Министерство образования Республики Беларусь

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Отчёт

по лабораторной работе №1

«Введение в программирование на скриптовых языках»

Вариант 13

|  |  |
| --- | --- |
| Выполнил: | студент группы 273602  Копнева Злата |
| Проверил: | Булыга Дмитрий Игоревич |

Цель: приобрести начальные навыки работы с языком программирования Python, методами работы с текстом и со строками.

Задачи: написать код программ.

Язык Python является одним из самых простых в изучении и самых приятных в использовании из языков программирования, получивших широкое распространение. Программный код на языке Python легко читать и писать, и, будучи лаконичным, он не выглядит загадочным. Python - очень выразительный язык, позволяющий уместить приложение в меньшее количество строк, чем на это потребовалось бы в других языках, таких как C++ или Java

Ряд особенностей выделяют Python среди прочих языков программирования.

1. Простой и легкий в освоении.

Python – простой и минималистичный язык, что дает возможность сосредоточиться на решении задачи, а не на самом языке.

1. Свободный и открытый.

Python – [свободное и открытое программное обеспечение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%B8_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (Free/Libré and Open Source Sotware - FLOSS) [9](https://www.yuripetrov.ru/edu/python/ch_02_01.html#id38). Общий обзор философии свободного ПО приведен на Видео 2.1.3 (англ.).

В основе свободного ПО лежит идея сообщества, которое делится своими знаниями; само движение руководствуется четырьмя принципами свободы:

* программу можно свободно использовать с любой целью («нулевая свобода»);
* можно изучать, как программа работает, и адаптировать её для своих целей («первая свобода») - условием этого является доступность исходного текста программы;
* можно свободно распространять копии программы — в помощь товарищу («вторая свобода»).
* программу можно свободно улучшать и публиковать свою улучшенную версию — с тем, чтобы принести пользу всему сообществу («третья свобода») - условием этой третьей свободы является доступность исходного текста программы и возможность внесения в него модификаций и исправлений.

1. Расширяемый и встраиваемый.

Python можно встраивать в программы на C/C++, чтобы предоставлять возможности написания сценариев их пользователям или для ускорения работы программы.

1. «Заряженный».

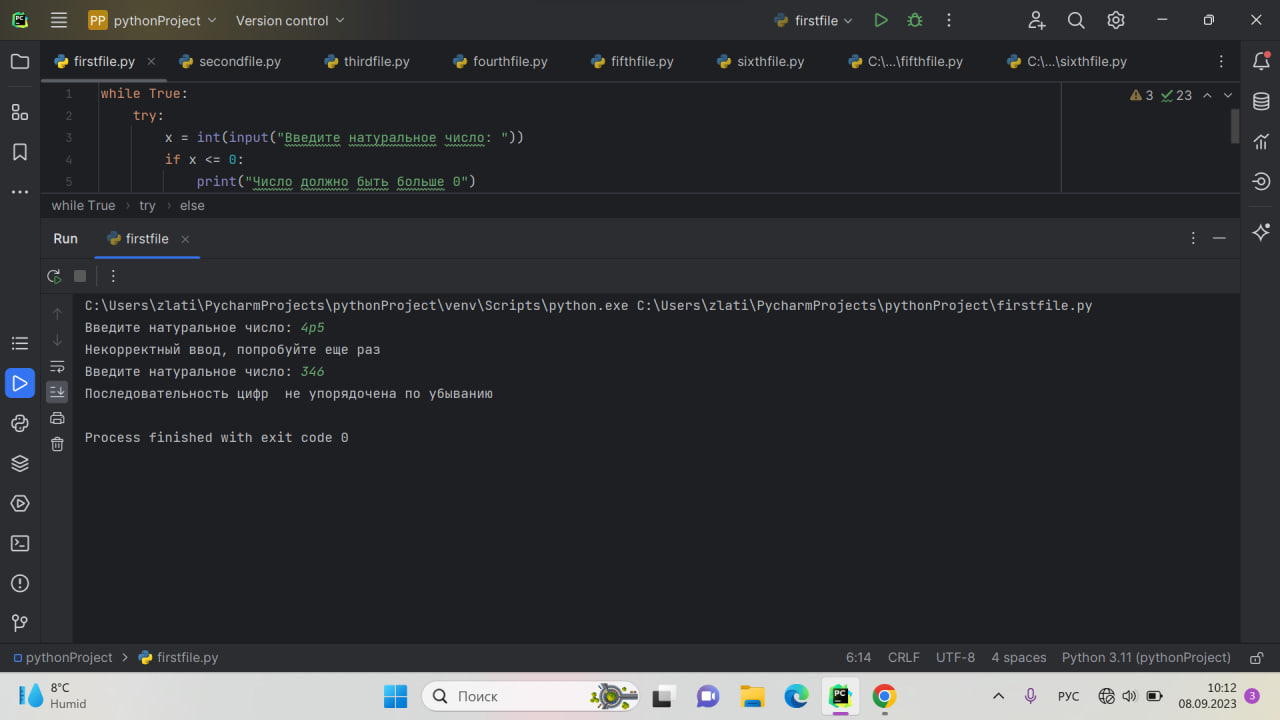
Python поставляется по принципу «все включено» (*англ.* Batteries Included) и имеет обширные возможности в стандартной библиотеке в дополнение к встроенным возможностям языка.

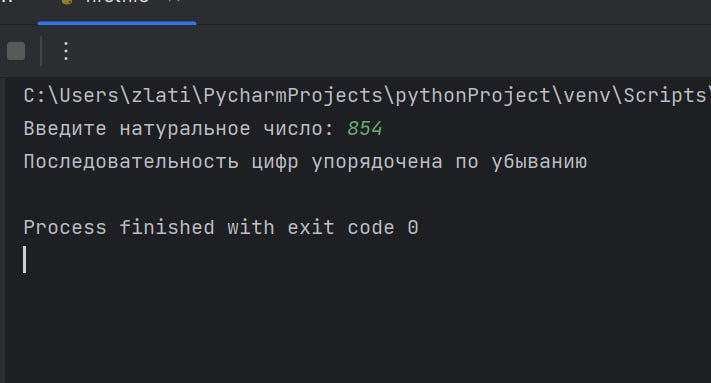
Стандартная библиотека позволяет решать различные задачи, связанные с использованием регулярных выражений, генерацией документации, проверкой блоков кода, распараллеливанием процессов, базами данных, сетью Интернет, электронной почтой, криптографией, GUI (графическим интерфейсом пользователя) и другим системно-зависимым функционалом.

Индивидуальное задание

1. Дано натуральное число. Напишите программу, которая определяет, является ли последовательность его цифр при просмотре справа налево упорядоченной по убыванию.

while True:  
 try:  
 x = int(input("Введите натуральное число: "))  
 if x <= 0:  
 print("Число должно быть больше 0")  
 else:  
 break  
 except ValueError:  
 print("Некорректный ввод, попробуйте еще раз")  
result = 'Последовательность цифр упорядочена по убыванию'  
while x // 10 != 0:  
 x, n = divmod(x, 10) # Немного сократим  
 if n >= x % 10:  
 result = 'Последовательность цифр не упорядочена по убыванию'  
 break # Нет смысла продолжать проверку  
print(result)



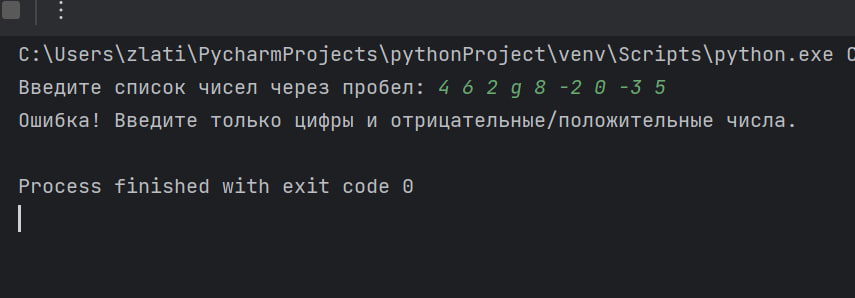


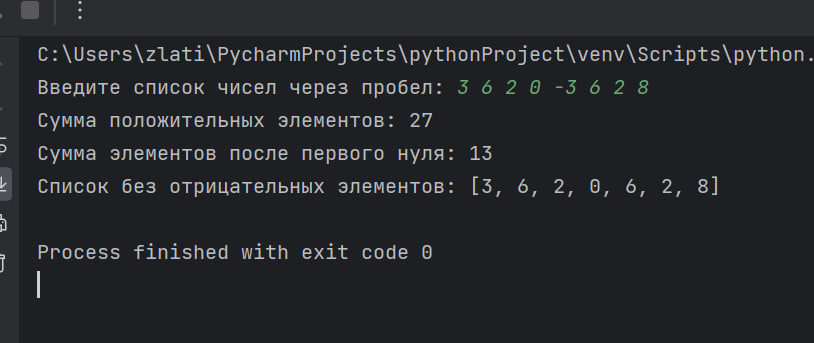
2. Посчитать, сколько пар (стоят рядом) верхнего и нижнего регистра находится в веденном с клавиатуры слове. (Пример HjkLM- 1 пара нижнего, 1 пара верхнего), а также сколько гласных букв в слове

def count\_letter\_pairs\_and\_vowels(word):  
 upper\_pairs = 0  
 lower\_pairs = 0  
 vowels = 0  
  
 for i in range(len(word) - 1):  
 current\_char = word[i]  
 next\_char = word[i + 1]  
  
 if current\_char.isalpha() and next\_char.isalpha():  
 if (current\_char.isupper() and next\_char.isupper()) or (current\_char.islower() and next\_char.islower()):  
 if current\_char.isupper():  
 upper\_pairs += 1  
 else:  
 lower\_pairs += 1  
  
 if current\_char.lower() in ['a', 'e', 'i', 'o', 'u']:  
 vowels += 1  
  
 return upper\_pairs, lower\_pairs, vowels  
  
# Пример использования  
word = input("Введите слово: ")  
upper\_pairs, lower\_pairs, vowels = count\_letter\_pairs\_and\_vowels(word)  
  
print(f"Пары верхнего регистра: {upper\_pairs}")  
print(f"Пары нижнего регистра: {lower\_pairs}")  
print(f"Гласных букв: {vowels}")

3. Найдите сумму положительных элементов списка. Найдите сумму элементов списка после первого нуля. Если нулевых элементов нет в списке, то выведите «Сумму посчитать нельзя». Удалить из списка все отрицательные элементы.

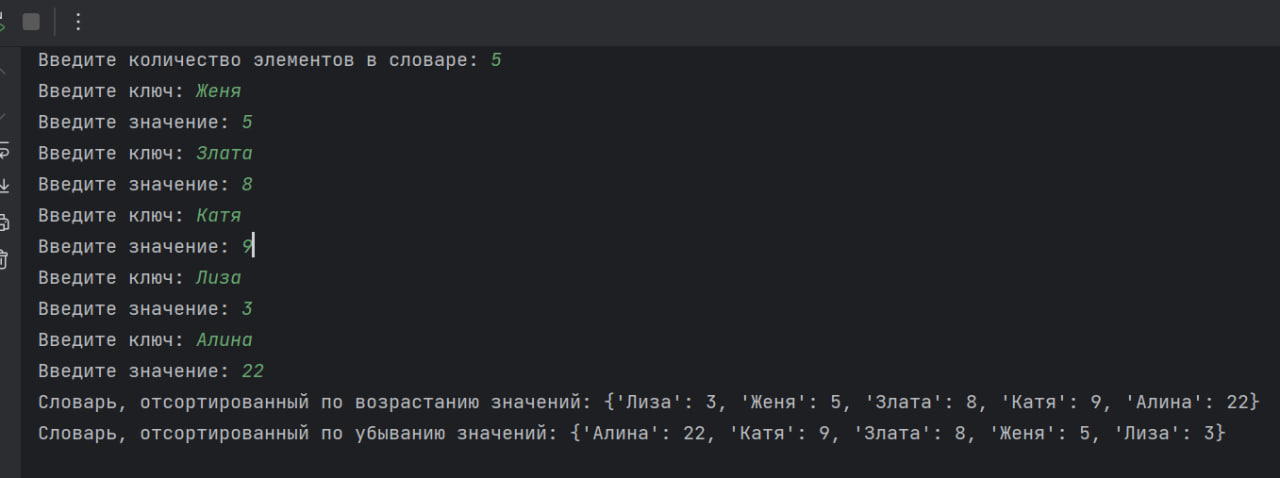
lst = input("Введите список чисел через пробел: ").split()  
  
# Проверим, что введены только цифры  
allowed\_chars = ['-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9']  
for num in lst:  
 for char in num:  
 if char not in allowed\_chars:  
 print("Ошибка! Введите только цифры и отрицательные/положительные числа.")  
 exit()  
  
lst = [int(i) for i in lst]  
  
# Найдем сумму положительных элементов списка  
sum\_pos = 0  
for i in lst:  
 if i > 0:  
 sum\_pos += i  
print("Сумма положительных элементов:", sum\_pos)  
  
# Найдем сумму элементов списка после первого нуля  
sum\_after\_zero = 0  
flag = False  
for i in lst:  
 if i == 0:  
 flag = True  
 elif flag:  
 sum\_after\_zero += i  
if flag:  
 print("Сумма элементов после первого нуля:", sum\_after\_zero)  
else:  
 print("Сумму посчитать нельзя")  
  
# Удалим из списка все отрицательные элементы  
lst = [i for i in lst if i >= 0]  
print("Список без отрицательных элементов:", lst)





4. Отсортируйте словарь по значению в порядке возрастания и убывания.

# Ввод ключей и значений с клавиатуры  
d = {}  
n = int(input("Введите количество элементов в словаре: "))  
for i in range(n):  
 key = input("Введите ключ: ")  
 value = int(input("Введите значение: "))  
 d[key] = value  
  
# Сортировка по возрастанию значений  
sorted\_d\_asc = sorted(d.items(), key=lambda x: x[1])  
print("Словарь, отсортированный по возрастанию значений:", dict(sorted\_d\_asc))  
  
# Сортировка по убыванию значений  
sorted\_d\_desc = sorted(d.items(), key=lambda x: x[1], reverse=True)  
print("Словарь, отсортированный по убыванию значений:", dict(sorted\_d\_desc))



5. Реализуйте программу «Кондитерская», которая будет включать в себя шесть пунктов меню. У вас есть словарь, где ключ – название продукции (торт, пирожное, маффин и т.д.). Значение – список, который содержит состав, цену (за 100гр) и кол-во (в граммах).

1. Просмотр описания: название – описание

2. Просмотр цены: название – цена.

3. Просмотр количества: название – количество.

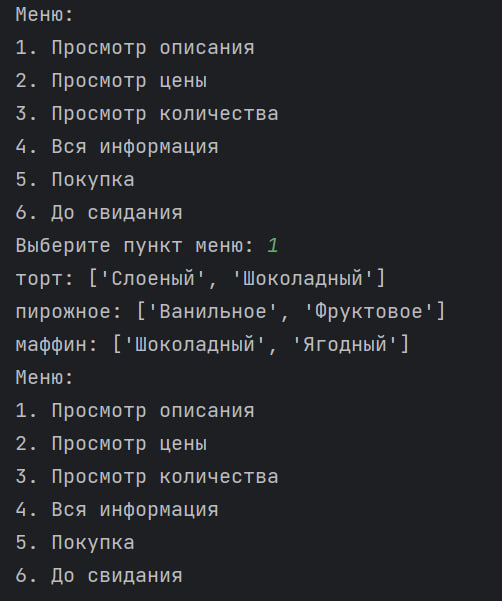
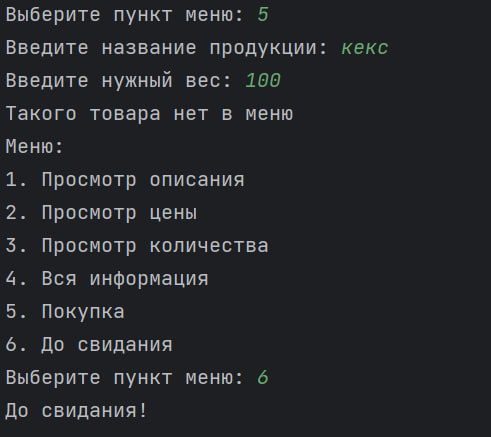
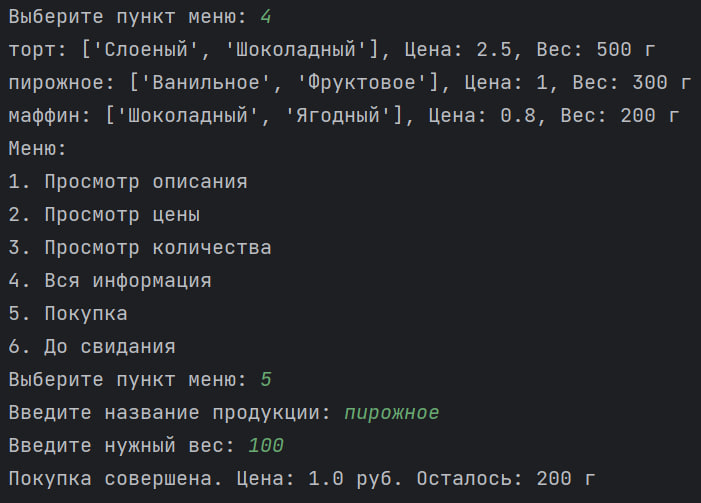
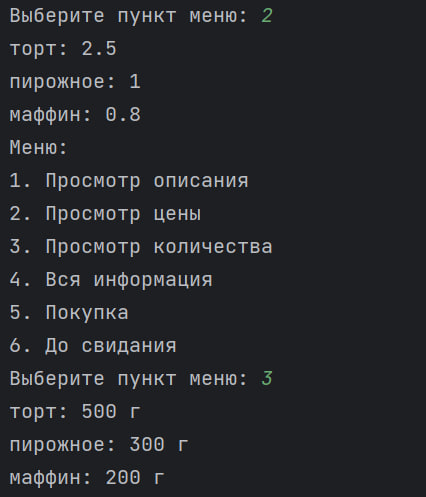
4. Всю информацию.

5. Покупка

В пункте «Покупка» необходимо совершить покупку, с клавиатуры вводите название продукции и его кол-во, n – выход из программы. Посчитать цену выбранных товаров и сколько товаров осталось в изначальном списке

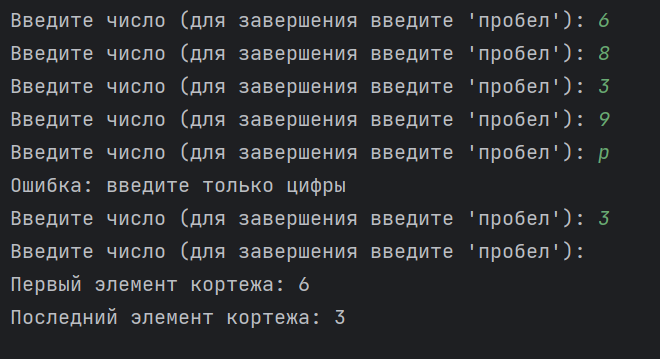
6. До свидания

menu = {  
 "торт": [["Слоеный", "Шоколадный"], 2.5, 500],  
 "пирожное": [["Ванильное", "Фруктовое"], 1, 300],  
 "маффин": [["Шоколадный", "Ягодный"], 0.8, 200]  
}  
  
def display\_description():  
 for item, details in menu.items():  
 print(f"{item}: {details[0]}")  
  
def display\_price():  
 for item, details in menu.items():  
 print(f"{item}: {details[1]}")  
  
def display\_quantity():  
 for item, details in menu.items():  
 print(f"{item}: {details[2]} г")  
  
def display\_all\_information():  
 for item, details in menu.items():  
 print(f"{item}: {details[0]}, Цена: {details[1]}, Вес: {details[2]} г")  
  
def buy\_item(item, quantity):  
 if item in menu:  
 if menu[item][2] >= quantity:  
 price = menu[item][1] \* (quantity / 100)  
 menu[item][2] -= quantity  
 print(f"Покупка совершена. Цена: {price} руб. Осталось: {menu[item][2]} г")  
 else:  
 print("Недостаточно товара на складе")  
 else:  
 print("Такого товара нет в меню")  
  
while True:  
 print("Меню:")  
 print("1. Просмотр описания")  
 print("2. Просмотр цены")  
 print("3. Просмотр количества")  
 print("4. Вся информация")  
 print("5. Покупка")  
 print("6. До свидания")  
  
 choice = input("Выберите пункт меню: ")  
  
 if choice == "1":  
 display\_description()  
 elif choice == "2":  
 display\_price()  
 elif choice == "3":  
 display\_quantity()  
 elif choice == "4":  
 display\_all\_information()  
 elif choice == "5":  
 item = input("Введите название продукции: ")  
 if item == 'п':  
 break  
 quantity = int(input("Введите нужный вес: "))  
 buy\_item(item, quantity)  
 elif choice == "6":  
 break  
 else:  
 print("Неверный пункт меню")  
  
print("До свидания!")

6. Дан кортеж целых чисел. Вывести на экран первый и последний элемент кортежа.

numbers = ()  
while True:  
 user\_input = input("Введите число (для завершения введите 'пробел'): ")  
 if user\_input == ' ':  
 break  
 elif not user\_input.isdigit():  
 print("Ошибка: введите только цифры")  
 else:  
 numbers += (int(user\_input),)  
  
if len(numbers) > 0:  
 print("Первый элемент кортежа:", numbers[0])  
 print("Последний элемент кортежа:", numbers[-1])  
else:  
 print("Кортеж пустой")



Вывод: приобрела начальные навыки работы с языком программирования Python, методами работы с текстом и со строками. Написала код программ с использованием скриптового языка программирования.