

Колледж космического машиностроения и технологий

ОТЧЕТ

По учебной практике УП.01.01 Разработка программных модулей

программного обеспечения для компьютерных систем

специальность 09.02.03 Программирование в компьютерных системах

Выполнили студенты:

Степаненко К.А

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Гусятинер Л. Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2021

**Содержание**

[**Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования.** 2](#_Toc69926563)

[1.1 Установка интерпретатора Python 3 и настройка окружения 2](#_Toc69926564)

[1.2 Техника работы в командной строке и среде IDLE 5](#_Toc69926565)

[1.3 Техника работы с линейными и разветвляющимися программами 7](#_Toc69926566)

[1.4 Техника работы с циклическими программами, цикл while 16](#_Toc69926567)

[1.5 Техника работы с числами 24](#_Toc69926568)

[1.6 Техника работы со строками 31](#_Toc69926569)

[1.7 Техника работы со списками 36](#_Toc69926570)

[1.8 Техника работы с циклом for и генераторами списков 41](#_Toc69926571)

[1.9 Техника работы с функциями 48](#_Toc69926572)

[1.10 Техника работы со словарями 54](#_Toc69926573)

[1.11 Техника работы с множествами 64](#_Toc69926574)

[1.12 Техника работы с кортежами 72](#_Toc69926575)

[1.13 Техника работы с файлами 78](#_Toc69926576)

[1.14 Техника работы с модулями 86](#_Toc69926577)

[1.15 Техника работы с классами 102](#_Toc69926578)

[**Раздел 2. Техника решения задач с использованием библиотек** 113](#_Toc69926579)

[2.1 Установка и настройка среды JetBrains PyCharm 113](#_Toc69926580)

[2.2 Техника работы с базами данных 119](#_Toc69926581)

[2.3 Техника работы с библиотекой tkinter 123](#_Toc69926582)

[2.4 Техника работы с библиотекой NumPy 126](#_Toc69926583)

[2.5 Техника работы с библиотекой Matplotlib 128](#_Toc69926584)

[2.6 Элементы работы с библиотекой PyQt 130](#_Toc69926585)

[2.7 Элементы работы с библиотекой PyGame 132](#_Toc69926586)

[**Раздел 3. Разработка проекта с графическим интерфейсом** 146](#_Toc69926587)

[3.1 Изучение входной и выходной документации 146](#_Toc69926588)

[3.2 Разработка требований к проекту. Построение диаграммы использования. 148](#_Toc69926589)

[3.3 Разработка сценария проекта 150](#_Toc69926590)

[3.4 Построение диаграммы классов 150](#_Toc69926591)

[3.5 Разработка базы данных 151](#_Toc69926592)

[3.6 Разработка главного модуля 155](#_Toc69926593)

[3.7 Разработка входящих модулей 156](#_Toc69926594)

[3.8 Тестирование и отладка 162](#_Toc69926595)

[3.9 Дневник 163](#_Toc69926596)

# **Техника решения задач с использованием библиотек**

## Установка и настройка среды JetBrains PyCharm

PyCharm - это кросс-платформенная среда разработки.

Переходим на страницу скачивания PyCharm <https://www.jetbrains.com/ru-ru/pycharm/>. Для скачивания доступно две версии: профессиональная и версия для сообщества. Версия для сообщества бесплатная. Ее и скачаем.

Запускаем скачанный .exe файл. В первом окне нас приветствует сам установщик. Смело кликаем «Next»:



Рисунок 108. Установщик PyCharm



Рисунок 109. Место установки PyCharm



Рисунок 110. Процесс установки PyCharm



Рисунок 111. Стартовое окно PyCharm

В следующем окне будет два поля. В первом укажите, где будет располагаться новый проект. Второе поле должно быть заполненным помолчанию. Там содержится путь к установленному ранее Python 

Рисунок 112. Выбор места для нового проекта и выбор интерпретатора

Дальше откроется окно самого редактора.

Теперь вы готовы начать писать программы на Python!

Кликните правой кнопкой на название вашего проекта и в раскрывающемся списке выберите пункт «New > Python file»: 

Рисунок 113. Создание нового Phython файла

Появится окно, в котором вы можете задать имя файлу. Задайте какое-нибудь имя и нажмите «OK».

Справа откроется сам файл. Пока он пустой. Пишем:



Рисунок 114. Написание программы

Теперь нужно запустить нашу небольшую (1 строка) программу. Для этого в верхнем меню перейдите в «Run > Run…».

﷟HYPERLINK "https://neuralnet.info/wp-content/uploads/2017/09/PyCharm\_4.png"

Рисунок 115. Запуск программы

В появившемся окне щелкните по названию вашего файла. Теперь программа запустится. В нижней части редактора должна появиться консоль с результатом



Рисунок 116. Выполнение программы

# Техника работы с базами данных

Приложения: sql\_bd.py

Python SQLite поставляется вместе с любой установленной версией Python, поэтому нет необходимости устанавливать его с помощью pip. Чтобы импортировать его, поскольку мы будем использовать Python3, мы импортируем модуль sqlite3.

Листинг 77. sql\_bd.py

import sqlite3  
  
  
class Sqliter:  
  
 *# Функция \_\_init\_\_ служит инициализацией БД* def \_\_init\_\_(self, name\_db):  
 self.connection = sqlite3.connect(name\_db)  
 self.cursor = self.connection.cursor()  
  
 *# Функция create\_table создаёт таблицу в нашей БД* def create\_table(self, table):  
 with self.connection:  
 table = ''.join(chr for chr in table if chr.isalnum())  
 if table != '':  
 self.cursor.execute(f"CREATE TABLE IF NOT EXISTS `{table}` (\  
 id integer PRIMARY KEY,\  
 `fio` text, `group` text, `direction` text)")  
 self.save()  
  
 *# Функция add\_student добавляет студента в БД, если его не существует* def add\_student(self, \*\*kwargs):  
 with self.connection:  
 data = kwargs  
 if kwargs.get('data') != None:  
 data = kwargs['data']  
 self.cursor.execute("INSERT INTO `students`\  
 (`fio`, `group`, `direction`) VALUES (?, ?, ?)",  
 (data['fio'], data['group'], data['direction']))  
 self.save()  
  
 *# Функция get\_id возвращает id записи в таблице* def get\_id(self, \*\*kwargs):  
 with self.connection:  
 try:  
 data = kwargs  
 if kwargs.get('data') != None:  
 data = kwargs.get('data')  
 return self.cursor.execute("SELECT \* FROM `students` WHERE \  
 `fio` = ? AND `group` = ? AND `direction` = ?",  
 (data['fio'], data['group'], data['direction'])).fetchall()[0][0]  
 except:  
 return -1  
  
 *# Функция save сохраняет изменения в БД* def save(self):  
 self.connection.commit()  
 print(f"{self.cursor.rowcount} отредактированно строк")  
  
 *# Функция close закрывает БД* def close(self):  
 self.connection.close()  
  
*# Функция input\_student возвращает данные о студенте в нужном для БД формате*def input\_student(\*\*kwargs):  
 data = kwargs  
 if kwargs.get('data') != None:  
 data = kwargs.get('data')  
 student = {  
 'fio': data['fio'],  
 'group': data['group'],  
 'direction': data['direction']  
 }  
 return student  
  
*# Создание БД*bd = Sqliter("bd")  
*# Создание таблицы students*bd.create\_table('students')  
  
  
student = input\_student(fio="Cipkov Il'ya Vladimirovich",\  
 group="P1-18", direction="Programmer")  
*# Если студента не находит в БД, то мы его добавляем в БД*if bd.get\_id(data=student) == -1:  
 bd.add\_student(data=student)  
*# Если такой студент существует, то в консоль выводит "Ne mogu("*else:  
 print('Ne mogu(')  
  
*# Прекращение работы с БД*bd.close()

После завершения кода в той же папке где находится python-file появится файл с БД и в ней должна появиться запись «*1 Cipkov Il'ya Vladimirovich P1-18 Programmer*». Просто так его открыть и посмотреть нельзя.

Для того чтобы увидеть БД, необходимо будет скачать DB Browser (SQLite). Переходим на страницу скачивания программы <https://sqlitebrowser.org/dl/>.



Рис. 117. Установщик DB Browser

После установки, открываем DB Browser для работы с ним.

Рис. 118. Окно DB Browser

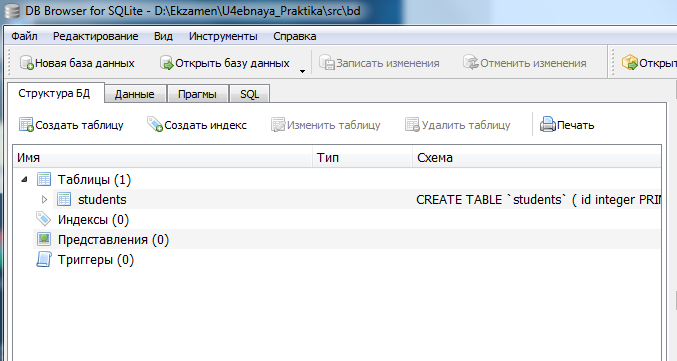
В открывшееся окно DB Browser’a перетаскиваем файл с БД.

Рис. 119. Таблица students в DB Browser

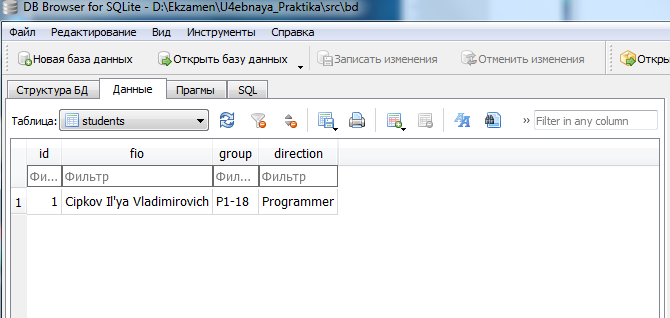
Чтобы просмотреть содержимое БД необходимо нажать на кнопку «Данные».

Рис. 120. Данные таблицы students

Видно, что в таблицу занеслись корректные данные.

# Техника работы с библиотекой tkinter

Приложения: MyTable.py

Библиотека Tkinter установлена в Python в качестве стандартного модуля, поэтому нам не нужно устанавливать что-либо для его использования. Чтобы импортировать его, пропишем import tkinter. В качестве примера для демонстрации tkinter будет приведена программа, которая принимает данные из формы и добавляет их в текстовый файл, также в программе есть возможность очищать весь файл полностью.

Листинг 78: MyTable.py

from tkinter import \*  
  
def main():  
 *#Функция save сохраняет введённые данные в строку и записывает строчку в файл* def save():  
 str = get\_name() + ' ' + group\_get() + ' ' + show\_parametrs() + m\_g()  
 with open("test.txt", "a") as f:  
 f.write(str + '\n')  
  
 *# Функция group\_get возвращает выбранную группу пользователем* def group\_get():  
 return my\_var.get()  
  
 *# Функция show\_parametrs возвращает все языки, которые выбрал пользователь* def show\_parametrs():  
 str = ""  
 for name, var in parametrs:  
 if var.get():  
 str += name + ' '  
 return str  
  
 *# Функция get\_name возвращает имя введённое пользователем* def get\_name():  
 return my\_var2.get()  
  
 *# Функция delete очищает текстовоый файл полностью* def delete():  
 f = open('test.txt', 'w+')  
 f.seek(0)  
 f.close()  
  
 *# Функция m\_g возвращает выбранный пол пользователем* def m\_g():  
 if v.get() == 1:  
 return "Мужской"  
 else:  
 return "Женский"  
  
 *# Создание объекта окна верхнего уровня (на нём будут распологаться все элементы* root = Tk()  
 *# Задание размера окна и его координат расположения* root.geometry("300x300+700+300")  
 *# StringVar используется для легкого отслеживания изменений в переменных tkinter* my\_var = StringVar('')  
 my\_var2 = StringVar('')  
 parametrs = (("Java", IntVar()), ("C#", IntVar()), ("C++", IntVar()),\  
 ("Python", IntVar()), ("Pascal", IntVar()))

*# Заполнение и задание расположения CheckBox'ов* x, y = 2, 0  
 for name, var in parametrs:  
 Checkbutton(root, text=f"{name}", variable=var).grid(row=x, column=y)  
 x += 1  
  
 *# Задаём местоположение текстовой метке и полю для ввода* lb1 = Label(text="Имя:").grid(row=0, column=0)  
 en = Entry(width=30, textvariable=my\_var2).grid(row=0, column=1, columnspan=3)  
  
 *# Задаём местоположение текстовой метке и SpinBox'у(в котором находится список групп)* lb2 = Label(text="Группа:").grid(row=1, column=0, columnspan=1)  
 mas = ('П1-18', 'П2-18', 'П1-19', 'П2-19', 'П3-19')  
 sb = Spinbox(width=7, values=mas, textvariable=my\_var,\  
 command=group\_get).grid(row=1, column=1)

*# Задаём местоположение кнопкам Save и Delete* b = Button(root, text="Save", width=25, height=2, command=save).grid(row=9, column=1, columnspan=2)  
 b1 = Button(root, text="Delete", width=25, height=2,\  
 command=delete).grid(row=10, column=1, columnspan=2)  
  
 *# Создаём и задаём местоположение текстовой метке и RadioButton'ам* v = IntVar()  
 lb3 = Label(text="Пол:").grid(row=8, column=0)  
 rb1= Radiobutton(root, text="Мужской", value=1,\  
 variable=v).grid(row=8, column=1, columnspan=1)  
 rb2 = Radiobutton(root, text="Женский", value=2,\  
 variable=v).grid(row=8, column=2, columnspan=1)  
  
 *# Запуск цикла обработки событий* mainloop()  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Форма имеет вид:

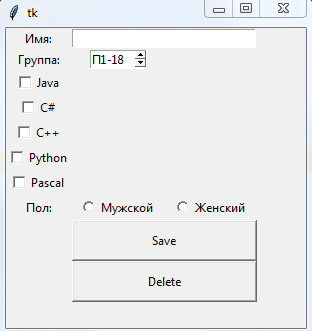


Рис. 121. Вид формы

Заполним форму данными:

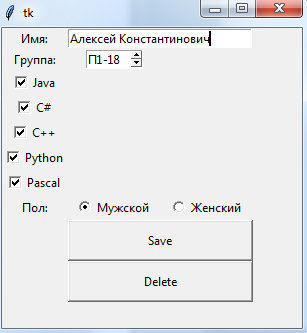


Рис. 122. Заполненная форма

Нажмём кнопку “Save”. В текущем каталоге с python-файлом появится текстовик, в котором и сохранятся наши данные.

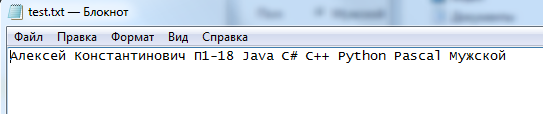


Рис. 123. Вывод в файл

# Техника работы с библиотекой NumPy

Приложения: primer.py

NumPy это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций.

Установка:

• Через терминал командой: “pip install numpy”

• По инструкции на сайте <https://www.scipy.org/install.html>

Импорт модуля numpy:

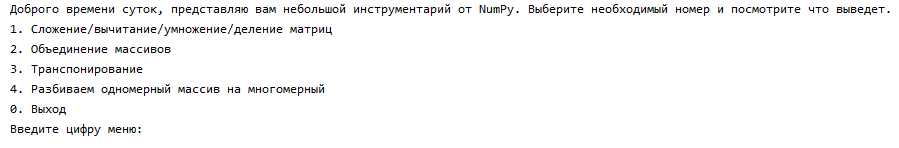
import numpy

Главной особенностью numpy является объект array. Массивы схожи со списками в python, исключая тот факт, что элементы массива должны иметь одинаковый тип данных, как float и int. С массивами можно проводить числовые операции с большим объемом информации в разы быстрее и, главное, намного эффективнее чем со списками.

Листинг 79: primer.py

import numpy  
  
  
def main():  
 *# Бесконечно выводим меню в консоль, пока пользователь не введёт 0* while(True):  
 menu()  
 print("Введите цифру меню: ")  
 num = int(input())  
 if(num == 1):  
 print("Введите функцию(-, +, \*, /):")  
 s = input()  
 function\_1(s)  
 print("\n")  
 elif(num == 2):  
 function\_2()  
 print("\n")  
 elif(num == 3):  
 function\_3()  
 print("\n")  
 elif(num == 4):  
 function\_4()  
 print("\n")  
 elif(num == 0):  
 *# Выход из программы* exit(0)  
 else:  
 print("Вы ввели не корректный номер функции. Попробуйте ещё раз\n")  
  
  
*# Функция menu обеспечивает вывод меню в консоль*def menu():  
 print("Доброго времени суток, представляю вам небольшой\  
 инструментарий от NumPy. Выберите необходимый номер и посмотрите что выведет.")  
 print("1. Сложение/вычитание/умножение/деление матриц")  
 print("2. Объединение массивов")  
 print("3. Транспонирование")  
 print("4. Разбиваем одномерный массив на многомерный")  
 print("0. Выход")  
  
*# Функция function\_1 выводит в консоль пример сложения/вычитания/деления/умножения матриц*def function\_1(s):  
 A = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
 B = A.copy()[::-1, ::-1]  
 if(s == "+"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A + B\n", A + B)  
 elif(s == "-"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A - B\n", A - B)  
 elif(s == "\*"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A \* B\n", A \* B)  
 elif(s == "/"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A / B\n", A // B)  
  
*# Функция function\_2 выводит в консоль пример объединения двух матриц в одну*def function\_2():  
 A = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
 B = A[::-1]  
 C = A[:, ::-1]  
 print("A\n", A)  
 print("\nB\n", B)  
 print("\nC\n", C)  
 print("\nstack(A, B, C)\n", numpy.stack((A, B, C)))  
  
*# Функция function\_3 выводит в консоль пример транспонирования матрицы*def function\_3():  
 A = numpy.array([[1, 2, 3], [4, 5, 6]])  
 print('A\n', A)  
  
 B = A.T  
 print('\nА транспонированный\n', B)  
  
*# Функция function\_4 выводит в консоль пример разбиения одномерного массива на многомерный*def function\_4():  
 A = numpy.arange(24)  
 B = A.reshape(4, 6)  
 C = A.reshape(4, 3, 2)  
 print("Массив размером 1х1\n", A)  
 print('\nМассив размером 4х6\n', B)  
 print('\nМассив размером 4х3х2\n', C)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':

main()

Рис. 124. Вывод меню в консоль

# Техника работы с библиотекой Matplotlib

Приложения: mat.py, mat2.py

Библиотека matplotlib в Python помогает нам отображать данные на графиках в простейшем виде.

Библиотеку matplotlib легко установить с помощью pip:

pip install matplotlib

Теперь мы готовы создать несколько примеров, используя эту библиотеку визуализации данных.

Листинг 80: mat.py

import matplotlib.pyplot as plt  
*# Координаты точек по x*x = [1, 5, 10, 15, 20]  
*# Координаты точек графика first по оси y*y1 = [1, 7, 3, 5, 11]  
*# Координаты точек графика second по оси y*y2 = [4, 3, 1, 8, 12]  
plt.figure(figsize=(7, 4)) *# Размер рисунка  
# Построение графика first*plt.plot(x, y1, 'o-r', alpha=0.7, label="first", lw=5, mec='b', mew=2, ms=10)  
*# Построение графика second*plt.plot(x, y2, 'v-.g', label="second", mec='r', lw=2, mew=2, ms=12)  
plt.legend() *# Отображение легенды*plt.grid(True) *# Отображение сетки*plt.show() *# Отображение графика*

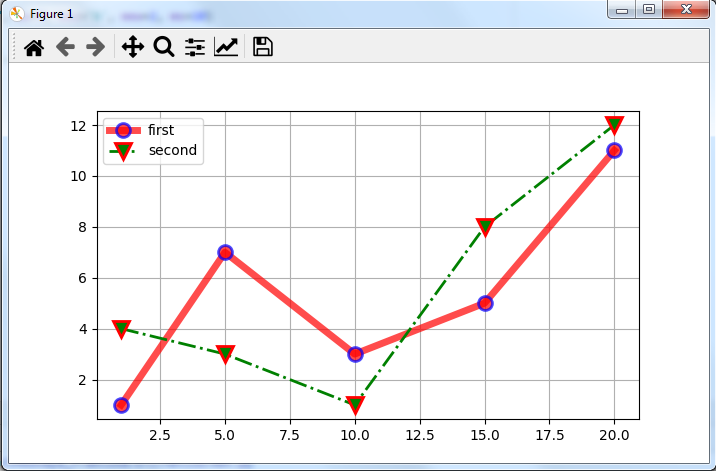


Рис. 125. Вывод программы

Листинг 81: mat2.py

import matplotlib.pyplot as plt  
import numpy as np  
*# Установка столбцов по значениям*p = [f"P{i}" for i in range(5)]  
*# Значения первой группы столбцов*g1 = [10, 21, 34, 12, 27]  
*# Значения второй группы столбцов*g2 = [17, 15, 25, 21, 26]  
width = 0.3 *# Ширина столбцов  
# [0, 1, 2, 3, 4]*x = np.arange(len(p))  
*# Размещение графиков рядом друг с другом*fig, ax = plt.subplots()  
rects1 = ax.bar(x - width/2, g1, width, label='g1')  
rects2 = ax.bar(x + width/2, g2, width, label='g2')  
ax.set\_title('Пример групповой диаграммы')  
*# Список местоположений тиков по оси x*ax.set\_xticks(x)  
*# Возвращает список текстовых экземпляров*ax.set\_xticklabels(p)  
*# Отображение легенды*ax.legend()  
plt.show()

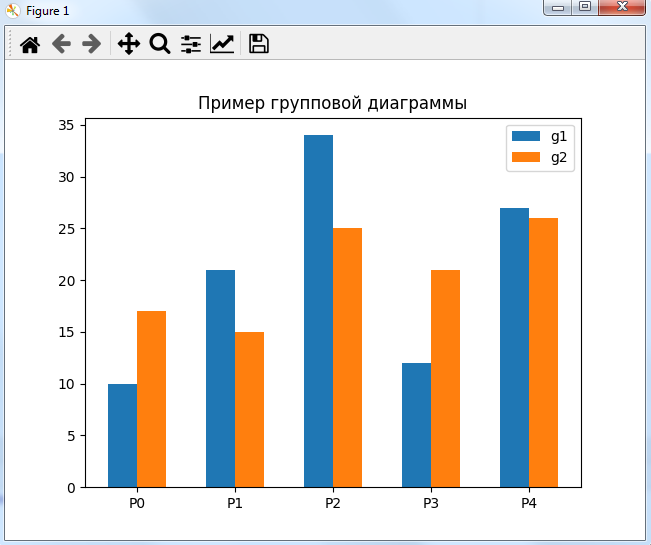


Рис. 126. Вывод программы

# Элементы работы с библиотекой PyQt

Приложения: MyQt.py

PyQt — набор расширений [графического](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [фреймворка](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) [Qt](https://ru.wikipedia.org/wiki/Qt" \o "Qt) для [языка программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [Python](https://ru.wikipedia.org/wiki/Python" \o "Python), выполненный в виде [расширения](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD) Python. Это мульти-платформенный инструментарий, который запускается на большинстве операционных систем, среди которых Unix, Windows и MacOS.

Для установки библиотеки используется команда:

pip install PyQt5

Листинг 82: MyQt.py

from PyQt5.QtWidgets import \*  
from PyQt5.QtCore import QSize  
import random  
  
  
  
*# Наследуемся от QMainWindow*class MainWindow(QMainWindow):  
 *# Создаём диалоговое окно, которое открывается при нажатии на кнопку* def dialog(self, lineEdit, strList, secretList):  
 mbox = QMessageBox() *# Создаём диалоговое окно* mbox.setWindowTitle('Окно')  
 *# Условия вывода определённого текста в окне* if(lineEdit.text() in strList):  
 mbox.setText(lineEdit.text() + " krut!")  
 elif(lineEdit.text() in secretList):  
 mbox.setText(lineEdit.text() + "\nЭто правда")  
 else:  
 mbox.setText("prosto text")  
 mbox.setDetailedText("Наберите одно из этих слов\n'Python', 'PyQt5', 'Qt', 'Django', 'QML'")  
 mbox.exec()  
  
 *# Переопределяем конструктор класса* def \_\_init\_\_(self):  
 *# Обязательно нужно вызвать метод супер класса* QMainWindow.\_\_init\_\_(self)  
  
 self.setMinimumSize(QSize(480, 80)) *# Устанавливаем размеры* self.setWindowTitle("Проверка автодополнения") *# Устанавливаем заголовок окна* central\_widget = QWidget(self) *# Создаём центральный виджет* self.setCentralWidget(central\_widget) *# Устанавливаем центральный виджет* grid\_layout = QGridLayout() *# Создаём QGridLayout* central\_widget.setLayout(grid\_layout) *# Устанавливаем данное размещение в центральный виджет* grid\_layout.addWidget(QLabel("Проверка автодополнения", self), 0, 0)  
  
 *# Создаём поле ввода* lineEdit = QLineEdit(self)  
 strList = ['Python', 'PyQt5', 'Qt', 'Django', 'QML'] *# Создаём список слов* secretList = ['Хочу домой', 'Я сделал домашку', 'ККМТ - сила'] *# Создаём секретный список* rand = random.randint(1,10)  
 *# Создаём QCompleter, в который устанавливаем список, а также указатель на родителя  
 # С вероятностью 50% в QCompleter может добавиться секретный список* if(rand % 2 == 0):  
 completer = QCompleter(strList, lineEdit)  
 else:  
 completer = QCompleter(strList + secretList, lineEdit)  
 lineEdit.setCompleter(completer) *# Устанавливает QCompleter в поле ввода* grid\_layout.addWidget(lineEdit, 0, 1) *# Добавляем поле ввода в сетку* qb = QPushButton("Submit") *# Создаём кнопку  
 # Создаём действие для кнопки* qb.clicked.connect(lambda : self.dialog(lineEdit, strList, secretList))  
 grid\_layout.addWidget(qb, 0, 2)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
  
 app = QApplication(sys.argv)  
 mw = MainWindow()  
 mw.show()  
 sys.exit(app.exec())

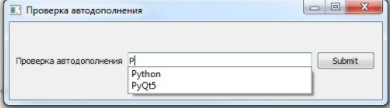


Рис. 127. Окно проверки автодополнения

# Элементы работы с библиотекой PyGame

Приложения: main.py

Pygame – это библиотека модулей для языка Python, созданная для разработки 2D игр.

Pygame не входит в стандартную библиотеку Python, то есть не поставляется с установочным пакетом, а требует отдельной установки. В Ubuntu и родственных дистрибутивах это можно сделать двумя способами – с помощью pip и apt-get:

python3 -m pip install -U pygame --user

или

sudo apt-get install python3-pygame

Для Windows:

Открыв командную строку (cmd) прописываем:

pip install pygame

Убедиться, что библиотека загружена, и что все установилось нормально, можно опять же в командной строке (cmd). Следует вписать следующие две строчки:

Python

Import pygame

Листинг 83. main.py

import pygame

import random

class Hero:

"""Класс Hero. Здесь описана вся работа с героем"""

positions = list() # Координаты героеа

# Цвета для героев

colors = {

'RED': (237, 77, 12),

'PURPLE': (130, 12, 247),

'TURQUOISE': (22, 224, 214),

'YELLOW': (213, 247, 12),

'ORANGE': (240, 121, 40)

}

def \_\_init\_\_(self, size, type\_hero):

"""Создание героя"""

if type\_hero == 'circle' or type\_hero == 'square':

self.width = size[0]

self.height = size[0]

else:

self.width = size[0]

self.height = size[1]

self.type\_hero = type\_hero

self.bgc = (125, 125, 125)

self.x = 100

self.y = 100

self.speed = 1

self.target = {'y': '', 'x': ''}

self.\_\_class\_\_.positions.append([self.x, self.y])

self.index = len(self.\_\_class\_\_.positions)-1

#def \_\_repr\_\_(self):

# return f"{\_\_class\_\_.\_\_name\_\_ }(({self.width}x{self.height}), ({self.x}, {self.y}))"

def update\_positions(self):

"""Обновляем позиции героя в общеи списке позиций"""

self.positions[self.index][0] = self.x

self.positions[self.index][1] = self.y

def get\_all\_positions(self, pos):

"""Вернет все позиции 'x' или 'y' """

if pos == 'x':

return [pos[0] for pos in self.positions]

else:

return [pos[1] for pos in self.positions]

def set\_map(self, x\_start, x\_end, y\_start, y\_end):

"""Создание карты для героя"""

# Устанавливаем границы

if self.type\_hero == 'circle':

x\_start = x\_start + self.width

x\_end = x\_end - self.width

y\_start = y\_start + self.width

y\_end = y\_end - self.width

elif self.type\_hero == 'rect' or self.type\_hero == 'square':

x\_end = x\_end - self.width

y\_end = y\_end - self.height

self.mapp = {

'x\_start':x\_start,

'x\_end': x\_end,

'y\_start': y\_start,

'y\_end': y\_end

}

def set\_color(self, \*\*kwargs):

"""Установление цвета герою"""

if kwargs.get('hex') != None:

self.bgc = kwargs['hex']

elif kwargs.get('color') != None:

self.bgc = self.colors.get(kwargs['color'], (125, 125, 125))

else:

self.bgc = random.choice( list(self.colors.values()) )

def set\_type(self, type\_hero):

"""Установление типа герою. (круг, квадрат, прямоугольник)"""

if type\_hero == 'circle' or type\_hero == 'square':

if self.width > self.height:

self.height = self.width

else:

self.width = self.height

self.type\_hero = type\_hero

self.set\_map(\

self.mapp['x\_start'],

self.mapp['x\_end'],

self.mapp['y\_start'],

self.mapp['y\_end'],

)

def set\_speed(self, speed):

"""Установление скорости герою"""

self.speed = speed

def add\_speed(self, speed):

"""Повышние скорости героя"""

if speed < 0:

self.sub\_speed(speed)

else:

self.speed += speed

print("speed: ", self.speed)

def sub\_speed(self, speed):

"""Понижение скорости героя"""

if speed < 0:

self.add\_speed(speed)

else:

self.speed -= speed

if self.speed < 0:

self.speed = 0

print("speed: ", self.speed)

def set\_pos(self, \*args):

"""Установление позиций героя"""

if args:

self.x = args[0]

self.y = args[1]

else:

self.x = random.randrange(self.mapp['x\_start'], self.mapp['x\_end'])

self.y = random.randrange(self.mapp['y\_start'], self.mapp['y\_end'])

self.update\_positions()

def draw(self, display):

"""Отрисовка героя"""

if self.type\_hero == 'circle':

pygame.draw.circle(display, self.bgc, (self.x, self.y), self.width)

elif self.type\_hero == 'rect':

pygame.draw.rect(display, self.bgc, (self.x, self.y, self.width, self.height))

elif self.type\_hero == 'square':

pygame.draw.rect(display, self.bgc, (self.x, self.y, self.width, self.width))

def set\_target(self, \*args):

"""Установление стороны в которую движется герой"""

if not args:

self.target['x'] = random.choice(['left', 'right'])

self.target['y'] = random.choice(['top', 'bottom'])

elif args[0] == 'left' or args[0] == 'right':

self.target['x'] = args[0]

elif args[1] == 'top' or args[1] == 'bottom':

self.target['y'] = args[1]

print(self.target)

def navigation(self):

"""Передвижение героя"""

if self.target['x'] == 'left':

self.x -= self.speed

elif self.target['x'] == 'right':

self.x += self.speed

if self.target['y'] == 'bottom':

self.y += self.speed

elif self.target['y'] == 'top':

self.y -= self.speed

# Уперся в другой объект

is\_altered = False

for pos in self.positions:

if (pos[0] == self.x) and (pos[1] == self.y):

is\_altered = True

if self.target['x'] != '':

if self.target['x'] == 'left':

self.target['x'] = 'right'

elif self.target['x'] == 'right':

self.target['x'] = 'left'

if self.target['y'] != '':

if self.target['y'] == 'top':

self.target['y'] = 'bottom'

elif self.target['y'] == 'bottom':

self.target['y'] = 'top'

# Уперся в границу

if not is\_altered:

if (self.x <= self.mapp['x\_start']):

self.target['x'] = 'right'

elif (self.x > self.mapp['x\_end']):

self.target['x'] = 'left'

if (self.y <= self.mapp['y\_start']):

self.target['y'] = 'bottom'

elif (self.y >= self.mapp['y\_end']):

self.target['y'] = 'top'

self.update\_positions()

# Функционал игры

def Update(display, map\_size, heroes):

global stop, mode

# Какая операция со скоростью.

# 1 - повысить скорость

# 0 - ничего не делать

# -1 - понизить скорость

operating\_speed = 0

# Какую кнопку нажали

for event in pygame.event.get():

# Выход

if event.type == pygame.QUIT:

pygame.quit()

quit()

# Нажатие кнопки = зажатию

elif mode == 'press':

if event.type == pygame.KEYDOWN:

# Приостановление игры

if event.key == pygame.K\_ESCAPE:

stop = not stop

print(stop)

if not stop:

# Смена мода

if event.key == pygame.K\_TAB:

mode = 'touch'

# Нажатие кнопки = нажатию

elif mode == 'touch':

if event.type == pygame.KEYDOWN:

# Приостановление игры

if event.key == pygame.K\_ESCAPE:

stop = not stop

print(stop)

if not stop:

# Создание нового героя

if event.key == pygame.K\_SPACE and not stop:

hero = Hero((25,), 'circle')

# hero = Hero((50,), 'square')

hero.set\_map( map\_size["x\_start"], map\_size["x\_end"], map\_size["y\_start"], map\_size["y\_end"] )

hero.set\_speed(1)

hero.set\_pos()

hero.set\_target()

# hero.set\_pos(50, 100)

# hero.set\_target('', '')

hero.set\_color()

heroes.append(hero)

del hero

# статус скорости = Повышение

elif event.key == pygame.K\_UP:

operating\_speed = 1

# # статус скорости = Понижение

elif event.key == pygame.K\_DOWN:

operating\_speed = -1

# Смена мода

elif event.key == pygame.K\_TAB:

mode = 'press'

# Отрисовка фона

display.fill(bgc)

# Если игра не пиостановлена

if not stop:

if mode == 'press':

keys = pygame.key.get\_pressed()

# Создание новых персонажей

if keys[pygame.K\_SPACE]:

hero = Hero((25,), 'circle')

# hero = Hero((50,), 'square')

hero.set\_map( map\_size["x\_start"], map\_size["x\_end"], map\_size["y\_start"], map\_size["y\_end"] )

hero.set\_speed(1)

hero.set\_pos()

hero.set\_target()

# hero.set\_pos(50, 100)

# hero.set\_target('', '')

hero.set\_color()

heroes.append(hero)

del hero

# статус скорости = Повышение

elif keys[pygame.K\_UP]:

operating\_speed = 1

# статус скорости = Понижение

elif keys[pygame.K\_DOWN]:

operating\_speed = -1

# Относиельно статуса скорости повышаем/понижаем скорость

for hero in heroes:

if operating\_speed == 1:

hero.add\_speed(1)

elif operating\_speed == -1:

hero.sub\_speed(1)

hero.navigation()

hero.draw(display)

# Возращаем нейтральный статус для скорости

if operating\_speed != 0:

operating\_speed = 0

# Обновляем окно

pygame.display.update()

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

# Инициализация окна

pygame.init()

# Данные для окна

display\_size = {"width": 800, "height": 600}

display = pygame.display.set\_mode((display\_size['width'], display\_size['height']))

pygame.display.set\_caption("Bruh")

# bgc = (255, 255, 255)

bgc = (40, 40, 40)

# Размер карты

map\_size = {

"x\_start": 0,

"x\_end": display\_size['width'],

"y\_start": 0,

"y\_end": display\_size['height'],

}

# Создание героя (шарика). Кладем его в спсок со всеми героями

heroes = list()

hero = Hero((25,), 'circle')

# hero = Hero((50,), 'square')

hero.set\_map( map\_size["x\_start"], map\_size["x\_end"], map\_size["y\_start"], map\_size["y\_end"] )

hero.set\_speed(1)

hero.set\_target()

hero.set\_pos()

hero.set\_color()

heroes.append(hero)

del hero

game = True # Запушена ли игра

stop = False # Приостановлена ли игра

mode = "touch" # Мод. Либо 'touch', либо 'press'

# Запуск игры

while game:

Update(display, map\_size, heroes)

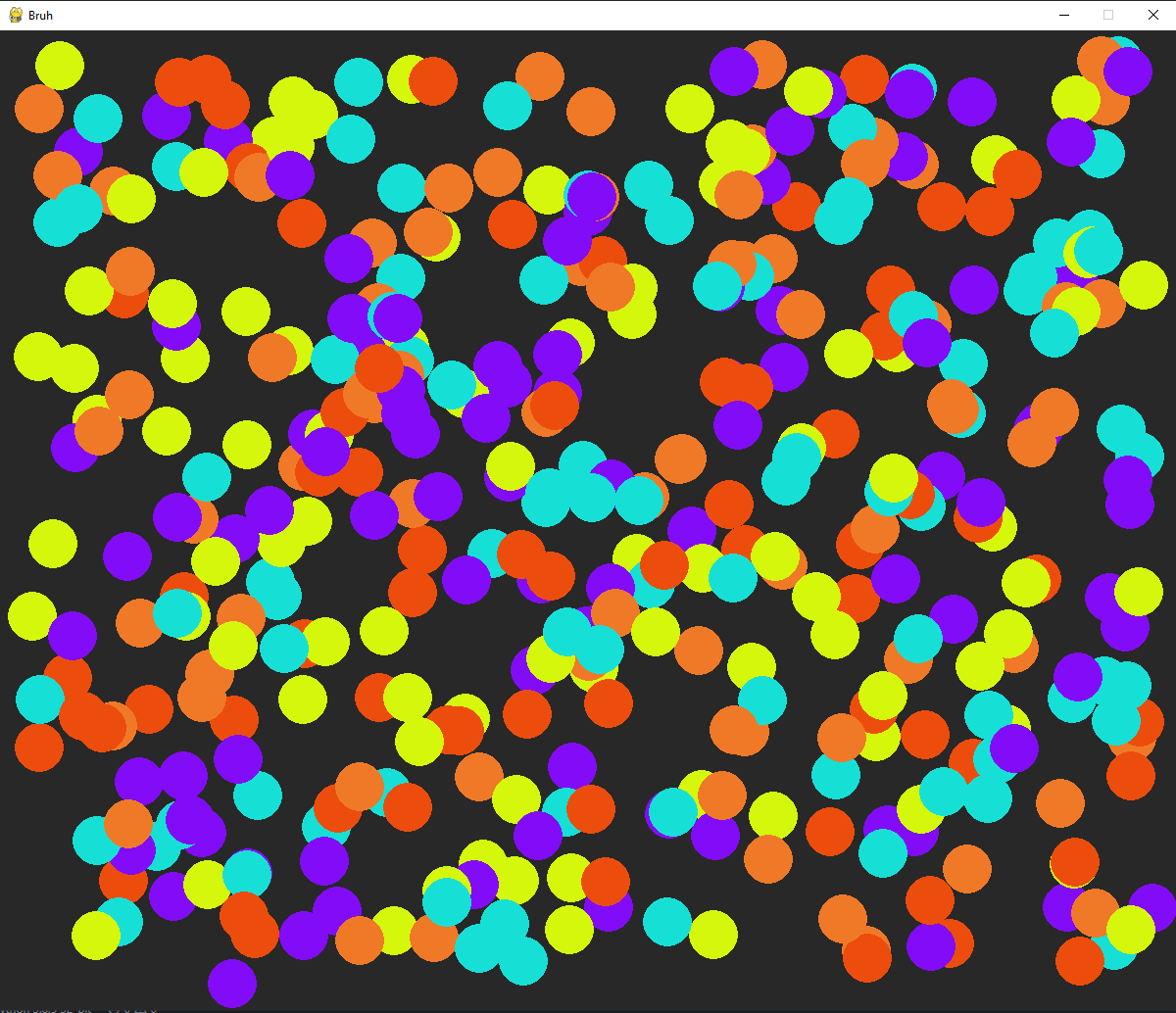


Рисунок 128. Скриншот игры

* 1. Дневник

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 10.12.2020 | Техника работы с линейными программами. Техника работы с разветвляющимися программами. Техника работы с циклическими программами. |  |
| 11.12.2020 | Техника работы с циклическими программами. Техника работы с числами. |  |
| 12.12.2020 | Техника работы со строками. Техника работы со списками. |  |
| 14.12.2020 | Техника работы со списками. Техника работы с циклом for и генераторами списков. |  |
| 15.12.2020 | Техника работы с функциями. Техника работы со словарями. |  |
| 16.12.2020 | Техника работы со словарями. Техника работы с множествами. |  |
| 17.12.2020 | Техника работы с кортежами. Техника работы с файлами. |  |
| 18.12.2020 | Техника работы с файлами. Техника работы с модулями. |  |
| 19.12.2020 | Техника работы с модулями. Техника работы с классами. |  |
| 21.12.2020 | Техника работы с классами. |  |
| 30.03.2021 | Установка и настройка среды JetBrains PyCharm. Техника работы с базами данных. |  |
| 31.03.2021 | Техника работы с базами данных. Техника работы с библиотекой tkinter. |  |
| 01.04.2021 | Техника работы с библиотекой tkinter |  |
| 02.04.2021 | Техника работы с библиотекой tkinter. Техника работы с библиотекой NumPy. |  |
| 05.04.2021 | Техника работы с библиотекой NumPy. Техника работы с библиотекой Matplotlib. |  |
| 06.04.2021 | Техника работы с библиотекой Matplotlib. Элементы работы с библиотекой PyQt. |  |
| 07.04.2021 | Элементы работы с библиотекой PyQt. |  |
| 08.04.2021 | Элементы работы с библиотекой PyQt. Элементы работы с библиотекой PyGame. |  |
| 09.04.2021 | Элементы работы с библиотекой PyGame. Выполнение отчёта и презентации по использованию библиотек. |  |
| 12.04.2021 | Выполнение отчёта и презентации по использованию библиотек. |  |
| 13.04.2021 | Изучение входной и выходной документации. Разработка требований к проекту. Построение диаграммы использования |  |
| 14.04.2021 | Разработка требований к проекту. Построение диаграммы использования. Разработка сценария проекта. |  |
| 15.04.2021 | Разработка сценария проекта. Построение диаграммы классов. |  |
| 16.04.2021 | Построение диаграммы классов. Разработка базы данных. |  |
| 19.04.2021 | Разработка базы данных. Разработка главного модуля. |  |
| 20.04.2021 | Разработка главного модуля. Разработка входящих модулей. |  |
| 21.04.2021 | Разработка входящих модулей. |  |
| 22.04.2021 | Разработка входящих модулей. Тестирование и отладка. Разработка документации. |  |
| 23.04.2021 | Разработка документации. Защита проекта. |  |
| 26.04.2021 | Защита проекта. Сдача зачёта по практике. |  |