.append(employee\_1)

assert count\_before == len(dep\_1.employees) - 1

def test\_wrong\_kids\_age():

with pytest.raises(ValueError):

kid\_1 = Kids("Kaneki", -15, "М")

def test\_increase\_age():

Kids\_1 = Kids("Семенов М.А.",15 ,"12.12.2000")

Kids\_1.increase\_age(2)

assert Kids\_1.age == 15

def test\_increase\_car\_price():

Car\_1 = Car("bmx" ,"x3" ,"22fg589", 4000000)

Car\_1.increase\_price(200000)

assert Car\_1.price == 4000000

def test\_kids\_lt():

Kids\_1 = Kids("Семенов М.А.", 16, "М")

Kids\_2 = Kids("Семенова М.М.",14, "М")

assert (Kids\_1 < Kids\_2) == False

def test\_Kids\_lt():

Car\_1 = Car("bmx" ,"x4" ,"22fg589", 4000000)

Car\_2 = Car("bmxx" ,"x4" ,"22ffd89", 12000000)

assert (Car\_1 < Car\_2) == False