



Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
имени дважды Героя Советского Союза, летчика-космонавта А.А. Леонова

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:

Денисов М.В.

_____ (подпись)

Ларченко М.А.

_____ (подпись)

Преподаватель:

Гусятинер Л.Б.

_____ (подпись)

_____ (оценка)

Королев, 2021

Содержание отчёта

Раздел 1. Техника решения задач с использованием структурного программирования.....	4
2. Установка интерпретатора Python 3 и настройка окружения.	4
3. Техника работы в командной строке и среде IDLE.....	8
4. Техника работы с линейными программами	10
4.2. Техника работы с линейными программами	11
5. Техника работы с циклическими программами цикл.....	14
5.2. Техника работы с циклическими программами цикл.....	16
6. Техника работы с числами	19
6.2. Техника работы с числами.	20
7. Техника работы со строками	23
8. Техника работы со списками	28
8.2. Техника работы со списками	28
9. Техника работы с циклом for и генераторами списков.....	29
9.2. Техника работы с циклом for и генераторами списков.....	30
10. Техника работы с функциями.....	31
11. Техника работы со словарями	32
11.2. Техника работы со словарями	34
12. Техника работы с множествами	37
12.2. Техника работы с множествами	38
13. Техника работы с кортежами.....	40
14. Техника работы с файлами	41
14.2. Техника работы с файлами	41
Раздел 2. Техника решения задач с использованием библиотек.....	42
2.1. Установка и настройка среды JetBrains PyCharm	42
2.2. Техника работы с базами данных	45
2.3. Техника работы с библиотекой tkinter.....	46
2.4. Техника работы с библиотекой NumPy.....	49
2.5. Техника работы с библиотекой Matplotlib	51
2.6. Элементы работы с библиотекой PyQt	52
2.7. Элементы работы с библиотекой PyGame.....	54
Раздел 3. Разработка проекта с графическим интерфейсом.	55
3.1. Изучение входной и выходной документации.	55
3.2. Разработка требований к проекту. Построение диаграммы использования.	56
3.3. Разработка сценария проекта.	57
3.4. Разработка базы данных.	57

3.5 Разработка главного модуля.	58
3.6 Тестирование и отладка	61
3.7 Разработка документации к проекту.....	62
Дневник.	64

Раздел 1 Техника решения задач с использованием структурного программирования.

2 Установка интерпретатора Python 3 и настройка окружения.

Для установки интерпретатора Python на компьютер, первое, что нужно сделать – это скачать дистрибутив. Загрузить его можно с официального сайта, перейдя по ссылке <https://www.python.org/downloads/>

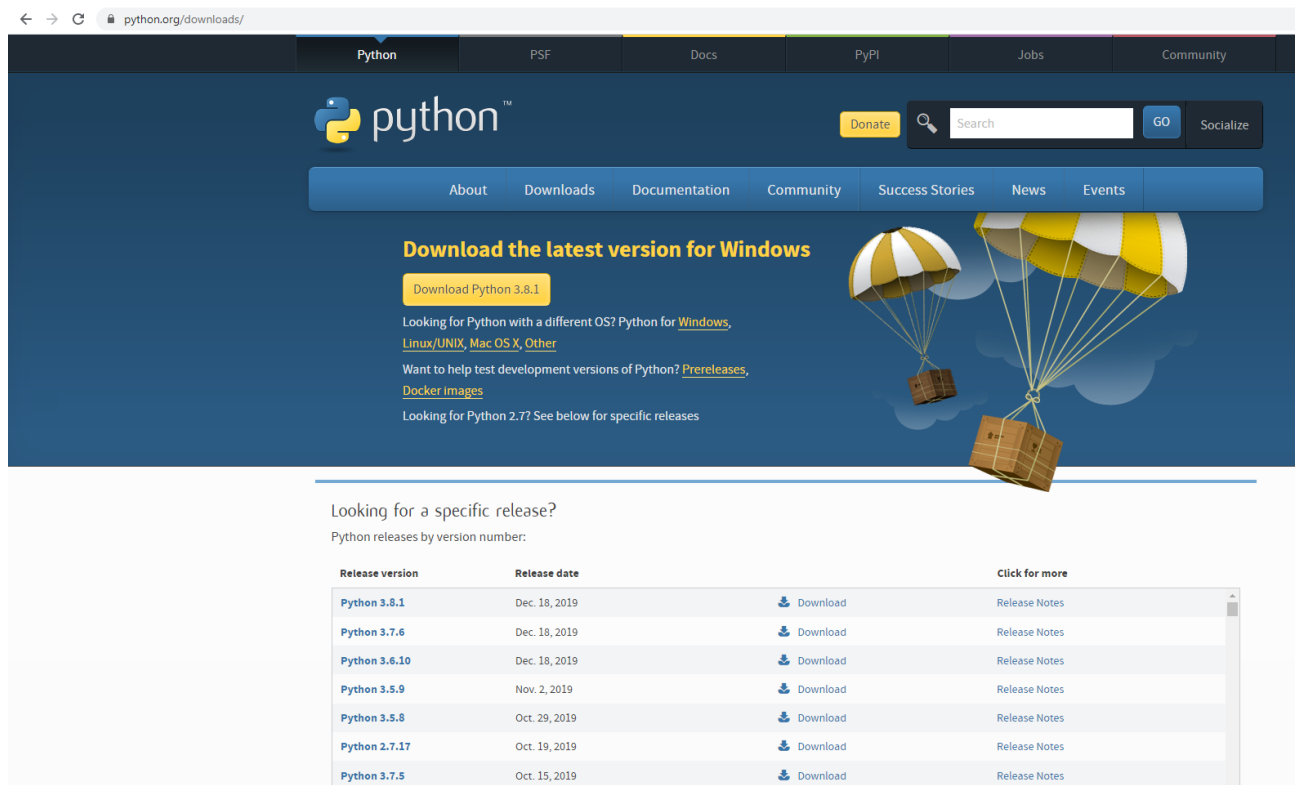


Рисунок 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.
2. Выбрать способ установки.

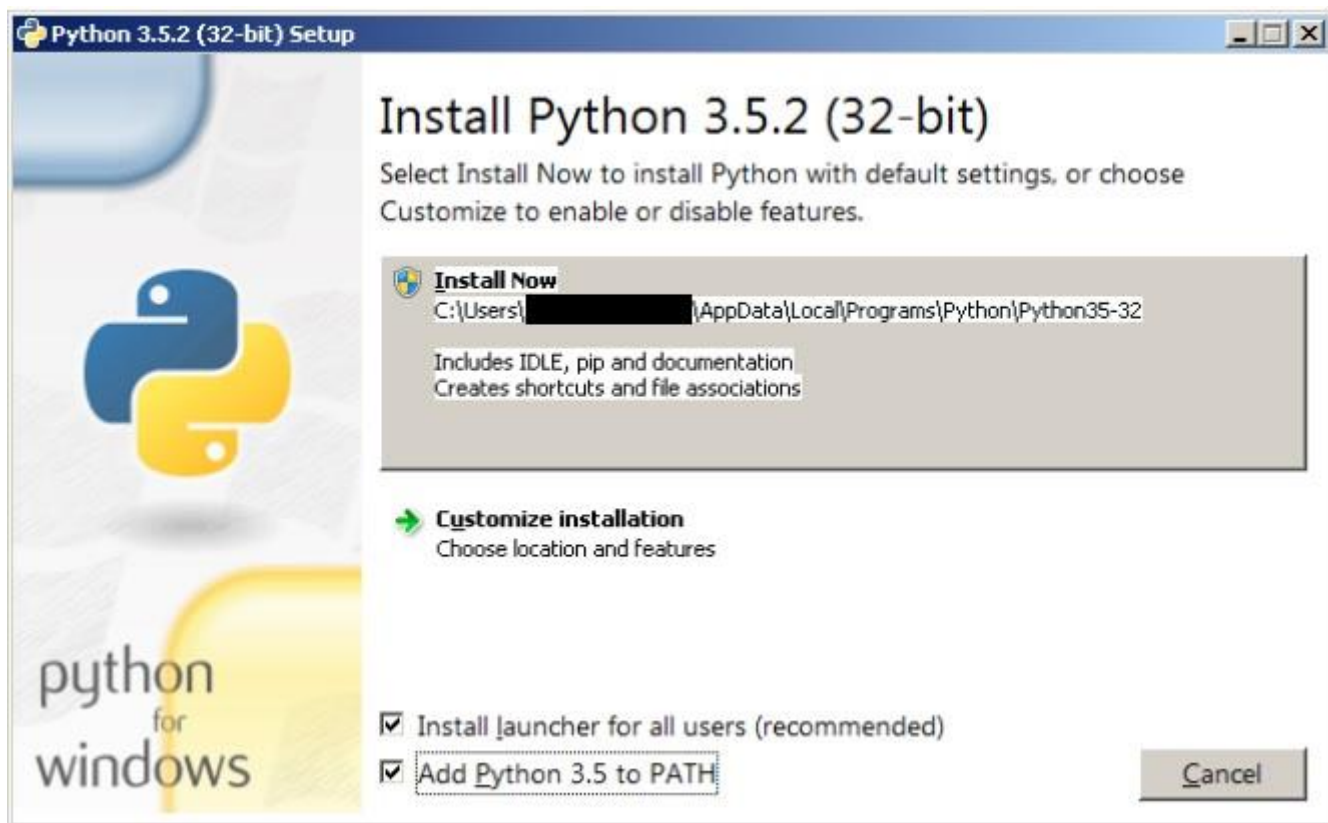


Рисунок 2. Установщик Python

3. Отметить необходимые опции установки (доступно при выборе Customize installation)



Рисунок 3. Опции установки

На этом шаге нам предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python. Выбираю:

- Documentation – установка документаций.
- pip – установка пакетного менеджера pip.
- tcl/tk and IDLE – установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса (tkinter).

4. Выбираем место установки (доступно при выборе Customize installation)

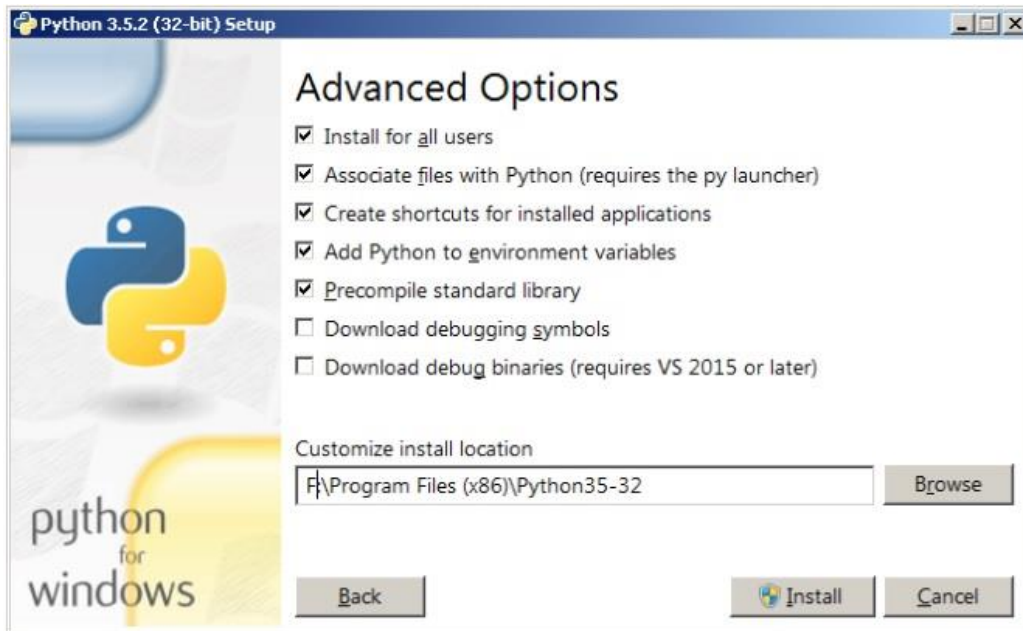


Рисунок 4. Продвинутые опции установки

5. После успешной установки:

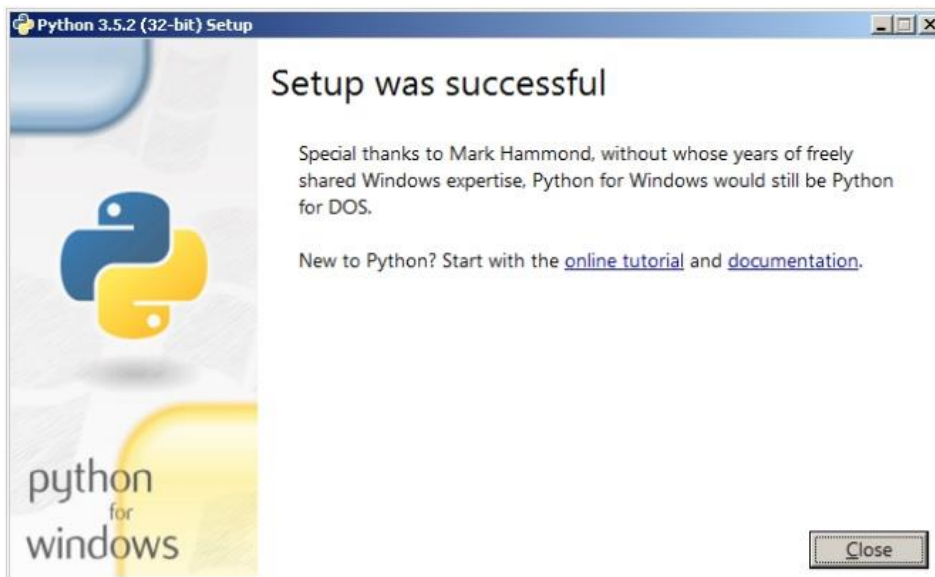


Рисунок 5. Сообщение об установке

Окружение Python представляет собой контекст, в котором выполняется код Python. Различают глобальные, виртуальные окружения и окружения

Conda. Окружение состоит из интерпретатора, библиотеки и нескольких установленных пакетов. Вместе они определяют, какие языковые конструкции и синтаксис допустимы, какие возможности операционной системы доступны и какие пакеты можно использовать.

В Visual Studio для Windows есть окно **Окружения Python**, которое позволяет управлять окружениями и выбрать одно из них в качестве окружения по умолчанию для новых проектов.

Окружения, обнаруженные Visual Studio, отображаются в окне **Окружения Python**. Для открытия окна выберите команду меню **Просмотр > Другие окна > Окружения Python**.

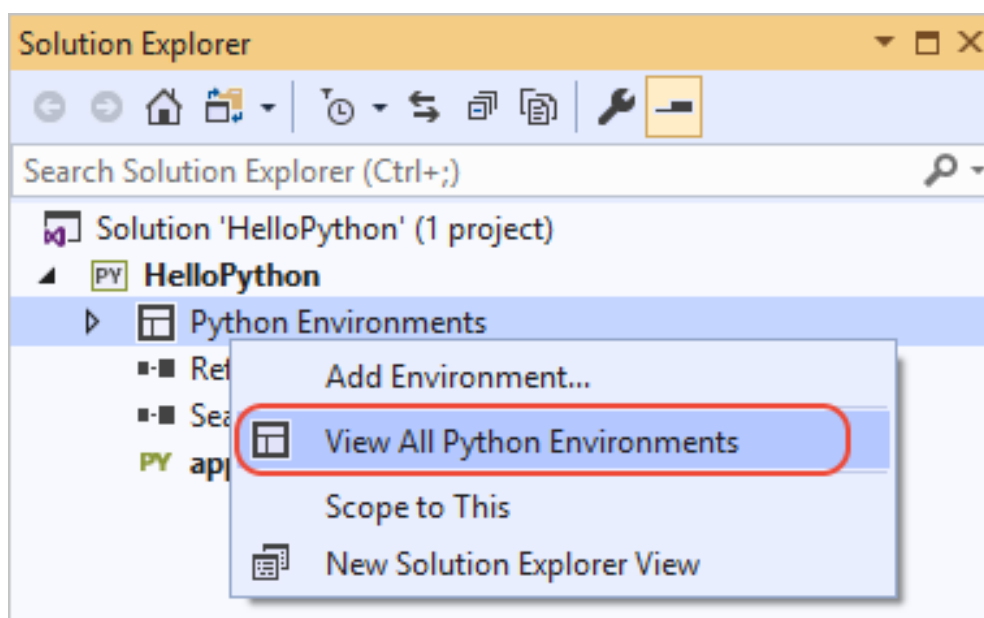


Рисунок 6. Показать Python инструменты

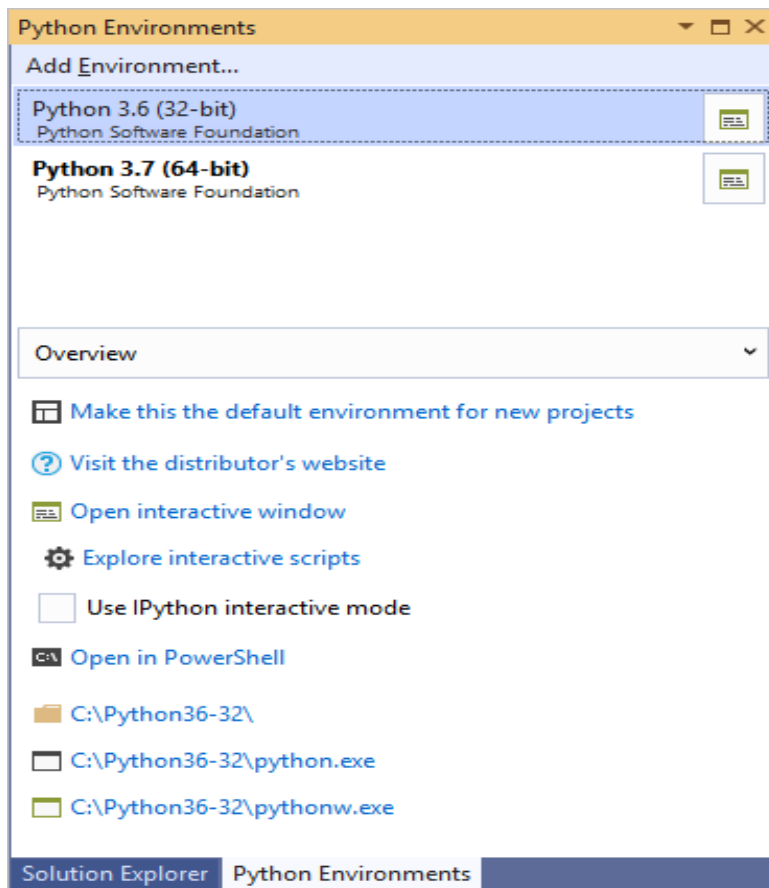


Рисунок 7. Выбор версии Python

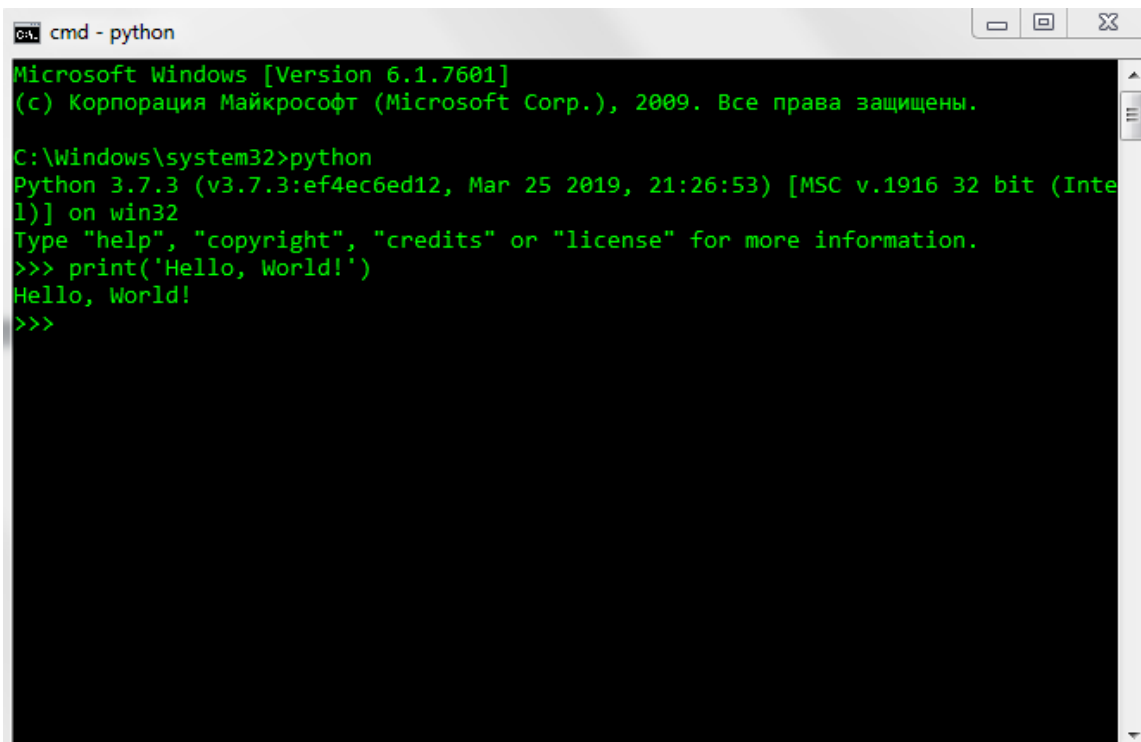
При выборе окружения в списке на вкладке **Обзор** Visual Studio отображаются различные свойства и команды для этого окружения. Например, как видно на рисунке выше, интерпретатор находится в папке *C:\Python36-32*. Четыре команды в нижней части вкладки **Обзор** открывают командную строку с выполняющимся интерпретатором.

Справа от каждого окружения в списке есть элемент управления, который позволяет открыть **интерактивное** окно для этого окружения.

Выполняя (запуская) команду “python” в вашем терминале, вы получаете интерактивную оболочку Python.

3 Техника работы в командной строке и среде IDLE

Листинг задания(1.3.1.txt)



```
cmd - python
Microsoft Windows [Version 6.1.7601]
(с) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Windows\system32>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('Hello, World!')
Hello, World!
>>>
```

Рисунок 8. Интерактивная оболочка Python

Существует несколько способов закрыть оболочку Python:

```
>>> exit()
```

или же

```
>>> quit()
```

Кроме того, **CTRL + D** закроет оболочку и вернет вас в командную строку терминала.

[IDLE](#) - простой редактор для Python, который поставляется вместе с Python.

Откройте IDLE в вашей системе выбора.

В оболочке есть подсказка из трех прямоугольных скобок:

```
>>>
```

Теперь напишите в подсказке следующий код:

```
>>> print("Hello, World")
```

Нажмите **Enter**.

```
>> print("Hello,world")
Hello,world
```

```
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print ("Hello,world")
Hello,world
]
>>> |
```

Рисунок 9.Первая программа

Продемонстрировать работу в командной строке, включая

- создание файла с кодом
- запуск
- import
- reload
- отработку ошибок

```
~ >>> python3 Python 3.8.6 (default, Sep 30 2020, 04:00:38)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>> import math
>>> import importlib
>>> importlib.reload(math)
<module 'math' from '/usr/lib/python3.8/lib-
dynload/math.cpython-38-x86_64-linux-gnu.so'> >>>
```

4 Техника работы с линейными программами

Листинг №1(1.4.1.txt)

Разработать программы по темам

- input

Функция input() в Python, ввод данных с клавиатура.

- print

Функция print() в Python, печатает объект.

- stdin, stdout, stderr модуля sys в Python.

- форматная строка и метод формат

```
x = input('Введите ваше имя: ')\nprint('Привет, {}'.format(x))
```

```
import sys\nsys.stdin.read(1)\nsys.stdout.write(x)\nsys.stdout.write('\n')
```

4.2. Техника работы с линейными программами

Задание 2.

Задание 2. Разработать программу с меню для демонстрации работы с типами данных:

список(list), словарь(dict), множество(set)

Меню -> выбор типа данных -> выбор метода -> краткая справка

Листинг №2(1.4.2.py)

“

```
list_menu = {
```

```
'append': 'добавляет элемент в конец списка',
```

```
'clear': 'удаляет все элементы из списка',
```

```
'copy': 'возвращает копию списка',
```

```
'count': 'возвращает количество элементов с заданным значением',
```

```

'extend': "добавить элементы списка (или любого итеративного) в конец
текущего списка",
'index': 'возвращает индекс первого элемента с указанным значением',
'insert': 'добавляет элемент в указанную позицию',
'pop': 'удаляет элемент в указанном положении',
'remove': 'удаляет первый элемент с указанным значением',
'reverse': 'меняет порядок списка',
'sort': 'сортировка списка'
}

dict_menu = {
'clear': 'удаляет все элементы из словаря',
'copy': 'возвращает копию словаря',
'fromkeys': 'возвращает словарь с указанными ключами и значением',
'get': 'возвращает значение указанного ключа',
'items': 'возвращает список, содержащий Кортеж для каждой пары ключ-
значение',
'keys': 'возвращает список, содержащий ключи словаря',
'pop': 'удаляет элемент с указанным ключом',
'popitem': 'удаляет последнюю вставленную пару ключ-значение',
'setdefault': 'возвращает значение указанного ключа. Если ключ не существует:
вставьте ключ с указанным значением',
'update': 'обновляет словарь с помощью указанных пар ключ-значение',
'values': 'возвращает список всех значений в словаре'
}

set_menu = {
'add': 'добавляет элемент в набор',
'clear': 'удаляет все элементы из набора',
'copy': 'возвращает копию набора',
'difference': 'возвращает набор, содержащий разницу между двумя или более
наборами',

```

'difference_update': 'удаляет элементы в этом наборе, которые также включены в другой, указанный набор',
'discard': 'удалить указанный элемент',
'intersection': 'возвращает набор, то есть пересечение двух других наборов',
'intersection_update': 'удаляет элементы в этом наборе, которые отсутствуют в других, указанных наборах',
'isdisjoint': 'возвращает, имеют ли два множества пересечение или нет',
'issubset': 'возвращает, содержит ли другой набор этот набор или нет',
'issuperset': 'возвращает, содержит ли этот набор другой набор или нет',
'pop': 'удаляет элемент из набора',
'remove': 'удаляет указанный элемент',
'symmetric_difference': 'возвращает набор с симметричными разностями двух наборов',
'symmetric_difference_update': 'вставляет симметричные разности из этого набора и другого',
'union': 'возвращает набор, содержащий объединение множеств,
'update': 'обновить набор с Союзом этот набор и другие
“

```
type_menu = {'list': list_menu, 'dict': dict_menu, 'set':  
set_menu}  
  
print('\n'.join(type_menu))  
sel_type = input('> ')  
if sel_type not in type_menu.keys():  
    print('no such entry')  
    exit(1)  
print('\n'.join(type_menu[sel_type].keys()))  
  
sel_method = input('> ')  
if sel_method not in type_menu[sel_type].keys():  
    print('no such entry')  
    exit(1)  
print(type_menu[sel_type][sel_method])
```

5 Техника работы с циклическими программами цикл while

Задание 1.

На плоскости нарисован квадрат заданного размера с левой нижней вершиной в начале координат. В квадрат вписывается окружность. Случайным образом в квадрате выбирается 1000 точек.

- а) нужно определить, сколько точек попало внутрь круга
- б) считая количество точек пропорциональным площади, найти отношение площадей
- круга и квадрата
- в) по этому отношению определить приближённое значение числа пи
- г) определить, насколько найденное значение отличается от "библиотечного".

Листинг №3(K5-1.1.py)

```
from random import randrange
import math

tochki = 1000
vershina = int(input())
radiys = vershina/2
schetchik = 0
for i in range(tochki):
    x = randrange(0, vershina)
    y = randrange(0, vershina)
    if ((x-radiys)**2 + (y-radiys)**2 <= radiys**2):
        schetchik += 1
otnoshenie = schetchik/tochki
p = 4*schetchik/tochki
raznica = 3.14-p

print("Точек внутри круга:", schetchik)
print("Отношение площадей круга и квадрата:", otnoshenie)
print("Приблизительное число PI:", p)
print("Отличие от библиотечного числа pi:", raznica)
```

Задание 2.

Придумать пример(ы) на использование break / continue / else.

Листинг №4. (k5_1.py)

```
import random

for i in range(5):
    x = random.randint(0, 25)
    if 10 <= x <= 15:
        print('выпало число', x, 'на итерации', i)
        break
    else:
        continue
else:
    print('ни одно число не попало в промежуток 10 <= x <= 15')
```

5.2 Техника работы с циклическими программами цикл while

Задание 2.

<https://stepik.org/lesson/3364/step/11?unit=947>

#Напишите программу, которая считывает со стандартного ввода #целые числа, по одному числу

#в строке, и после первого введенного нуля выводит сумму #полученных на вход чисел.

Листинг №5(5.2.5.py)

```
sum = 0
num = 1

while num != 0:
    num = int(input())
    sum += num

print(sum)
```

Задание 3.

Разработать программу для нахождения наибольшего общего делителя

Листинг №6(k5_2.py)

```
from collections import Counter

# https://ru.wikipedia.org/wiki/Перебор\_делителей
# на больших числах типа lkkk вроде работает быстро
def integer_factorization(n):
    tmp = n
    p_nums = []
    while True:
        for p_num in prime_numbers():
            if tmp % p_num == 0:
```



```

        tmp = tmp / p_num
        p_nums.append(p_num)
        break

    i = 2
    j = 0
    while (i * i <= tmp) and (j != 1):
        if tmp % i == 0:
            j = 1
            i += 1

    if j == 0:
        p_nums.append(int(tmp))
        break
    return Counter(p_nums)

def prime_numbers():
    n = 2
    while True:
        i = 2
        j = 0
        while (i * i <= n) and (j != 1):
            if n % i == 0:
                j = 1
                i += 1

        if j != 1:
            yield n

        n += 1

def gcd(nums):
    diff = integer_factorization(nums[0]) &
integer_factorization(nums[1])
    if diff == Counter():
        return None
    else:
        gcd_n = 1
        for n in diff:
            gcd_n *= n
        return gcd_n

n1 = int(input('Введите первое число НОД: '))
n2 = int(input('Введите второе число НОД: '))
print(gcd([n1, n2]))

```

Задание 4.

С использованием результата задания 2 разработать программу для #нахождения наименьшего общего кратного
Листинг №7(k5.1.py)

```

sum = 0
num = 1
sum1 = 0
while num != 0:
    sum1 = sum
    num = int(input())
    sum += num
i = min(sum, sum1)
while True:
    if i%sum==0 and i%sum1==0:
        break
    i += 1
print(i)

```

Задание 5.

#напишите программу, которая выводит часть последовательности 1 #2 2 3 3 3 4 4 4 4 5 5 5 5 5 ...

#(число повторяется столько раз, чему равно).

#На вход программе передаётся неотрицательное целое число n — #столько элементов

#последовательности должна отобразить программа.

#На выходе ожидается последовательность чисел, записанных через #пробел в одну строку.

#Например, если n = 7, то программа должна вывести 1 2 2 3 3 3 #4.

#Sample Input:

#7

#Sample Output:

#1 2 2 3 3 3 4

Листинг 8(<https://stepik.org/lesson/3369/step/8?unit=952>)

```

konec = int(input())
chislo = 1
schetchik = 0
while schetchik != konec:
    for i in range(chislo):
        print(chislo, end=' ')
    schetchik += 1
    if schetchik == konec:
        break
    chislo += 1

```

6 Техника работы с числами

Задание 1.

#Составить и выполнить по 3 примера использования модулей для работы с
#дробными числами (fractions), для точных вычислений #(decimal).

```
///1///  
import fractions
```

```
>>>Fraction(153,272)          #автоматическое уменьшение дроби  
Fraction(9,16)
```

```
>>>Fraction(1,2) + Fraction(3,4) #двоичные операции над дробью  
Fraction(5,4)
```

```
>>>Fractions.gcd(6,9)         #Наибольший общий делитель
```

```
///2///  
import decimal
```

```
>>>num1=Decimal("0.1")  
>>>num2=Decimal("0.7")      #Округление десятичной дроби  
>>>print(num1+num2)         с плавающей точкой  
0.8
```

```
>>>getcontext().prec=2  
>>>print(Decimal("4.34")/4)  #Точность в значениях  
4
```

```
>>>decimal.Decimal("3.14")   #Представление в виде дроби
```

6.2 Техника работы с числами.

Задание 1.

Подготовить инструкцию по использованию модулей `math` и `cmath`.

Арифметические функции

Эти функции выполняют различные арифметические операции, такие как вычисление пола, потолка или абсолютного значения числа с использованием функций `floor(x)`, `ceil(x)` и `fabs(x)` соответственно. Функция `ceil(x)` вернет наименьшее целое число, которое больше или равно x . Аналогично, `floor(x)` возвращает наибольшее целое число, меньшее или равное x . Функция `fabs(x)` возвращает абсолютное значение x , также можно выполнять нетривиальные операции, такие как вычисление факториала числа с использованием `factorial(x)`. Факториал является произведением целого числа и всех положительных целых чисел, меньших его. Он широко используется при работе с комбинациями и перестановками. Его также можно использовать для вычисления значения функций синуса и косинуса.

Тригонометрические функции

Эти функции связывают углы треугольника по бокам. У них много приложений, в том числе изучение треугольников и моделирование периодических явлений, таких как звуковые и световые волны. Имейте в виду, что угол, который вы предоставляете, находится в радианах. Можно рассчитать `sin(x)`, `cos(x)` и `tan(x)` непосредственно с помощью этого модуля. Однако нет прямой формулы для вычисления `cosec(x)`, `sec(x)` и `cot(x)`, но их значение равно обратному значению, возвращаемому `sin(x)`, `cos(x)` и `tan(x)` соответственно. Вместо того, чтобы вычислять значение тригонометрических функций под определенным углом,

также можно сделать обратный и рассчитать угол, в котором они имеют заданное значение, используя $\text{asin}(x)$, $\text{acos}(x)$ и $\text{atan}(x)$.

Гиперболические функции

Гиперболические функции являются аналогами тригонометрических функций, которые основаны на гиперболе вместо круга. В тригонометрии точки $(\cos b, \sin b)$ представляют точки единичного круга. В случае гиперболических функций точки $(\cosh b, \sinh b)$ представляют точки, которые образуют правую половину равносторонней гиперболы. Точно так же, как тригонометрические функции, вы можете непосредственно вычислить значение $\sinh(x)$, $\cosh(x)$ и $\tanh(x)$. Остальные значения могут быть рассчитаны с использованием различных отношений между этими тремя значениями. Существуют также другие функции $\text{asinh}(x)$, $\text{acosh}(x)$ и $\text{atanh}(x)$, которые могут быть использованы для вычисления обратных соответствующих гиперболических значений.

Сложные числа

Сложные числа хранятся внутри с использованием прямоугольных или декартовых координат. Комплексное число z будет представлено в декартовых координатах как $z = x + iy$, где x представляет действительную часть, а y представляет собой мнимую часть. Другим способом их представления является использование полярных координат.

В этом случае комплексное число z будет определяться комбинацией модуля r и фазового угла ϕ . Модуль r является расстоянием между комплексным числом z и началом. Угол ϕ - угол против часовой стрелки, измеренный в радианах от положительной оси x до отрезка линии, соединяющего z и начало координат.

При работе с комплексными числами модуль `cmath` может оказать большую помощь. Модуль комплексного числа может быть рассчитан с использованием встроенной функции `abs()`, и его фаза может быть рассчитана с использованием функции `phase(z)`, доступной в модуле `cmath`. Вы можете

преобразовать комплексное число в прямоугольной форме в полярную форму, используя `polar(z)`, которая вернет пару (r, ϕ) , где $r = \text{abs}(z)$, а $\phi = \text{phase}(z)$.

Заключение

Все эти функции, о которых мы говорили выше, имеют свои конкретные приложения. Например, можно использовать функцию `factorial(x)` для решения проблем с перестановкой и комбинацией. Можно использовать тригонометрические функции для преобразования вектора в декартовы координаты. Также можно использовать тригонометрические функции для имитации периодических функций, таких как звуковые и световые волны.

Аналогично, кривая веревки, висящая между двумя полюсами, может быть определена с использованием гиперболической функции. Поскольку все эти функции доступны непосредственно в модуле `math`, очень легко создавать небольшие программы, которые выполняют все эти задачи.

7 Техника работы со строками

Задание 1.

#С клавиатуры вводятся строки, последовательность заканчивается #точкой. Выведите буквы введенных слов в верхнем регистре, #разделяя их пробелами.

Листинг №9(K7-1.1.py)

```
slovo = input()
while slovo != ".":
    print(" ".join(slovo.upper()))
    slovo = input()
```

Задание 2.

#Известно, что для логина часто не разрешается использовать #строки содержащие пробелы. Но пользователю нашего сервиса #особенно понравилась какая-то строка. Замените пробелы в строке #на символы нижнего подчеркивания, чтобы строка могла сгодиться #для логина. Если строка состоит из одного слова, менять ничего #не нужно.

#Sample Input: python sila

#Sample Output: python_sila

Листинг №10(K7-1.2.py)

```
print(input().replace(' ', '_'))
```

Задание 3.

#Уберите точки из введенного IP-адреса. Выведите сначала четыре #числа через пробел, а затем сумму получившихся чисел.

Sample Input:

192.168.0.1

Sample Output:

192 168 0 1

361

Листинг №11(K7-1.3.py)

```
chisla = input().replace(".", " ")
print(chisla)
chisla =chisla.split()
```

```
print(int(chisla[0])+int(chisla[1])+int(chisla[2])+int(chisla[3])
))
```

Задание 4.

Программист логирует программу, чтобы хорошо знать, как она себя ведет (эта весьма распространенная и важная практика). Он использует разные типы сообщений для вывода ошибок (error), предупреждений (warning), информации (info) или подробного описания (verbose). Сообщения отличаются по внешнему виду. Назовем модификаторами такие символы, которые отличают сообщения друг от друга, позволяя программисту понять, к какому из типов относится сообщение. Модификаторы состоят из двух одинаковых символов и записываются по разу в начале и в конце строки.

@@ обозначает ошибку

!! обозначает предупреждение

// обозначает информационное сообщение

** обозначает подробное сообщение

Напишите программу, которая принимает строки до точки и выводит, какого типа это сообщение. Если сообщение не содержит модификаторов, проигнорируйте его.

Sample Input:

!! cannot resolve this method !!

@@ invalid type @@

@@ StackOverFlowException @@

// here I change the variables name //

** this class is used for operating with the database, including CRUD operations and registering new users **

error on line 42

// TODO: optimize recursive calls //

Sample Output:

предупреждение

ошибка

ошибка

информация

подробное сообщение

информация

Листинг №12(K-7-1.4.py)

```
while True:
    b = input()
    if b == "***":
        print("подробное сообщение")
    elif "@" in b:
        print("ошибка")
    elif "!!" in b:
        print("предупреждение")
    elif "/" in b:
        print("информация")
    elif "." in b:
        break
```

Задание 1.

Подготовить сравнительную инструкцию по использованию форматирования строк

f-строки в Python

Начиная с версии 3.6 в Python появился новый тип строк — f-строки, которые буквально означают «formatted string». Эти строки улучшают читаемость кода, а также работают быстрее чем другие способы форматирования.

1. Конкатенация. Грубый способ форматирования, в котором мы просто склеиваем несколько строк с помощью операции сложения:

```
>>> name = "Дмитрий"
>>> age = 25
>>> print("Меня зовут " + name + ". Мне " + str(age) + " лет.")
>>> Меня зовут Дмитрий. Мне 25 лет.
```

2. %-форматирование. Самый популярный способ, который перешел в Python из языка C. Передавать значения в строку можно через списки и кортежи, а также и с помощью словаря. Во втором случае значения помещаются не по позиции, а в соответствии с именами.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print("Меня зовут %s. Мне %d лет." % (name, age))
>>> Меня зовут Дмитрий. Мне 25 лет.
>>> print("Меня зовут %(name)s. Мне %(age)d лет." % {"name":
name, "age": age})
>>> Меня зовут Дмитрий. Мне 25 лет.
```

3. Template-строки. Этот способ появился в Python 2.4, как замена %-форматированию (PEP 292), но популярным так и не стал. Поддерживает передачу значений по имени и использует \$-синтаксис как в PHP.

```
>>> from string import Template
>>> name = "Дмитрий"
>>> age = 25
>>> s = Template('Меня зовут $name. Мне $age лет.')
>>> print(s.substitute(name=name, age=age))
>>> Меня зовут Дмитрий. Мне 25 лет.
```

4. Форматирование с помощью метода format(). Этот способ появился в Python 3 в качестве замены %-форматированию. Он также поддерживает передачу значений по позиции и по имени.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print("Меня зовут {}. Мне {} лет.".format(name, age))
>>> Меня зовут Дмитрий. Мне 25 лет.
>>> print("Меня зовут {name} Мне {age} лет.".format(age=age,
name=name))
>>> Меня зовут Дмитрий. Мне 25 лет.
```

5. f-строки. Форматирование, которое появилось в Python 3.6 (PEP 498). Этот способ похож на форматирование с помощью метода format(), но гибче, читабельней и быстрее.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print(f"Меня зовут {name} Мне {age} лет.")
>>> Меня зовут Дмитрий. Мне 25 лет.
```

Погружение в F-строки

f-строки делают очень простую вещь — они берут значения переменных, которые есть в текущей области видимости, и подставляют их в строку. В

самой строке вам лишь нужно указать имя этой переменной в фигурных скобках.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print(f"Меня зовут {name} Мне {age} лет.")
>>> Меня зовут Дмитрий. Мне 25 лет.
```

f-строки также поддерживают расширенное форматирование чисел:

```
>>> from math import pi
>>> print(f"Значение числа pi: {pi:.2f}")
>>> Значение числа pi: 3.14
```

С помощью f-строк можно форматировать дату без вызова метода strftime():

```
>>> from datetime import datetime as dt
>>> now = dt.now()
>>> print(f"Текущее время {now:%d.%m.%Y %H:%M}")
>>> Текущее время 24.02.2017 15:51
```

Они поддерживают базовые арифметические операции. Да, прямо в строках:

```
>>> x = 10
>>> y = 5
>>> print(f"{x} x {y} / 2 = {x * y / 2}")
>>> 10 x 5 / 2 = 25.0
```

Позволяют обращаться к значениям списков по индексу:

```
>>> planets = ["Меркурий", "Венера", "Земля", "Марс"]
>>> print(f"Мы живим не планете {planets[2]}")
>>> Мы живим не планете Земля
```

А также к элементам словаря по ключу:

```
>>> planet = {"name": "Земля", "radius": 6378000}
>>> print(f"Планета {planet['name']}. Радиус {planet['radius']/1000} км.")
>>> Планета Земля. Радиус 6378.0 км.
```

Причем вы можете использовать как строковые, так и числовые ключи. Точно

также как в обычном Python коде:

```
>>> digits = {0: 'ноль', 'one': 'один'}
>>> print(f"0 - {digits[0]}, 1 - {digits['one']}")
>>> 0 - ноль, 1 - один
```

Вы можете вызывать в f-строках методы объектов:

```
>>> name = "Дмитрий"
>>> print(f"Имя: {name.upper()}")
>>> Имя: ДМИТРИЙ
```

А также вызывать функции:

```
>>> print(f"13 / 3 = {round(13/3)}")
>>> 13 / 3 = 4
```

f-строки очень гибкий и мощный инструмент для создания самых разнообразных шаблонов.

8 Техника работы со списками

Задание 1.

https://pythontutor.ru/lessons/lists/problems/more_than_neighbours/

Задача «Больше своих соседей»

Дан список чисел. Определите, сколько в этом списке элементов, которые больше двух своих соседей, и выведите количество таких элементов. Крайние элементы списка никогда не учитываются, поскольку у них недостаточно соседей.

Листинг №13(K8-1.1.py)

```
chisla = [int(i) for i in input().split()]
schetchik = 0
for i in range(2, len(chisla)):
    if chisla[i-2] < chisla[i-1] > chisla[i]:
        schetchik += 1
print(schetchik)
```

8.2 Техника работы со списками

Задание 1.

Array112. Дан массив A размера N.

Упорядочить его по возрастанию методом сортировки простым обменом: просматривать массив, сравнивая его соседние элементы и меняя их местами, если левый элемент пары больше правого; повторить описанные действия N - 1 раз. Для контроля за выполняемыми действиями выводить содержимое массива после каждого просмотра.

Учесть, что при каждом просмотре количество анализируемых пар можно уменьшить на 1.

Листинг №14(K8-2.1.py)

```
chislo = int(input())
mas= []
```

```

for x in range(chislo):
    mas.append(int(input()))
print(*mas)
for i in range(chislo - 1):
    for j in range(chislo - i - 1):
        if mas[j] > mas[j + 1]:
            mas[j], mas[j + 1] = mas[j + 1], mas[j]
        print(*mas)
print("\n Отсортированный массив:", *mas)

```

9 Техника работы с циклом for и генераторами списков

Задание 1.

```

# Каждый студент может программировать только на одном языке
# и занимать только одну позицию.
# Дан текстовый файл, содержащий перечень студентов с указанием языка и
позиции
# (каждый студент с новой строки)
# Требуется
# 1. Получить список студентов с указанием языка и позиции
# 2. Сформировать список всевозможных команд
# 3. Вывести список команд с указанием состава и названия команды:
# Команда 1
# coder: ...
# designer: ...
# tester: ...
# writer: ...
#
# Команда 2
# ...
# Пункты 1 и 2 выполнить с использованием генераторов списка
# Name Team Speciality Lang

```

Листинг №15(K9_1.py и Tester.txt)

```
import sys

print('Team analyzer v0.1')

if len(sys.argv) > 1:
    filename = sys.argv[1]
else:
    filename = 'teams.txt'

file = open(filename)

people = [x.split() for x in file]
#print(people)
print('Lines readed:', len(people))
teams = {x[1] for x in people}

specialities = ['coder', 'writer', 'tester', 'designer']

for team in teams:
    print(team)
    team_members = [x for x in people if x[1] == team]
    for speciality in specialities:
        member = [x for x in team_members if x[2] == speciality]
        if len(member) == 0:
            print(' {}: no member with this
speciality'.format(speciality))
        else:
            member = member[0]
            print(' {}: {} {}'.format(speciality, member[0],
member[3]))
```

9.2 Техника работы с циклом for и генераторами списков

Задание 5.

Matrix56. Дана матрица размера М х N (N — четное число). Поменять местами

левую и правую половины матрицы.

Листинг №16(K9_2.py)

```
from random import randint
import sys

if len(sys.argv) < 3:
    print('too few arguments')
```

```

    exit(1)

M = int(sys.argv[1])
N = int(sys.argv[2])

if (N % 2) != 0:
    print('N is not even')
    exit(1)

def print_matrix(mat):
    for row in mat:
        print(' '.join(map(str, row)))

matrix = [[randint(0, 9) for _ in range(N)] for _ in range(M)]
print('original:')
print_matrix(matrix)

for row in matrix:
    for x in range(N//2):
        # print(x, N-x-1)
        row[x] += row[N-x-1]
        row[N-x-1] = row[x] - row[N-x-1]
        row[x] = row[x] - row[N-x-1]

print('mirrored:')
print_matrix(matrix)

```

10 Техника работы с функциями

Задание 2.

Func6. Описать функцию SumRange(A, B) целого типа, находящую сумму всех целых

чисел от A до B включительно (A и B — целые). Если $A > B$, то функция возвращает 0.

С помощью этой функции найти суммы чисел от A до B и от B до C, если даны числа A, B, C.

Листинг №17(K10_1.py)

```

def SumRange(A: int, B: int) -> int:
    return sum(range(A, B+1))

A = int(input('A: '))
B = int(input('B: '))

```

```
C = int(input('C: '))

print('sum A -> B:', SumRange(A, B))
print('sum B -> C:', SumRange(B, C))
```

Задание 3.

Использовать lambda, filter.

Array55. Дан целочисленный массив A размера N (≤ 15). Переписать в новый целочисленный

массив B все элементы с нечетными порядковыми номерами (1, 3, ...) и вывести размер

полученного массива B и его содержимое. Условный оператор не использовать.

Листинг №18(K10_2.py)

```
print(
    list(
        map(lambda x: x[1],
            filter(lambda x: (x[1] % 2) != 0,
                enumerate(
                    map(int, input(': ').split())
                )
            )
        )
    )
)
```

11 Техника работы со словарями

Задание 2

<https://pythontutor.ru/lessons/dicts/problems/permissions/>

Задача «Права доступа»

Условие. В файловую систему одного суперкомпьютера проник вирус, который сломал контроль за правами доступа к файлам. Для каждого файла известно, с какими действиями можно к нему обращаться:

- # запись W,
- # чтение R,
- # запуск X.

В первой строке содержится число N — количество файлов содержащихся в данной файловой системе. В следующих N строчках содержатся имена файлов и допустимых с ними операций, разделенные пробелами. Далее указано число M — количество запросов к файлам. В последних M строках указан запрос вида Операция Файл. К одному и тому же файлу может быть применено любое количество запросов.

Вам требуется восстановить контроль над правами доступа к файлам (ваша программа для каждого запроса должна будет возвращать ОК если над файлом выполняется допустимая операция, или же Access denied, если операция недопустима.

Листинг №19(k11_1.py)

```
N = int(input('File count: '))

operations = ('r', 'w', 'x')

files = dict()
for _ in range(N):
    x = input().strip().split()
    files[x[0]] = map(str.lower, x[1:])
    # if files[x[0]] not in operations:

M = int(input('Operation count: '))
```

```

for _ in range(M):
    x = input().split()

    if x[1] not in files:
        print('File not exist')
        continue
    if x[0].lower() not in operations:
        print('Invalid operation')
        continue

    if x[0].lower() in files[x[1]]:
        print('OK')
    else:
        print('Access denied')

```

11.2 Техника работы со словарями

Задание 1. <https://stepik.org/lesson/243394/step/4?unit=215740>

Телефонная книга. Этап 1. Коля устал запоминать телефонные номера и заказал у Вас

программу, которая заменила бы ему телефонную книгу. Коля может послать программе

два вида запросов: строку, содержащую имя контакта и его номер, разделенные пробелом,

или просто имя контакта. В первом случае программа должна добавить в книгу новый номер,

во втором – вывести номер контакта. Ввод происходит до символа точки. Если введенное

имя уже содержится в списке контактов, необходимо перезаписать номер.

Sample Input:

Ben 89001234050

Alice 210-220

Alice

Alice 404-502

Nick +16507811251

Ben

Alex +4(908)273-22-42

Alice

Nick

Robert 51234047129

Alex

.

Sample Output:

210-220

89001234050

404-502

+16507811251

+4(908)273-22-42

Задание 2. <https://stepik.org/lesson/243394/step/8?unit=215740>

Телефонная книга. Этап 2. Коля понял, что у многих из его знакомых есть несколько

телефонных номеров и нельзя хранить только один из них. Он попросил доработать Вашу

программу так, чтобы можно было добавлять к существующему контакту новый номер или даже

несколько номеров, которые передаются через запятую. По запросу телефонного номера

должен выводиться весь список номеров в порядке добавления, номера должны разделяться

запятой. Если у контакта нет телефонных номеров, должна выводиться строка "Не найдено".

Sample Input:

Ben 89001234050, +70504321009

Alice 210-220

Alice

Alice 404-502, 894-005, 439-095

Nick +16507811251

Ben

Alex +4(908)273-22-42

Alice

Nick

Robert 51234047129, 92174043215

Alex

Robert

.

Sample Output:

210-220

89001234050, +70504321009

210-220, 404-502, 894-005, 439-095

+16507811251

+4(908)273-22-42

51234047129, 92174043215

Задание 3. <https://stepik.org/lesson/243394/step/13?unit=215740>

Телефонная книга. Этап 3. Коле очень понравилась Ваша программа, однако он стал

замечать, что иногда в его телефонную книгу попадают номера в некорректном формате.

Чтобы не сохранять недействительные номера, он попросил Вас обрабатывать только номера,

соответствующие критериям:

- номер должен начинаться либо с +7, либо с 8 и состоять из 11 цифр.

- блоки цифр могут разделяться пробелами или дефисами.

- вторая, третья и четвертая цифры могут помещаться в скобки.

Если программа встречает некорректный номер, она должна его проигнорировать. В обратном

случае она должна привести номер к виду +7 (900) 800-70-60 и запомнить. Остальной

функционал программы остается без изменений.

Sample Input:

Ben 89001234050, +7 050 432 10-09

Alice 404-502, 894053212-65, 439-095

Nick +1(650)781 12-51

Ben

Alex +4(908)273-22-421, 8 (908) 273-22-42

Alice

Nick

Robert 51234047129, 89174043215

Alex

Robert

Sample Output:

+7 (900) 123-40-50, +7 (050) 432-10-09

+7 (940) 532-12-65

Не найдено

+7 (908) 273-22-42

+7 (917) 404-32-15

Листинг №20(k11_2..py)

```
import re

print('Welcome to SasContacts v0.1')

book = dict()
while (cmd := input()) != '.':
    cmd = cmd.lower().split(maxsplit=1)
    if len(cmd) == 1:
        if cmd[0] in book:
            print(', '.join(book[cmd[0]]))
        else:
            print('Не найдено')
    if len(cmd) > 1:
        for x in cmd[1].split(','):
            #очистка строки
            x = x.strip()
            if re.match(r'(?:\+7|8)(?:[()]\s#-)*\d{10}', x):
                #очистка 2
                x = x.replace(' ', '').replace('-', '')
                x = x.replace('(', '').replace(')', '')
                x =
re.sub(r'(?:\+7|8)(\d{3})(\d{3})(\d{2})(\d{2})', r'+7 (\1) \2-\3-\4', x)

            #print(x)
            if cmd[0] in book:
                book[cmd[0]].append(x)
            else:
                book[cmd[0]] = [x]
```

12 Техника работы с множествами

Задание 1. https://pythontutor.ru/lessons/sets/problems/number_of_unique/

Задача «Количество различных чисел»

Условие. Дан список чисел. Определите, сколько в нем встречается различных чисел.

Листинг №21(k12_1.py)

```
import random
```

```

print(
    len(
        set(
            map(int, input('Введите числа через пробел:
').split())
        )
    )
)

```

```

print('Задание 1')

```

```

x = [random.randint(0,20) for _ in range(10)]
print(x)
print(len(set(x)))

```

Задание 2. https://pythontutor.ru/lessons/sets/problems/number_of_coincidental/

Задача «Количество совпадающих чисел»

Условие. Даны два списка чисел. Посчитайте, сколько чисел содержится одновременно как

в первом списке, так и во втором.

```

print('Задание 2')

```

```

x = [random.randint(0,20) for _ in range(10)]
y = [random.randint(0,20) for _ in range(10)]

```

```

print(x)
print(y)
print(len(set(x) & set(y)))

```

Задание 3. https://pythontutor.ru/lessons/sets/problems/sets_intersection/

Задача «Пересечение множеств»

Условие. Даны два списка чисел. Найдите все числа, которые входят как в первый,

так и во второй список и выведите их в порядке возрастания.

```

print('Задание 3')

```

```

x = [random.randint(0,20) for _ in range(10)]
y = [random.randint(0,20) for _ in range(10)]

```

```

print(x)
print(y)
print(sorted(set(x) & set(y)))

```

12.2 Техника работы с множествами

Задание 1. <https://stepik.org/lesson/3380/step/3?unit=963>

Простейшая система проверки орфографии может быть основана на использовании списка известных слов.

Если введённое слово не найдено в этом списке, оно помечается как "ошибка".

Попробуем написать подобную систему.

На вход программе первой строкой передаётся количество *d* известных нам слов, после чего

на *d* строках указываются эти слова.

Затем передаётся количество *l* строк текста для проверки, после чего *l* строк текста.

Выведите уникальные "ошибки" в произвольном порядке. Работу производите без учёта регистра.

Sample Input:

4

champions

we

are

Stepik

3

We are the champignons

We Are The Champions

Stepic

Sample Output:

stepic

champignons

the

Листинг №22(k12_2.py)

```
words = set()
d = int(input('d: '))
for _ in range(d):
    words.add(input().lower())

text_words = list()
l = int(input('l: '))
for _ in range(l):
    text_words.extend(input().lower().split())

print('\nOutput:')
print(*(set(text_words)-words), sep='\n')
```

13 Техника работы с кортежами

Задание 1. <https://stepik.org/lesson/193753/step/4?unit=168148>
Вывести чётные
Необходимо вывести все четные числа на отрезке [a; a * 10].
Sample Input:
2
Sample Output:
(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
Листинг №23(k13_1.py)

```
x = int(input())

print(tuple(range(x, x*10+1, 2)))
```


14 Техника работы с файлами

Задание 1. <http://ptaskbook.com/ru/tasks/text.php>

Text5. Дана строка S и текстовый файл. Добавить строку S в конец файла.

Листинг №24(k14_1.py)

```
S = input('S: ')
filename = input('File name: ')

f = open(filename, 'a')
f.write(S)
```

14.2 Техника работы с файлами

Задание 1. (Л.Б.)

При разработке курсовых проектов студентами 3 курса программистов ККМТ выбираются

различные направления, например, "графика", "базы данных"..

и предпочтения по языкам и средам "Си++", "Delphi"...

В каждой строке текстового файла хранятся следующие сведения о курсовых проектах:

Фамилия Имя Отчество; Группа; Год; Тема; Направления (список через запятую);

Языки и среды (список через запятую)

Например,

Иванов Иван Иванович;П1-21;2023;Картинки в базе;графика;Pascal,Lazarus

...

Программа должна читать входной файл и выдавать на экран ответы на вопросы

1. Какое направление встречается чаще всего

2. Какие языки и среды появились в дипломах в 2017 г.

Листинг №25(k14_2.py)

```
import sys

if len(sys.argv) > 1:
    filename = sys.argv[1]
else:
    filename = 'diploms.txt'

file = open(filename)

diploms = [x.split(';') for x in file]

print('Какое направление встречается чаще всего:')

направления = dict()
for x in diploms:
    напр = [a.strip() for a in x[4].split(',')]
    for н in напр:
        if н in направления:
            направления[н] += 1
        else:
            направления[н] = 1

print(sorted(направления.items())[0][0])

print('Какие языки и среды появились в дипломах в 2017:')

languages = set()

for x in [y for y in diploms if y[2] == '2017']:
    languages = languages | set([a.strip() for a in

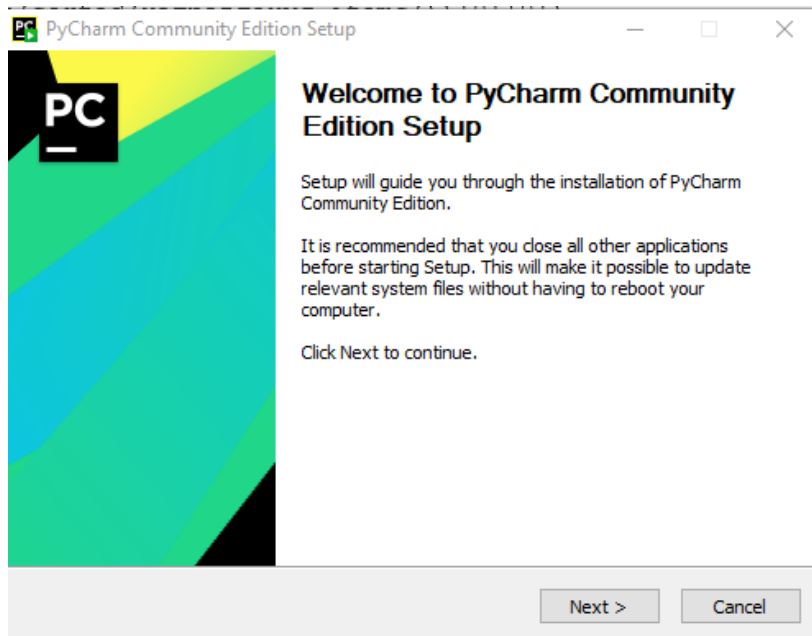
print(*languages, sep=', ')
```

Раздел 2 Техника решения задач с использованием библиотек.

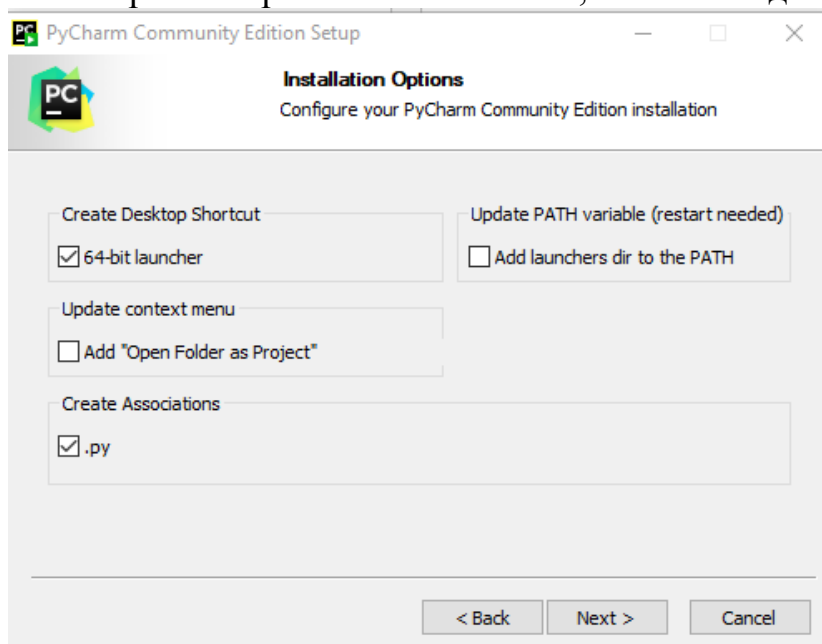
2.1 Установка и настройка среды JetBrains PyCharm

PyCharm - Интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

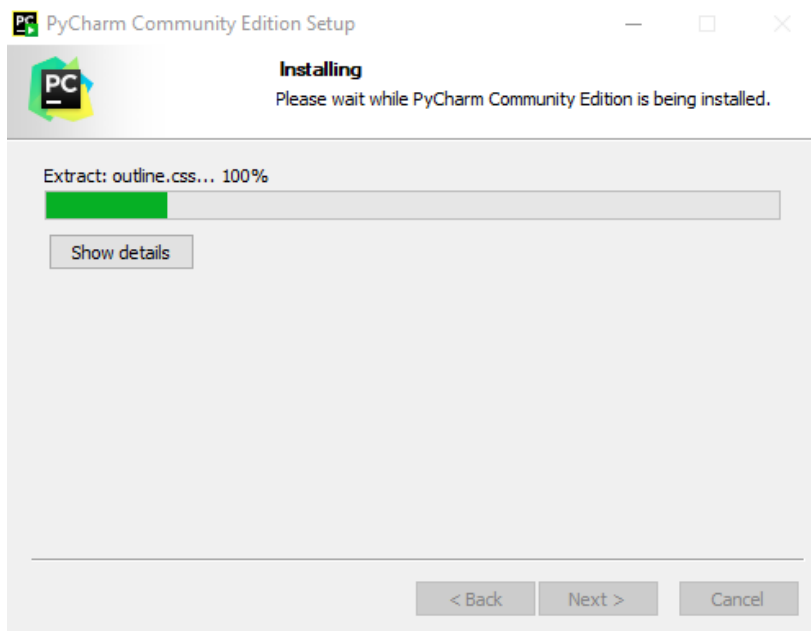
Чтобы скачать PyCharm нужно перейти на официальный сайт <https://www.jetbrains.com/pycharm/> , после этого мы выбираем версию для сообщества. Запускаем установщик.



Нас встречает приветственное окно, нажимаем далее.

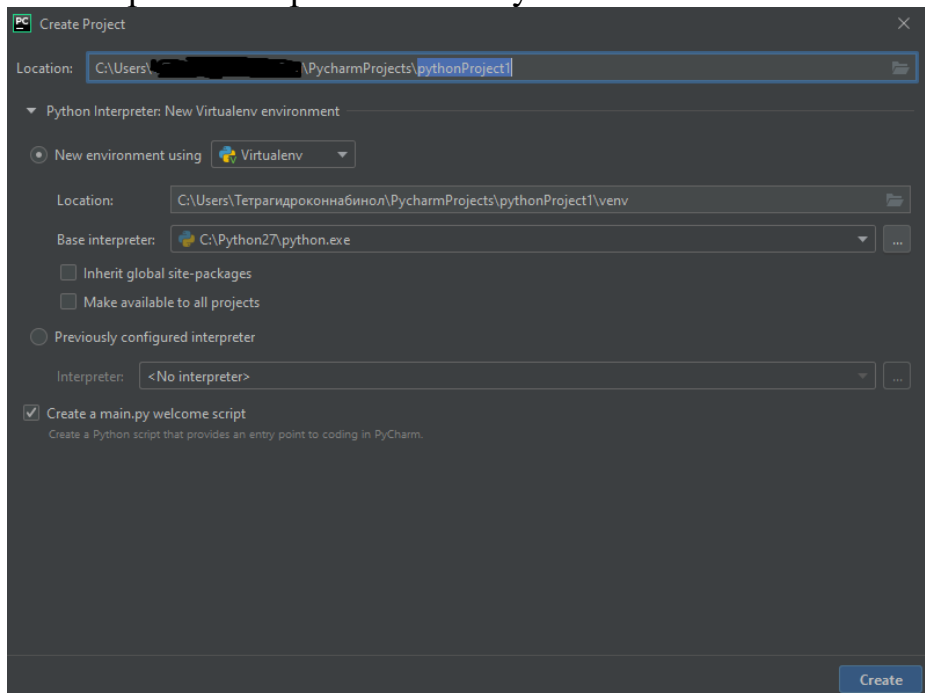


Выбираем настройки, которые будут удобны именно вам.

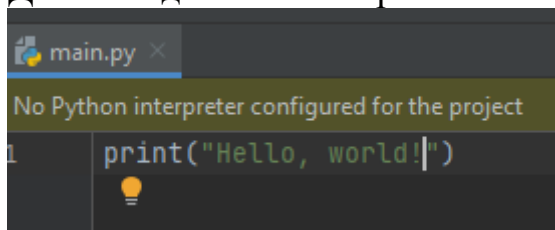


Устанавливаем.

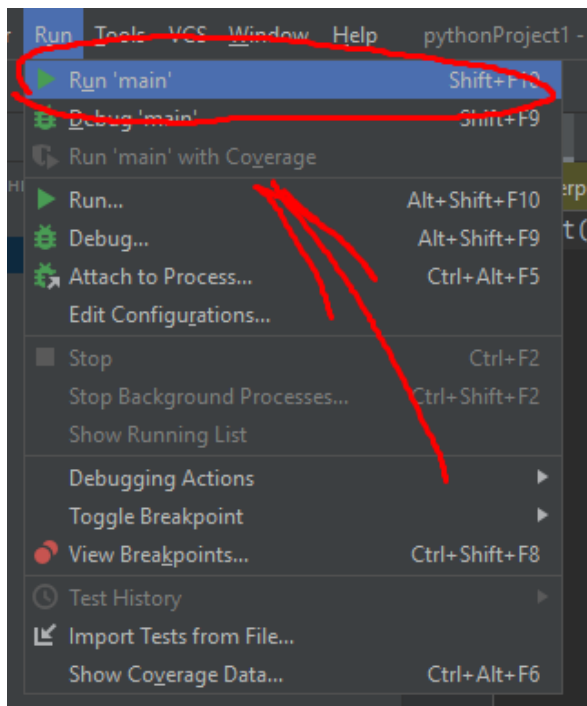
Нас встречает стартовое окно PyCharm.



Далее создаём новый проект.



Пишем программу.



Запускаем. В нижней части окна должен вывести результат в консоли.

2.2 Техника работы с базами данных

Импортируем модуль `sqlite3`. Также вместе с установкой `sqlite3` модуля нам потребуется установить и сам SQLite. Переходим на официальный сайт <https://www.sqlite.org/index.html>.

Листинг задания (Baza.py)

```
import sqlite3

class SQLighter:

    def __init__(self, imya):
        self.connection = sqlite3.connect(imya)
        self.cursor = self.connection.cursor()

    def create_table(self, baza):
        with self.connection:
            baza = "".join(chr for chr in baza if chr.isalnum())

        print(baza)
        if baza != "":
```

```

        self.cursor.execute(f"CREATE TABLE `{baza}` (\n
            `id` integer PRIMARY KEY,\n
            `marka` text, `nomer` text, `model` text)")
        self.save()

    def get_id(self, **kwargs):
        with self.connection:
            data = kwargs
            if kwargs.get('data') != None:
                data = kwargs.get('data')
            return self.cursor.execute("SELECT * FROM\n
`mashina` WHERE \n
            `marka` = ? AND `nomer` = ? AND `model` =\n
            ?",\n
            (data['marka'], data['nomer'], data['model'])).fetchall()[0][0]

    def add_mashina(self, **kwargs):
        with self.connection:
            data = kwargs
            if kwargs.get('data') != None:
                data = kwargs.get('data')

            self.cursor.execute("INSERT INTO `mashina` \n
            (`marka`, `nomer`, `model`) VALUES (?, ?, ?)", \n
            (data['marka'], data['nomer'], data['model']))
            self.save()

    def print_table(self):
        for s in self.cursor.execute("SELECT * FROM `mashina`"):
            print(s)

    def save(self):
        self.connection.commit()
        print(f"{self.cursor.rowcount} отредактировано строк")

    def close(self):
        self.connection.close()

bd = SQLighter('baza')
# bd.create_table("mashina")
bd.add_mashina(marka="BMW", nomer="12GHJ0", model='Bruh')
bd.print_table()
bd.close()

```

2.3 Техника работы с библиотекой tkinter

Tkinter – модуль для создания графического интерфейса.

Листинг 27 (Desktop.py)

```
from tkinter import *
from tkinter import messagebox
import time
#from threading import Thread

class Priloj():
    def __init__(self):
        self.root = Tk()
        self.root.title("Zdraste")
        width = 350
        height = 350
        x = 700
        y = 300
        self.root.geometry(f"{width}x{height}+{x}+{y}")
        self.root.iconbitmap("death.ico")
        self.label = Label(self.root, text="Программа по
улучшению комюнити", bg="#9932CC", relief=GROOVE,
wraplength=200, font="TimesNewRoman 17",
fg="#00FF00").pack(anchor = N)
        self.menu_buttons()

        self.button = Button(self.root, width=5, height=2,
bg="red", command=lambda:
self.root.config(bg="red")).pack(anchor=W)
        self.button = Button(self.root, width=5, height=2,
bg="blue", command=lambda:
self.root.config(bg="blue")).pack(anchor=W)
        self.button = Button(self.root, width=5, height=2,
bg="black", command=lambda:
self.root.config(bg="black")).pack(anchor=W)

    def svet(self):
        while True:
            self.root.config(bg = "purple")
            time.sleep(0.2)
            self.root.config(bg="blue")
            time.sleep(0.2)
            self.root.config(bg="red")
            time.sleep(0.2)
            self.root.config(bg="green")
            time.sleep(0.2)
            self.root.config(bg="yellow")
            time.sleep(0.2)

    def menu_buttons(self):
        self.button = Button(self.root, width=50, height=5,
text="Слабость", bg="#1E90FF", command=text_area).pack()
```

```

        self.button = Button(self.root, width=25, height=3,
text="Слабость", bg="#1E90FF", command=text_field).pack()
        self.button = Button(self.root, width=25, height=3,
text="Слабость", bg="#1E90FF", command=lambda:
messagebox.showwarning("Переведи", "YBUUTHS YT K>LB")).pack()

def text_area():
    def smile():
        label = Label(text, text="Молодец", bg="purple")
        text.window_create(INSERT, window=label)

    root=Tk()

    text = Text(root, width=100, height=100)
    text.pack()

    button = Button(text, text="Улыбнись :)", command=smile)
    button.pack()

    root.mainloop()

def text_field():
    def insertText():
        s = "Спасибо за урок Роман"
        text.insert(1.0, s)

    def getText():
        s = text.get(1.0, END)
        label['text'] = s

    def deleteText():
        text.delete(1.0, END)

    root = Tk()

    text = Text(root, width=30, height=5)
    text.pack()

    frame = Frame(root)
    frame.pack()

    b_insert = Button(frame, text="Вставить",
command=insertText)
    b_insert.pack(side=LEFT)

    b_get = Button(frame, text="Получить", command=getText)
    b_get.pack(side=LEFT)

    b_delete = Button(frame, text="Удалить", command=deleteText)
    b_delete.pack(side=LEFT)

```



```
label = Label(root)
label.pack()

root.mainloop()
```

```
Priloj()
mainloop()
```

После выполнение данной программы мы получим определённые данные в таблицу.

	id	marka	nomer	model	
	Фи...	Фил...	Филь...	Фил...	
1	1	BMW	12GHJ0	Bruh	
2	2	BMW	12GHJ0	Bruh	
3	3	BMW	12GHJ0	Bruh	

2.4 Техника работы с библиотекой NumPy

NumPy — библиотека с открытым исходным кодом для языка программирования Python. Возможности: поддержка многомерных массивов; поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами.

Демонстрация некоторых функций NumPy

Преобразование списка в массив:

```
A = np.array([[1, 2, 3], [4, 5, 6]])
```

Копирование массива:

```
B = A.copy()
```

Создание нулевого массива:

```
A = np.zeros((2, 3))
```

Просмотр всех доступных типов:

```
np.sctypes
```

Действия с массивами

```
import numpy as np
```

```
arr = (100 - 0) * np.random.random_sample((5,)) - 5  
arr = arr.astype(np.int64)
```

```
print(arr)
```

```
print(arr.sum()) # Сумма массивов  
print(arr.min()) # Минимальный массив  
print(arr.max()) # Максимальный массив  
print(arr.mean()) # Средний массив  
print(arr.std())  
print(np.median(arr)) # Медиана
```

Работа с элементами вектора

```
import numpy as np
```

```
V = np.array((1, 2, 3, 4))  
print(V[0])  
print(V[-1])  
print(V[1:-2])  
print(V[:2])
```

Расширенное индексирование

```
import numpy as np  
x = np.array([[ 0,  1,  2],[ 3,  4,  5],[ 6,  7,  8],[ 9, 10,  
11]])
```

```
print 'Our array is:'  
print x  
print '\n'
```

```
rows = np.array([[0,0],[3,3]])
```

```
cols = np.array([[0,2],[0,2]])
y = x[rows,cols]

print 'The corner elements of this array are:'
print y
```

2.5 Техника работы с библиотекой Matplotlib

Matplotlib — библиотека на языке программирования Python для визуализации данных двумерной графикой. Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях. Matplotlib написан и поддерживался в основном Джоном Хантером и распространяется на условиях BSD-подобной лицензии.

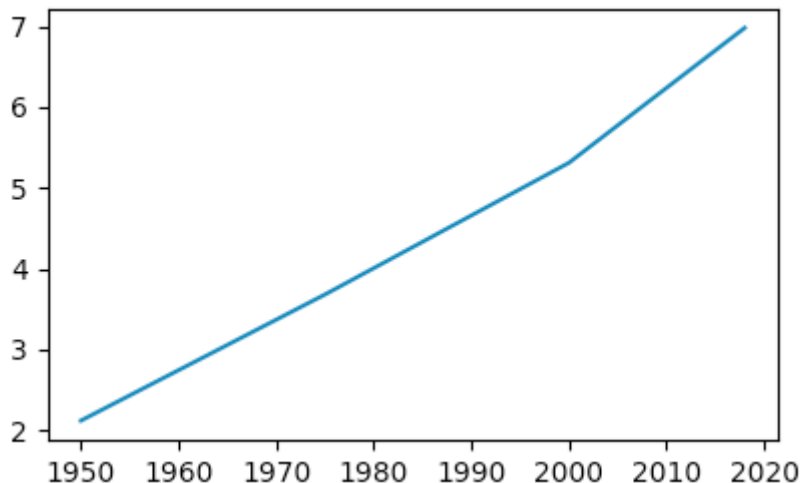
Линейные графики

Листинг 28(LineiGr.py)

```
import matplotlib.pyplot as plt

year = [1950, 1975, 2000, 2018]
population = [2.12, 3.681, 5.312, 6.981]

plt.plot(year, population)
plt.show()
```

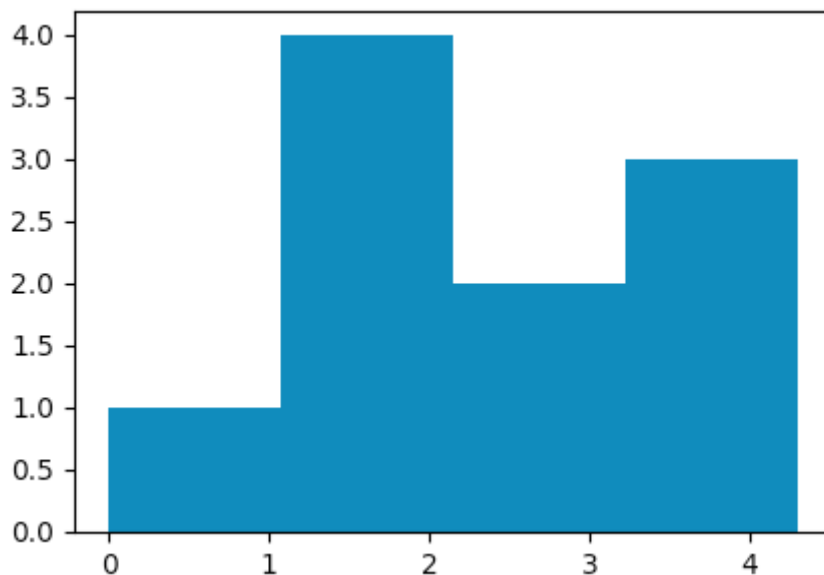


Гистограмма

Листинг 29(Gistogr.py)

```
import matplotlib.pyplot as plt

values = [0, 1.2, 1.3, 1.9, 4.3, 2.5, 2.7, 4.3, 1.3, 3.9]
plt.hist(values, bins = 4)
plt.show()
```



Кривые

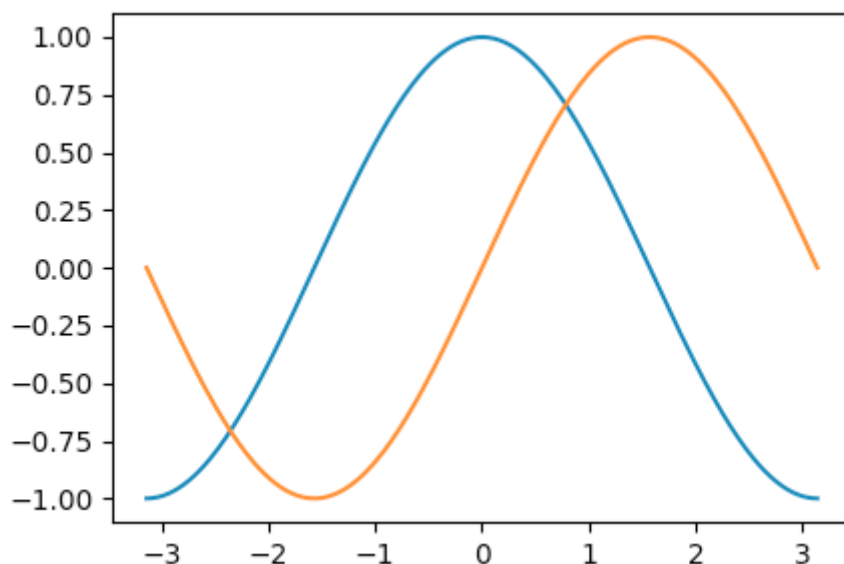
Листинг 30(Krivie.py)

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

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
cos, sin = np.cos(X), np.sin(X)

plt.plot(X, cos)
plt.plot(X, sin)

plt.show()
```



2.6 Элементы работы с библиотекой PyQt

PyQt — набор расширений графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python. PyQt

разработан британской компанией Riverbank Computing. PyQt работает на всех платформах, поддерживаемых Qt: Linux и другие UNIX-подобные ОС, Mac OS X и Windows.

Листинг 31(PyQT.py)

```
import sys

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QVBoxLayout
from PyQt5.QtWidgets import QWidget

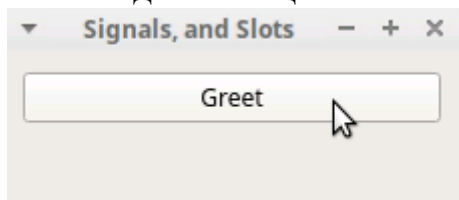
def greeting():
    if msg.text():
        msg.setText("")
    else:
        msg.setText("Hello World!")

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('Signals and slots')
layout = QVBoxLayout()

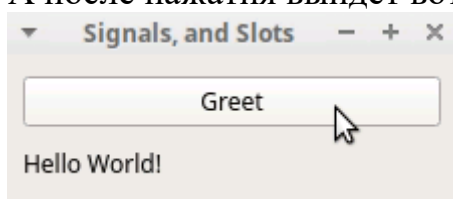
btn = QPushButton('Greet')
btn.clicked.connect(greeting)

layout.addWidget(btn)
msg = QLabel('')
layout.addWidget(msg)
window.setLayout(layout)
window.show()
sys.exit(app.exec_())
```

После запуска данной программы выйдет окно с кнопкой, после нажатия на неё выйдет сообщение.



А после нажатия выйдет вот это



2.7 Элементы работы с библиотекой PyGame

Pygame - это библиотека для разработки игр на языке программирования Python.

Листинг 28(MatveyGame)

```
import pygame

pygame.init()

display_width = 800
display_height = 800

display =
pygame.display.set_mode((display_height,display_width))

hero_x = 100
hero_y = 100
hero_width = 20
hero_height = 25
speed = 2

pygame.display.set_caption('ABOBA')

def Update():
    global hero_x,hero_y
    game = True
    while game:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()

        keys = pygame.key.get_pressed()
        if keys[pygame.K_RIGHT] and hero_x < 780:
            hero_x += speed
        if keys[pygame.K_DOWN] and hero_y < 780:
            hero_y += speed
        if keys[pygame.K_LEFT] and hero_x > 1:
            hero_x -= speed
        if keys[pygame.K_UP] and hero_y > 1:
            hero_y -= speed
        if (hero_x > 699 and hero_y > 699):
            pygame.quit()
            quit()
        display.fill((247,164,127))
        pygame.draw.rect(display, (255, 0, 0), (700, 700, 100,
100))
```

```
pygame.draw.rect(display, (126,165,243), (hero_x,hero_y,hero_height,hero_width))
pygame.display.update()
Update()
```



В данном приложении, если синим квадратом зайти на территорию красного квадрата, то произойдёт выход из игры.

Раздел 3 Разработка проекта с графическим интерфейсом.

3.1 Изучение входной и выходной документации.

На вход подаётся Модель, Компания, Цена изделия, которые мы пытаемся продать. Удаление данных производится на сайте на основе порядкового номера.

Входные данные:

1. Данные о изделии
 - 1.1. Модель
 - 1.2. Компания
 - 1.3. Цена
2. Добавление новых данных в документ
 - 2.1. Модель
 - 2.2. Компания
 - 2.3. Цена

Выходные данные:

1. Демонстрация данных
 - 1.1. Модель
 - 1.2. Компания
 - 1.3. Цена

3.2 Разработка требований к проекту. Построение диаграммы использования.

- 1) Разработать хост на котором будут держаться данные.
- 2) Использовать базу данных (MySQL).
- 3) Реализовать добавление / удаление / редактирование записей.
- 4) Реализовать вывод содержимого базы данных в табличном виде.

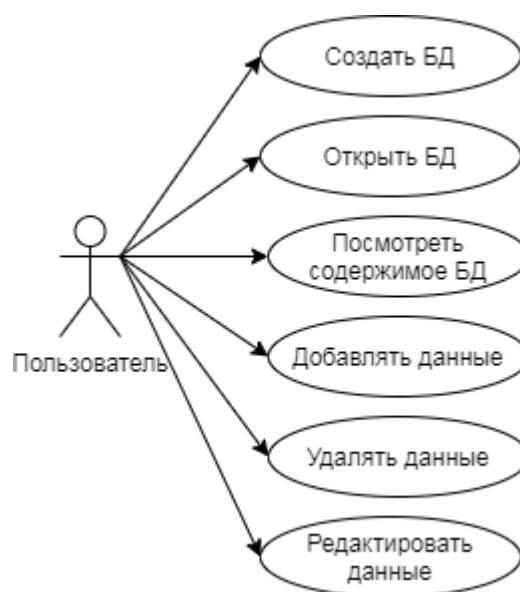


Рис 1 Диаграмма использования.

3.3 Разработка сценария проекта.

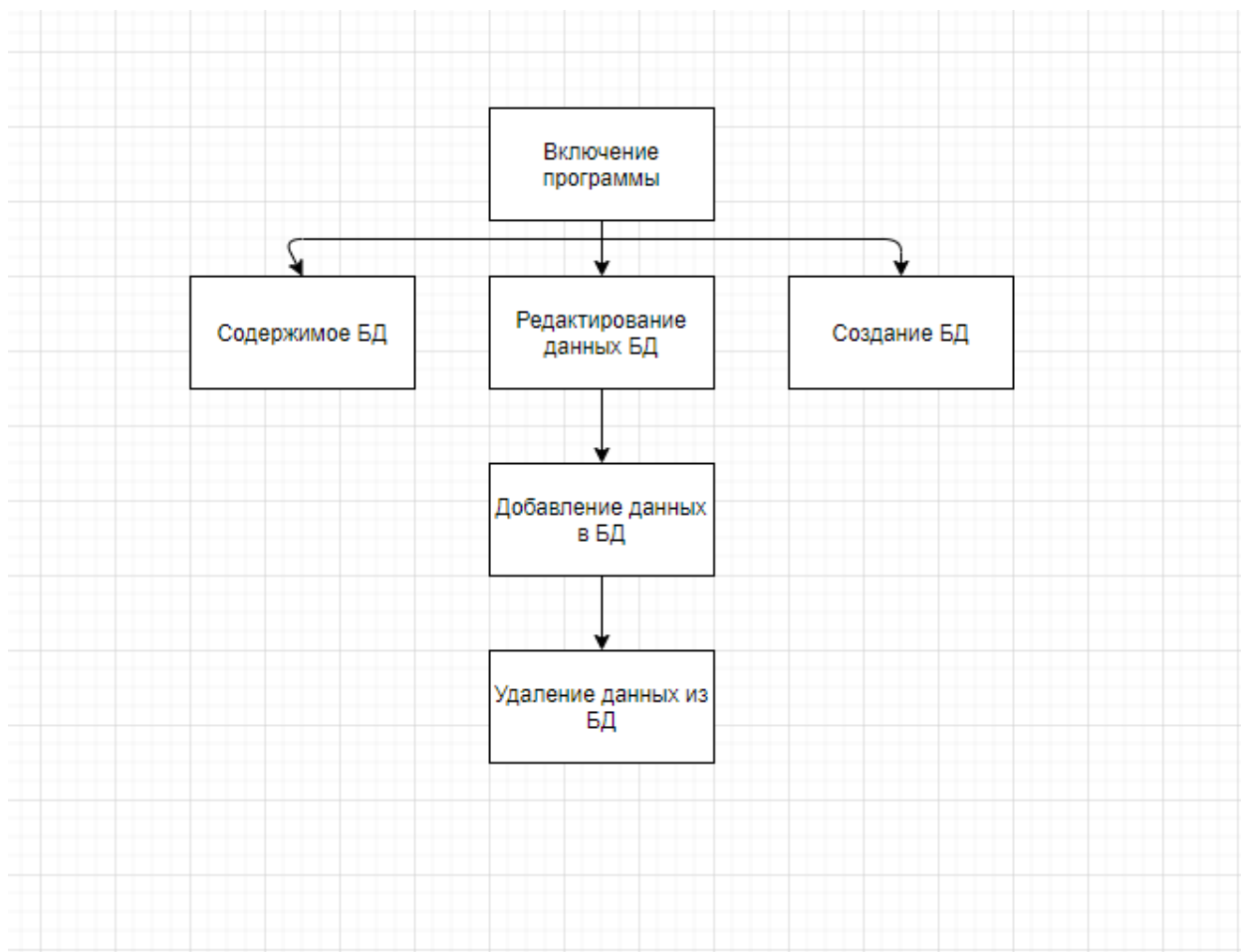


Рис 2 Сценарий программы.

3.4 Разработка базы данных.

В базу данных этого проекта входит одна таблица, в которую входят 4 столбца: Id, Model, Company и Price.

The screenshot shows the 'products' table structure in SQL Server Enterprise Manager. The table is located in the 'productdb' schema. The columns are: Id (INT, PK, NN), Model (VARCHAR(45)), Company (VARCHAR(45)), and Price (INT). The 'Company' column is currently selected, and its properties are shown in the bottom pane.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Model	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Company	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Price	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Company Data Type: VARCHAR(45)
Charset/Collation: Default Charset Default Collation: Default Collation Default:

Рис.3 Структура базы данных.

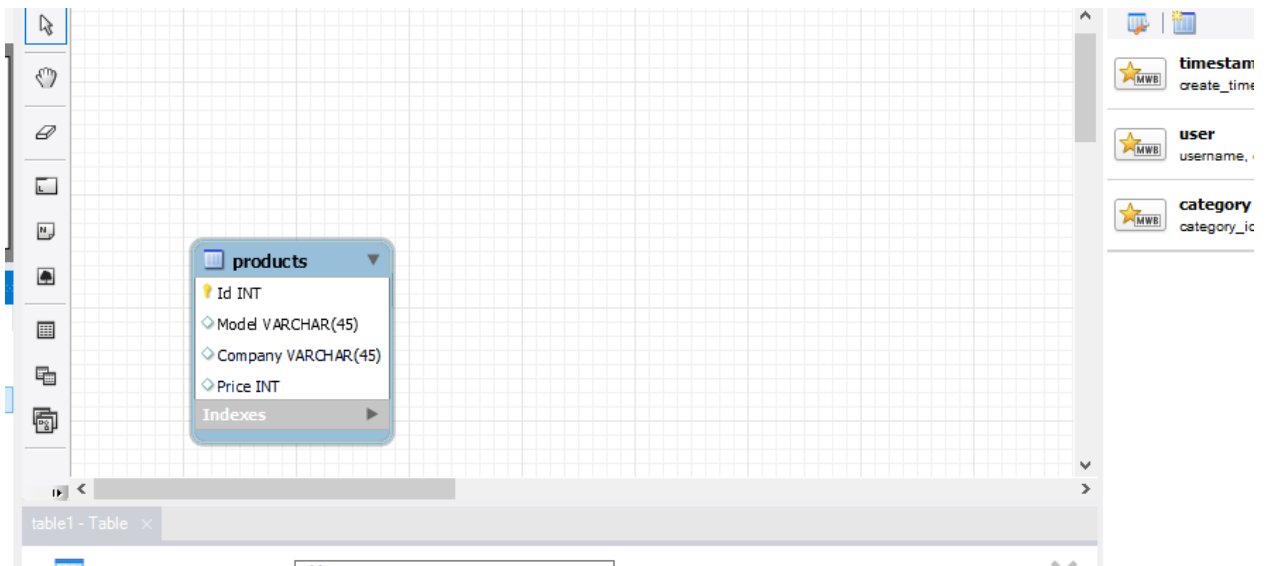


Рис.4 Заполнение базы данных.

3.5 Разработка главного модуля.

Перед работой с приложением нам надо установить (подготовить) драйвер.

Приложения: Main.go и index.html

```
package main

import (
    "database/sql"
    "fmt"
    "html/template"
    "log"
    "net/http"

    _ "github.com/go-sql-driver/mysql"
    "github.com/gorilla/mux"
)

type Product struct {
    Id      int
    Model   string
    Company string
    Price   int
}

var database *sql.DB

func DeleteHandler(w http.ResponseWriter, r *http.Request) {
    vars := mux.Vars(r)
    id := vars["id"]
}
```

```

_, err := database.Exec("delete from productdb.Products where id = ?", id)
if err != nil {
    log.Println(err)
}

http.Redirect(w, r, "/", 301)
}

func EditPage(w http.ResponseWriter, r *http.Request) {
    vars := mux.Vars(r)
    id := vars["id"]

    row := database.QueryRow("select * from productdb.Products where id = ?", id)
    prod := Product{}
    err := row.Scan(&prod.Id, &prod.Model, &prod.Company, &prod.Price)
    if err != nil {
        log.Println(err)
        http.Error(w, http.StatusText(404), http.StatusNotFound)
    } else {
        tpl, _ := template.ParseFiles("templates/edit.html")
        tpl.Execute(w, prod)
    }
}

func EditHandler(w http.ResponseWriter, r *http.Request) {
    err := r.ParseForm()
    if err != nil {
        log.Println(err)
    }
    id := r.FormValue("id")
    model := r.FormValue("model")
    company := r.FormValue("company")
    price := r.FormValue("price")

    _, err = database.Exec("update productdb.Products set model=?, company=?, price=? where id = ?",
        model, company, price, id)

    if err != nil {
        log.Println(err)
    }
    http.Redirect(w, r, "/", 301)
}

func CreateHandler(w http.ResponseWriter, r *http.Request) {
    if r.Method == "POST" {

        err := r.ParseForm()
        if err != nil {
            log.Println(err)

```

```

    }
    model := r.FormValue("model")
    company := r.FormValue("company")
    price := r.FormValue("price")

    _, err = database.Exec("insert into productdb.Products (
model, company, price) values (?, ?, ?)",
        model, company, price)

    if err != nil {
        log.Println(err)
    }
    http.Redirect(w, r, "/", 301)
} else {
    http.ServeFile(w, r, "templates/create.html")
}
}

func IndexHandler(w http.ResponseWriter, r *http.Request) {

    rows, err := database.Query("select * from productdb.Product
s")
    if err != nil {
        log.Println(err)
    }
    defer rows.Close()
    products := []Product{}

    for rows.Next() {
        p := Product{}
        err := rows.Scan(&p.Id, &p.Model, &p.Company, &p.Price)
        if err != nil {
            fmt.Println(err)
            continue
        }
        products = append(products, p)
    }

    tpl, _ := template.ParseFiles("templates/index.html")
    tpl.Execute(w, products)
}

func main() {

    db, err := sql.Open("mysql", "root:password@/productdb")

    if err != nil {
        log.Println(err)
    }
    database = db
    defer db.Close()

    router := mux.NewRouter()

```

```

    router.HandleFunc("/", IndexHandler)
    router.HandleFunc("/create", CreateHandler)
    router.HandleFunc("/edit/{id:[0-9]+}", EditPage).Methods("GET")
    router.HandleFunc("/edit/{id:[0-9]+}", EditHandler).Methods("POST")
    router.HandleFunc("/delete/{id:[0-9]+}", DeleteHandler)

    http.Handle("/", router)

    fmt.Println("Server is listening...")
    http.ListenAndServe(":8181", nil)
}

```

А также файл HTML который даст нам простое оформление страницы для БазыДанных.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Products</title>
  </head>
  <body>
    <h2>Список товаров</h2>
    <p><a href="/create">Добавить</a></p>
    <table>
      <thead><th>Id</th><th>Model</th><th>Company</th><th>
Price</th><th></th></thead>
      {{range . }}
      <tr>
        <td>{{.Id}}</td>
        <td>{{.Model}}</td>
        <td>{{.Company}}</td>
        <td>{{.Price}}</td>
        <td><a href="/edit/{{.Id}}">Изменить</a> |
          <a href="/delete/{{.Id}}">Удалить</a>
        </td>
      </tr>
      {{end}}
    </table>
  </body>
</html>

```

3.6 Тестирование и отладка

В ходе написания проекта при запуске программы были получены ошибки:

```

create : Имя "create" не распознано как имя командлета, функции, файла сценария или выполняемой программы. Проверьте правильность написания имени, а также наличие и правильность пути, после чего повторите попытку.
строка:1 знак:1
+ create database productdb;
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (create:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

```

Рис.5 Сообщение об ошибке

При поиске и исправления ошибок программа запустилась.

Финальный вид программы

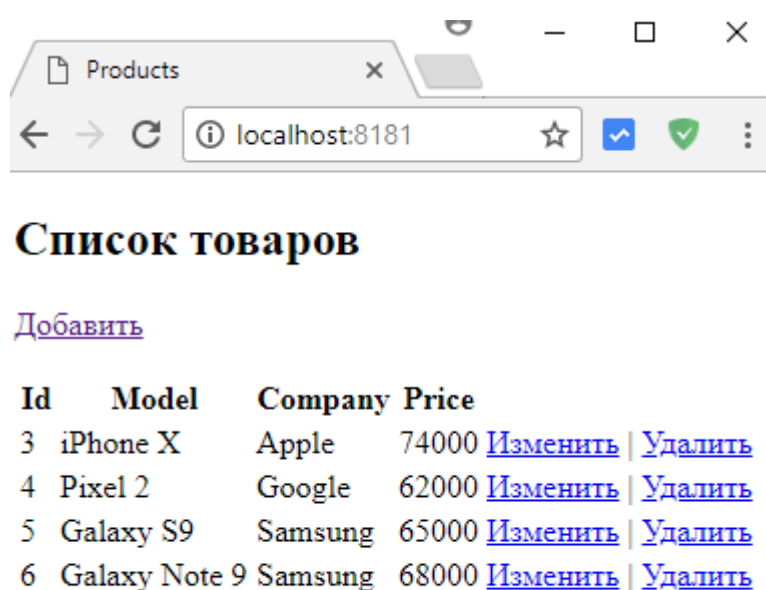


Рис.6 Финальный вид программы

3.7 Разработка документации к проекту.

Назначение программы.

Функциональное назначение.

Данная программа предназначена для добавления, удаления, редактирования и просмотра записей.. При этом программа взаимодействует с базой данных.

Эксплуатационное назначение программы.

С помощью данной системы пользователь может отслеживать данные добавлять или удалять их.

Состав функций:

- 1) Проверка попадания введенной даты в промежуток;
- 2) Запись информации о посещениях;
- 3) Добавление информации о появлении студента;
- 4) Вывод списка опоздавших;
- 5) Поиск студента с наибольшим количеством опозданий;
- 6) Поиск студента с наибольшим временем опозданий.

Условия выполнения программы

Минимальные требования для аппаратных средств.

Для работы программы требуется:

- ОС: Windows XP, 7, Vista, 8, 8.1, 10
- Процессор: Intel Celeron 1800 MHz
- Оперативная память: 256 MB ОЗУ
- Видеокарта: Intel HD Graphics
- DirectX: Версии 9.0
- Место на диске: 15 MB

Минимальный состав программных средств

Системные программные средства, используемые специальным программным обеспечением «Муром», должны быть представлены локализованной версией операционной системы Windows XP, Windows Vista или Windows 7

Требования к персоналу (пользователю)

Пользователь программы (оператор) должен обладать начальными навыками работы с компьютером.

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.	
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.	