Упражнения по программированию главы 8

```
# coding: utf-8
```

Упражнение по программированию 8.1. Инициалы

```
def main():
    # Получить от пользователя входные данные
    full_name = input ('Введите свое полное имя: ')

    # Разбить по пробелами
    name = full_name.split()

    # Первый символ каждого имени является инициалом
    for string in name:
        print(string[0].upper(), sep='', end='')
        print('.', sep=' ', end='')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.2. Сумма цифр в строке

```
def main():
    # Получить от пользователя строку чисел.
    number_string = input('Ввести последовательность цифр ' \
                          'без отделяющих символов: ')

    # Вызвать функцию string_total и сохранить сумму.
    total = string_total(number_string)

    # Показать сумму.
    print('Сумма цифр в введенной ' \
          'Вами строке равняется', total)

# Функция string_total получает строковое значение и
# возвращает сумму всех цифр, содержащихся в строке.
# Функция предполгает, что строковое значение не содержит нечисловые символы.
def string_total(string):
    # Локальные переменные
    total = 0
    number = 0

    # Перебрать все символы в строковом значении.
    for i in range(len(string)):
        # Преобразовать символ в целое число.
        number = int(string[i])
        # Прибавть значение в промежуточную сумму.
        total += number
```

```
    # Вернуть total.
    return total

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.3. Принтер дат

```
def main():
    # Локальные переменные
    day = 0
    month_num = 0
    month_name = ''
    date_string = ''
    month_list = ['января', 'февраля','марта',
                  'апреля', 'мая','июня', 'июля',
                  'августа', 'сентября', 'октября',
                  'ноября', 'декабря']

    # Получить от пользователя дату в формате дд/мм/гггг.
    date_string = input('Введите дату в формате дд/мм/гггг: ')

    # Разбить строку с датой date_string.
    date_list = date_string.split('/')

    # Получить номера месяца и дня.
    day = date_list[0]
    month_num = int(date_list[1])
    year = date_list[2]

    # Получить название месяца.
    month_name = month_list[month_num - 1]

    # Создать строковое значение для даты в длинном формате.
    long_date = day + ' ' + month_name + ' ' + year + ' г.'

    # Показать дату в длинном формате.
    print(long_date)

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.4. Конвертер азбуки Морзе

```
def main():
    # Локальные переменные
    morse_string = ''
    index = 0
```

```
# Список кодов азбуки Морзе
morse_list = [' ', '--..--', '.-.-.-', '-----',
              '.----', '..---', '...--', '....-',
              '.....', '-....', '--...', '---..',
              '----.', '.-', '-...', '-.-.', '-..', '.', '..-.',
              '--.', '....', '..', '.---', '-.-', '.-..', '--',
              '-.', '---', '.--.', '--.-', '.-.', '...', '-', '..-',
              '...-', '.--', '-..-', '-.-', '--..']

# Получить от пользователя строковое значение.
morse_string = input('Введите строковое значение ' \
                     'для конвертации в коды азбуки Морзе: ')

# Перебрать символы в строке, определить индекс кода
# азбуки Морзе в списке и показать код для этого символа.
for ch in morse_string:
    # Преобразовать символ в верхний регистр.
    ch = ch.upper()

    # Определить индекс в списке.
    if ch == ' ':
        index = 0
    elif ch == ',':
        index = 1
    elif ch == '.':
        index = 2
    elif ch == '?':
        index = 3
    elif ch == '0':
        index = 4
    elif ch == '1':
        index = 5
    elif ch == '2':
        index = 6
    elif ch == '3':
        index = 7
    elif ch == '4':
        index = 8
    elif ch == '5':
        index = 9
    elif ch == '6':
        index = 10
    elif ch == '7':
        index = 11
    elif ch == '8':
        index = 12
    elif ch == '9':
        index = 13
    elif ch == 'A':
```

```
    index = 14
elif ch == 'B':
    index = 15
elif ch == 'C':
    index = 16
elif ch == 'D':
    index = 17
elif ch == 'E':
    index = 18
elif ch == 'F':
    index = 19
elif ch == 'G':
    index = 20
elif ch == 'H':
    index = 21
elif ch == 'I':
    index = 22
elif ch == 'J':
    index = 23
elif ch == 'K':
    index = 24
elif ch == 'L':
    index = 25
elif ch == 'M':
    index = 26
elif ch == 'N':
    index = 27
elif ch == 'O':
    index = 28
elif ch == 'P':
    index = 29
elif ch == 'Q':
    index = 30
elif ch == 'R':
    index = 31
elif ch == 'S':
    index = 32
elif ch == 'T':
    index = 33
elif ch == 'U':
    index = 34
elif ch == 'V':
    index = 35
elif ch == 'W':
    index = 36
elif ch == 'X':
    index = 37
elif ch == 'Y':
    index = 38
```

```
        elif ch == 'Z':
            index = 39

        # Показать коды азбуки Морзе для данного символа.
        print (morse_list[index], ',', sep='', end='')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.5. Алфавитный переводчик номера телефона

```
def main():
    # Локальные переменные
    digit_list = ['2','3','4','5','6','7','8','9']
    alpha_phone_number = ''
    num_phone_number = ''

    # Получить от пользователя строковое значение.
    alpha_phone_number = input('Введите телефонный ' \
                               'номер в формате' \
                               ' XXX-XXX-XXXX: ')

    # Перебрать символы строкового значения, отыскивая
    # для каждого символа номер индекса в списке цифр.
    # Построить строковое значение и показать цифры.
    for ch in alpha_phone_number:
        # Определить, является ли символ буквой.
        if ch.isalpha():
            # Если да, то преобразовать символ в верхний регистр.
            ch = ch.upper()
            # Определить номер индекса для символа
            # из списка цифр.
            if ch == 'A' or ch == 'B' or ch == 'C':
                index = 0
            elif ch == 'D' or ch == 'E' or ch == 'F':
                index = 1
            elif ch == 'G' or ch == 'H' or ch == 'I':
                index = 2
            elif ch == 'J' or ch == 'K' or ch == 'L':
                index = 3
            elif ch == 'M' or ch == 'N' or ch == 'O':
                index = 4
            elif ch == 'P' or ch == 'Q' or ch == 'R' or ch == 'S':
                index = 5
            elif ch == 'T' or ch == 'U' or ch == 'V':
                index = 6
            elif ch == 'W' or ch == 'X' or ch == 'Y' or ch == 'Z':
                index = 7
            # Присвоить символу цифру из списка.
```

```
            ch = digit_list[index]

        # Конкатенировать цифры в строковое значение.
        num_phone_number = num_phone_number + ch

    # Показать цифры телефонного номера.
    print('Телефонный номер:', num_phone_number)

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.6. Среднее количество слов

```
def main():
    # Локальные переменные
    num_sentences = 0
    total_words = 0
    average_words = 0.0
    words = []

    try:
        # Открыть файл text.txt для чтения.
        # Файл находится в подпапке data
        infile = open(r'data\text.txt', 'r')

        # Прочитать данные в список.
        # Каждый элемент списка является предложением.
        sentences = infile.readlines()

        # Количество предложений равняется длине списка.
        num_sentences = len(sentences)

        # Количеством значений в каждом списке
        # является количество слов в предложении.
        for item in sentences:
            words = item.split()
            total_words += len(words)

        # Вычислить среднее количество слов.
        average_words = float(total_words) / num_sentences

        # Показать среднее количество слов.
        print ('Среднее количество слов в строке:', average_words)

        # Закрыть файл.
        infile.close()

   # Обработать любые ошибки, которые могут произойти.
    except IOError:
        print('Произошла ошибка при открытии файла.')
```

```
    except:
        print('Произошла ошибка.')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.7. Анализ символов

```
def main():
    # Локальные переменные
    num_upper = 0
    num_lower = 0
    num_space = 0
    num_digits = 0
    data = ''

    # Открыть файл text.txt для чтения.
    # Файл находится в подпапке data
    infile = open(r'data\text.txt', 'r')

    # Прочитать данные из файла.
    data = infile.read()

    # В цикле перебрать каждый символ в файле.
    # Определить, находится ли символ в верхнем регистре,
    # в нижнем регистре, является цифрой или пробелом, и
    # вести учет промежуточной суммы по каждому показателю.
    for ch in data:
        if ch.isupper():
            num_upper = num_upper + 1
        if ch.islower():
            num_lower = num_lower + 1
        if ch.isdigit():
            num_digits = num_digits + 1
        if ch.isspace():
            num_space = num_space + 1

    # Закрыть файл.
    infile.close()

    # Показать итоговые значения.
    print('Буквы в верхнем регистре:', num_upper)
    print('Буквы в нижнем регистре:', num_lower)
    print('Цифры:', num_digits)
    print('Пробелы:', num_space)

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.8. Корректор предложений

```
def main():
    # Получить от пользователя строковое значение.
    user_string = input('Введите строковое значение, чтобы программа ' \
                        'напечатала предложения с заглавной буквы: ')

    # Вызвать функцию capitalize, сохранив результат.
    result = capitalize(user_string)

    # Показать результат.
    print(result)

# Функция capitalize получает строковое значение и возвращает
# то же самое значение, в котором первые буквы всех
# предложений будут напечатаны заглавными буквами
def capitalize(string):
    # Инициализировать переменные
    result = ''
    new_sentence = True
    result_word = ''

    # Получить все слова в строковом значении.
    words = string.split()

    # По каждому слову в строковом значении:
    for item in words:
       # Это слово является началом нового предложения.
        if new_sentence:
            # Создать новое слово, в котором первый символ
                    # переведен в верхний регистр.
            result_word = item[0].upper() + item[1:]
        else:
            # Ничего не делать.
            result_word = item

        # Добавить результирующее слово в строковое значение.
        result = result + result_word + ' '

        # Если последний символ в слове указывает на конец
        # предложения, то назначить флаговой переменной flag,
        # чтобы обеспечить, что следующее слово будет
        # рассматриваться, как новое предложение.
        if item[-1] == '.' or item[-1] == '?' or item[-1] == '!':
            new_sentence = True
        else:
            new_sentence = False

    # Вернуть результат.
    return result
```

```
# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.9. Гласные и согласные

```
def main():
    # Локальные переменные
    vowels = 0
    consonants = 0

    # Получить от пользователя строковое значение.
    user_string = input('Введите строковое значение: ')

    # Вызвать функцию vowel_counter,
    # сохранив результат.
    vowels = vowel_counter(user_string)

    # Вызвать функцию consonant_counter,
    # сохранив результат.
    consonants = consonant_counter(user_string)

    # Показать результаты.
    print('Введенное Вами строковое значение содержит', vowels, \
          'гласных и', consonants, 'согласных.')

# Функция vowel_counter получает строковое значение и
# возвращает количество гласных в строковом значении.
def vowel_counter(string):
    # Задать локальные переменные
    count = 0
    vowels = 'аеёиоуыэюя'

    # По каждому символу определить,
    # является ли он гласным.
    for ch in string:
        if vowels.find(ch) >= 0:
            count = count + 1

    # Вернуть количество гласных в строковом значении.
    return count

# Функция consonant_counter возвращает строковое значение и
# возвращает количество согласных в строковом значении.
def consonant_counter(string):
    # Задать локальные переменные
    count = 0
    consonants = 'бвгджзйклмнпрстфхцчшщъь'

    # По каждому символу определить,
```

```
    # является ли он согласным.
    for ch in string:
        if consonants.find(ch) >= 0:
            count = count + 1

    # Вернуть количество согласных в строковом значении.
    return count

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.10. Самый частотный символ

```
# Функция показывает символ, который в строковом значении
# появляется чаще всех. Если несколько символов имеют
# одинаковую самую высокую частоту, то она показывает
# первый символ с этой частотой.
def main():

    # Задать локальные переменные
    count = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
    letters = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'
    index = 0
    frequent = 0

    # Получить от пользователя входные данные.
    user_string = input('Введите строковое значение: ')

    for ch in user_string:

        ch = ch.upper()

        # Определить, какой буквой является этот символ.
        index = letters.find(ch)
        if index>=0:

            # Увеличить элемента массива, соответствующий этой букве.
            count[index] = count[index] + 1

    for i in range(len(count)):
        if count[i] > count[frequent]:
            frequent = i

    print('Самый частотный символ в строковом значении: ', \
          letters[frequent], '.', sep='')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.11. Разделитель слов

```
# Функция исходит из того, что входное значение
# не содержит имен собственных.
def main():
    # Задать локальную переменную.
    result = ''

    # Получить от пользователя входные данные.
    user_string = input('Ввести строковое значение: ')

    # Скопировать первую букву в строковом значении
    # в верхнем регистре.
    result = result + user_string[0]

    for i in range(1, len(user_string)):

        ch = user_string[i]

        # Если следующий символ находится в верхнем регистре, то
        # вставить пробел для нового слова и преобразовать
        # букву в нижний регистр.
        if ch.isupper():
            ch = ch.lower()
            result = result + ' '

        result = result + ch

    print(result)

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.12. Молодежный жаргон

```
def main():
    # Задать локальные переменные.
    result = ''
    current_word = ''
    ENDING = 'КИ'

    # Получить от пользователя данные.
    user_string = input('Введите строковое значение: ')

    # Разделить входное значение на отдельные слова.
    words = user_string.split()

    # Цикл, который изменяет каждое слово.
    for i in range(len(words)):
```

```
        item = words[i].upper()

        # Для однобуквенных слов добавить только окончание.
        if len(item) == 1:
            current_word = item + ENDING

        # Для слов с двумя и более буквами,
        # изменить порядок следования и добавить окончание.
        else:
            current_word = item[1:] + item[0] + ENDING

        # Добавить адаптированное слово к результату.
        result = result + current_word

        # Если есть еще слова, то добавить к результату пробел.
        if i < len(words) + 1:
            result = result + ' '

    # Показать результат.
    print(result)

# Вызвать главную функцию.
main()
```

Упражнение по программированию 8.13 (части 1 и 2). Лотерея PowerBall — распространенность чисел

```
# Именованная константа
LOTTERY_NUMBERS = 69

# Функция get_all_numbers возвращает список с лотерейными
# числами файла pbnumbers.txt. Числа появляются в том
# порядке, в каком они были прочитаны из файла.
def get_all_numbers():
    # Открыть файл с лотерейными числами.
    # Файл находится в подпапке data
    pblottery_file = open(r'data\pbnumbers.txt', 'r')

    # Прочтитать содержимое файла в список.
    pblottery = pblottery_file.readlines()

    # Закрыть файл.
    pblottery_file.close()

    # Удалить из каждого элемента символ новой строки.
    for i in range(len(pblottery)):
        pblottery[i] = pblottery[i].rstrip('\n')

    # Разбить каждый элемент на отдельные числа и сохранить
    # отдельные регулярные числа в списке под названием lotto_nums.
```

```
    lotto_nums = []
    for i in range(len(pblottery)):
        number_set = pblottery[i].split()
        for j in range(len(number_set)):
            lotto_nums.append(int(number_set[j]))

    # Вернуть список lotto_nums.
    return lotto_nums

# Функция get_frequency принимает список чисел и определяет
# частоту каждого значения в списке. Параметр max_value
# обозначает максимальное значение, хранящееся в списке.
def get_frequency(number_list, max_value):
    # Создать список для частоты каждого числа.
    # Каждый элемент списка инициализируется нулем.
    frequency = [0] * (max_value + 1)
    for i in range(len(number_list)):
        # Получить следующее лотерейное число в списке.
        num = number_list[i]

        # Увеличить частоту этого числа.
        frequency[num] += 1

    # Вернуть частотный список.
    return frequency

# Функция position_of_highest_value возвращает позицию
# самого большого значения в списке num_list.
def position_of_highest_value(num_list):
    highest = 0
    highest_position = 0
    for i in range(len(num_list)):
        if num_list[i] > highest:
            highest = num_list[i]
            highest_position = i

    return highest_position

# Функция most_common принимает частотный список freq_list и возвращает
# другой список, в котором элемент 0 содержит позицию самого большого
# значения в списке freq_list, элемент 1 содержит позицию второго
# самого большого значения в списке freq_list, и т.д.
def most_common(freq_list):
    # Создать пустой список для позиций самых распространенных значений.
    common_sorted = []

    # Сделать копию списка freq_list.
    temp_list = []
    for item in freq_list:
```

```
        temp_list.append(item)

    for i in range(len(temp_list)):
        position = position_of_highest_value(temp_list)
        common_sorted.append(position)
        temp_list[position] = -1

    # Вернуть список common_sorted.
    return common_sorted

def main():
    # Получить список всех лотерейных чисел.
    lotto_nums = get_all_numbers()

    # Получить частоту каждого числа.
    frequency = get_frequency(lotto_nums, LOTTERY_NUMBERS)

    # Получить список наиболее распространенных значений.
    sorted_by_most_common = most_common(frequency)

    # Показать 10 наиболее распространенных чисел.
    print('10 наиболее распространенных чисел (по убыванию)')
    print ('-------------------------------------------------')
    for i in range(10):
        print(sorted_by_most_common[i])

    # Показать 10 наименее распространенных чисел.
    sorted_by_most_common.reverse()
    print('\n10 наиболее распространенных чисел (по возрастанию)')
    print ('-------------------------------------------------')
    for i in range(1, 11):
        print(sorted_by_most_common[i])

# Вызвать главную функцию
main()
```

Упражнение по программированию 8.13 (часть 3). Лотерея PowerBall — "созревшие" числа

```
# Именованные константы
LOTTERY_NUMS = 69
MAX_NUM_OVERDUE = 10

# Функция get_all_numbers возвращает список с лотерейными
# числами файла pbnumbers.txt. Числа появляются в том
# порядке, в каком они были прочитаны из файла.
def get_all_numbers():
    # Открыть файл с лотерейными числами.
    # Файл находится в подпапке data
    pblottery_file = open(r'data\pbnumbers.txt', 'r')
```

```
    # Прочитать содержимое файла в список.
    pblottery = pblottery_file.readlines()

    # Закрыть файл.
    pblottery_file.close()

    # Удалить из каждого элемента символ новой строки.
    for i in range(len(pblottery)):
        pblottery[i] = pblottery[i].rstrip('\n')

    # Разбить каждый элемент на отдельные числа и сохранить
    # отдельные регулярные числа в списке под названием lotto_nums.
    lotto_nums = []
    for i in range(len(pblottery)):
        number_set = pblottery[i].split()
        for j in range(len(number_set)):
            lotto_nums.append(int(number_set[j]))

    # Вернуть список lotto_nums.
    return lotto_nums

# Функция get_last_positions принимает в качестве аргумента список
# чисел и максимальное количество, найденное в списке. Она создает
# список last_position, в котором last_position[i] содержит
# последний индекс в number_list, который содержит i.
# Эта функция возвращает список last_position.
def get_last_position(number_list, max_number):
    # Создать список для последней позиции каждого числа.
    # Каждый элемент списка инициализируется нулем.
    last_position = [0] * (max_number + 1)

    for i in range(len(number_list)):
        # Получить следующее лотерейное число в списке.
        num = number_list[i]

        # Сохранить позицию этого числа.
        last_position[num] = i

    # Вернуть список last_position.
    return last_position

# Функция position_of_lowest_value возвращает позицию
# минимального значения в списке num_list.
def position_of_lowest_value(num_list):
    lowest = num_list[1]
    lowest_position = 1
    for i in range(2, len(num_list)):
        if num_list[i] < lowest:
```

```
            lowest = num_list[i]
            lowest_position = i

    return lowest_position

# Функция most_overdue принимает pos_list и возвращает другой
# список, в котором элемент 0 содержит самое созревшее значение в
# pos_list, элемент 1 содержит второе самое созревшее значение в
# pos_list, и т.д.
def most_overdue(pos_list):
    # Создать пустой список для самых созревших значений.
    overdue_sorted = []

    # Сделать копию списка pos_list.
    temp_list = []
    for item in pos_list:
        temp_list.append(item)

    # Получить максимальное значение в списке temp_list.
    max_value = max(temp_list)

    # Определить максимальное количество созревших
    # чисел для отслеживания.
    if MAX_NUM_OVERDUE < len(temp_list):
        num_overdue = MAX_NUM_OVERDUE
    else:
        num_overdue = len(temp_list)

    # Получить самые созревшие числа.
    for i in range(num_overdue):
        position = position_of_lowest_value(temp_list)
        overdue_sorted.append(position)
        temp_list[position] = max_value + 1

    # Вернуть список common_sorted.
    return overdue_sorted

def main():
    # Получить список всех лотерейных чисел.
    lotto_nums = get_all_numbers()

    # Получить список последних позиций.
    last_position_list = get_last_position(lotto_nums, LOTTERY_NUMS)

    # Получить список самых созревших значений.
    most_overdue_nums = most_overdue(last_position_list)

    # Определить максимальное количество созревших чисел
    # для отслеживания.
```

```
    if MAX_NUM_OVERDUE < len(most_overdue_nums):
        num_overdue = MAX_NUM_OVERDUE
    else:
        num_overdue = len(most_overdue_nums)

    # Показать 10 самых "созревших" чисел.
    print('10 самых "созревших" чисел')
    print('--------------------------')
    for i in range(num_overdue):
        print(most_overdue_nums[i])

# Вызвать главную функцию
main()
```

Упражнение по программированию 8.13 (часть 4). Лотерея PowerBall — частоты чисел

```
# Именованные константы
LOTTERY_NUMBERS = 69
POWERBALL_NUMBERS = 26

def main():
    # Получить список всех чисел лотереи.
    lottery_list = get_numbers()

    # Создать списки для частоты каждого числа.
    # Списки инициализируются нулем для каждого элемента.
    reg_frequency = [0] * (LOTTERY_NUMBERS + 1)
    pb_frequency = [0] * (POWERBALL_NUMBERS + 1)

    # Получить частоту каждого регулярного числа.
    for i in range(len(lottery_list[0])):
        # Получить следующее число в списке.
        num = lottery_list[0][i]

        # Увеличить частоту этого числа.
        reg_frequency[num] += 1

    # Получить частоту каждого числа лотереи PowerBall.
    for i in range(len(lottery_list[1])):
        # Получить следующее число в списке.
        num = lottery_list[1][i]

        # Увеличить частоту этого числа.
        pb_frequency[num] += 1

    # Показать частоту каждого регулярного числа.
    print('Частоты регулярных чисел')
    print('------------------------')
    for i in range(1, len(reg_frequency)):
```

```
        print(i, 'было выбрано', reg_frequency[i], 'раз.')

    # Показать частоту каждого числа лотереи PowerBall.
    print('\nЧастоты чисел лотереи PowerBall')
    print('-------------------------------')
    for i in range(1, len(pb_frequency)):
        print(i, 'было выбрано', pb_frequency[i], 'раз.')

# Функция get_numbers возвращает двумерный список с двумя
# элементами. Первый элемент - это список регулярных лотерейных
# чисел, и 2-й элемент - это список чисел лотереи PowerBall.
def get_numbers():
    # Открыть файл с лотерейными числами.
    # Файл находится в подпапке data
    pblottery_file = open(r'data\pbnumbers.txt', 'r')

    # Прочтитать содержимое файла в список.
    work_list = pblottery_file.readlines()

    # Закрыть файл.
    pblottery_file.close()

    # Удалить из каждого элемента символ новой строки.
    for i in range(len(work_list)):
        work_list[i] = work_list[i].rstrip('\n')

    # Разбить каждый элемент на отдельные числа и сохранить
    # отдельные регулярные числа в списке под названием lotto_nums
    # и отдельные числа лотереи PowerBall в список pb_numbers.
    lotto_nums = []
    pb_numbers = []
    for i in range(len(work_list)):
        number_set = work_list[i].split()
        for j in range(len(number_set) - 1):
            lotto_nums.append(int(number_set[j]))
        pb_numbers.append(int(number_set[len(number_set)-1]))

    pblottery = [[],[]]
    pblottery[0] = lotto_nums
    pblottery[1] = pb_numbers

    # Вернуть список pblottery.
    return pblottery

# Вызвать главную функцию
main()
```

Упражнение по программированию 8.14 (часть 1). Цены на бензин — средняя цена за год

```
# Именованные константы
STARTING_YEAR = 1993
ENDING_YEAR = 2013

# Функция get_price принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент с ценой
# в виде вещественного числа.
def get_price(str):
    # Разбить строковое значение по дефисам.
    items = str.split(':')
    # Вернуть цену как вещественное число.
    return float(items[1])

# Функция get_month принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ММ
# в виде целого числа.
def get_month(str):
    # Разбить строковое значение по дефисам.
    items = str.split('-')
    # Вернуть месяц как целое число.
    return int(items[0])

# Функция get_day принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ДД
# в виде целого числа.
def get_day(str):
    # Разбить строковое значение по дефисам.
    items = str.split('-')
    # Вернуть день как целое число.
    return int(items[1])

# Функция get_year принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ГГГГ
# в виде целого числа.
def get_year(str):
    # Разбить строковое значение по двоеточию.
    items = str.split(':')
    # Разбить строковое значение по дефисам.
    date_items = items[0].split('-')
    # Вернуть год как целое число.
    return int(date_items[2])

# Функция get_yearly_average имеет два параметра: gas_list и year.
# Параметр gas_list - это список строковых значений, которые имеют
# формат ММ-ДД-ГГГГ:Цена. Параметр year - это числовое значение,
# являющееся годом. Эта функция возвращает среднюю цену для всех
# элементов, в которых компонент ГГГГ равняется параметру year.
```

```
def get_yearly_average(gas_list, year):
    # Инициализировать накопитель.
    total = 0

    # Инициализировать счетчик.
    count = 0

    # Выполнить обход списка, получая сумму всех цен
    # за указанный год.
    for e in gas_list:
        if get_year(e) == year:
            total += get_price(e)
            count += 1

    # Вычислить среднее.
    average = total / count

    # Вернуть среднее.
    return average

def main():
    # Открыть файл.
    # Файл находится в подпапке data
    gas_file = open(r'data\GasPrices.txt', 'r')

    # Прочитать содержимое файла в список.
    gas_list = gas_file.readlines()

    # Показать среднегодовые цены.
    for i in range(STARTING_YEAR, ENDING_YEAR + 1):
        print('Средняя цена в ', i,
              ' составила $', format(get_yearly_average(gas_list, i), '.2f'),
              sep = '')

# Вызвать главную функцию
main()
```

Упражнение по программированию 8.14 (часть 2). Цены на бензин — средняя цена за месяц

```
# Функция get_price принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент с ценой
# в виде вещественного числа.
def get_price(str):
    # Разбить строковое значение по дефисам.
    items = str.split(':')
    # Вернуть цену как вещественное число.
    return float(items[1])

# Функция get_month принимает строковое значение, которое находится
```

```
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ММ
# в виде целого числа.
def get_month(str):
    # Разбить строковое значение по дефисам.
    items = str.split('-')
    # Вернуть месяц как целое число.
    return int(items[0])

# Функция get_year принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ГГГГ
# в виде целого числа.
def get_year(str):
    # Разбить строковое значение по двоеточию.
    items = str.split(':')
    # Разбить строковое значение по дефисам.
    date_items = items[0].split('-')
    # Вернуть год как целое число.
    return int(date_items[2])

# Функция display_monthly_averages выполняет обход списка gas_list,
# вычисляя и показывая среднемесячную цену.
def display_monthly_averages(gas_list):
    month_names = ['январь', 'февраль', 'март', 'апрель', 'май',
                   'июнь', 'июль', 'август', 'сентябрь', 'октябрь',
                   'ноябрь', 'декабрь']
    current_month = get_month(gas_list[0])
    current_year = get_year(gas_list[0])
    total = 0
    count = 0
    average = 0

    # Выполнить обход списка.
    for e in gas_list:
        if (get_month(e) == current_month) and (get_year(e) == current_year):
            total += get_price(e)
            count += 1
        else:
            average = total / count
            print('Средняя цена за ', month_names[current_month-1],
                  ', ', current_year, ': $',
                  format(average, '.2f'), sep='')
            current_month = get_month(e)
            current_year = get_year(e)
            total = 0
            count = 0

    # Показать среднее значение за последний месяц.
    print('Средняя цена за ', month_names[current_month-1],
          ', ', current_year, ': $',
```

```
          format(average, '.2f'), sep='')

def main():
    # Открыть файл.
    # Файл находится в подпапке data
    gas_file = open(r'data\GasPrices.txt', 'r')

    # Прочитать содержимое файла в список.
    gas_list = gas_file.readlines()

    # Показать среднемесячные цены.
    display_monthly_averages(gas_list)

# Вызвать главную функцию
main()
```

Упражнение по программированию 8.14 (часть 3). Цены на бензин — наибольшая и наименьшая цены в году

```
# Функция get_price принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент с ценой
# в виде вещественного числа.
def get_price(str):
    # Разбить строковое значение по дефисам.
    items = str.split(':')
    # Вернуть цену как вещественное число.
    return float(items[1])

# Функция get_year принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ГГГГ
# в виде целого числа.
def get_year(str):
    # Разбить строковое значение по двоеточию.
    items = str.split(':')
    # Разбить строковое значение по дефисам.
    date_items = items[0].split('-')
    # Вернуть год как целое число.
    return int(date_items[2])

# Функция display_highest_per_year выполняет обход списка
# gas_list, показывая самую высокую цену в каждом году.
def display_highest_per_year(gas_list):
    current_year = get_year(gas_list[0])
    highest = get_price(gas_list[0])

       # Выполнить обход списка
    for e in gas_list:
        if get_year(e) == current_year:
            if get_price(e) > highest:
                highest = get_price(e)
```

```
        else:
            print('Самая высокая цена в ', current_year, ': $',
                  format(highest, '.2f'), sep='')
            current_year = get_year(e)
            highest = get_price(e)

    # Показать самую высокую цену для последнего года.
    print('Самая высокая цена в ', current_year, ': $',
          format(highest, '.2f'), sep='')

# Функция display_lowest_per_year выполняет обход списка
# gas_list, показывая самую низкую цену в каждом году.
def display_lowest_per_year(gas_list):
    current_year = get_year(gas_list[0])
    lowest = get_price(gas_list[0])

    # Выполнить обход списка.
    for e in gas_list:
        if get_year(e) == current_year:
            if get_price(e) < lowest:
                lowest = get_price(e)
        else:
            print('Самая низкая цена в ', current_year, ': $',
                  format(lowest, '.2f'), sep='')
            current_year = get_year(e)
            lowest = get_price(e)

    # Показать самую низкую цену для последнего года.
    print('Самая низкая цена в ', current_year, ': $',
          format(lowest, '.2f'), sep='')

def main():
    # Открыть файл.
    # Файл находится в подпапке data
    gas_file = open(r'data\GasPrices.txt', 'r')

    # Прочитать содержимое файла в список.
    gas_list = gas_file.readlines()

    # Показать самые высокие цены в году.
    display_highest_per_year(gas_list)

    # Показать самые низкие цены в году.
    display_lowest_per_year(gas_list)

# Вызвать главную функцию
main()
```

Упражнение по программированию 8.14 (часть 4). Цены на бензин — список цен, упорядоченный по возрастанию

```
# Функция get_price принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент с ценой
# в виде вещественного числа.
def get_price(str):
    # Разбить строковое значение по дефисам.
    items = str.split(':')
    # Вернуть цену как вещественное число.
    return float(items[1])

# Функция get_date принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ММ-ДД-ГГГГ
# в виде строкового значения.
def get_date(str):
    # Разбить строковое значение по двоеточию.
    items = str.split(':')
    # Разбить строковое значение по дефисам.
    return str(items[0])

# Функция lowest_element_position возвращает позицию
# элемента в списке g_list с самым низким значением.
def lowest_element_position(g_list):
    lowest = get_price(g_list[0])
    position = 0
    for i in range(1, len(g_list)):
        if get_price(g_list[i]) < lowest:
            lowest = get_price(g_list[i])
            position = i

    # Вернуть позицию самого низкого значения.
    return position

# Функция create_low_to_high_file создает файл с именем
# low_to_high.txt, содержащий элементы списка gas_list,
# отсортированные в порядке возрастания цены.
def create_low_to_high_file(gas_list):
    # Сделать копию списка gas_list.
    temp_list = []
    for e in gas_list:
        temp_list.append(e)

    # Открыть файл для записи.
    # Файл будет находиться в подпапке data.
    outputfile = open(r'data\low_to_high.txt', 'w')

    while (len(temp_list) > 0):
        # Получить индекс элемента с самой низкой ценой.
        lowest_index = lowest_element_position(temp_list)
```

```
        # Получить этот элемент.
        lowest_line = temp_list[lowest_index]

        # Записать этот элемент в файл.
        outputfile.write(lowest_line)

        # Удалить этот элемент из списка.
        del temp_list[lowest_index]

    # Закрыть файл.
    outputfile.close()

def main():
    # Открыть файл.
    # Файл находится в подпапке data
    gas_file = open(r'data\GasPrices.txt', 'r')

    # Прочитать содержимое файла в список.
    gas_list = gas_file.readlines()

    # Создать файл с элементами, отсортированными
    # по цене в порядке возрастания.
    create_low_to_high_file(gas_list)

# Вызвать главную функцию
main()
```

Упражнение по программированию 8.14 (часть 5). Цены на бензин — список цен, упорядоченный по убыванию

```
# Функция get_price принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент с ценой
# в виде вещественного числа.
def get_price(str):
    # Разбить строковое значение по дефисам.
    items = str.split(':')
    # Вернуть цену как вещественное число.
    return float(items[1])

# Функция get_date принимает строковое значение, которое находится
# в формате ММ-ДД-ГГГГ:Цена. Она возвращает компонент ММ-ДД-ГГГГ
# в виде строкового значения.
def get_date(str):
    # Разбить строковое значение по двоеточию.
    items = str.split(':')
    # Разбить строковое значение по дефисам.
    return str(items[0])

# Функция highest_element_position возвращает позицию
```

```
# элемента в списке g_list с самым высоким значением.
def highest_element_position(g_list):
    highest = get_price(g_list[0])
    position = 0
    for i in range(1, len(g_list)):
        if get_price(g_list[i]) > highest:
            #print('* true *', get_price(g_list[i]), '>', highest)
            highest = get_price(g_list[i])
            position = i

    # Вернуть позицию самого высокого значения.
    return position

# Функция create_high_to_low создает файл с именем
# high_to_low.txt, содержащий элементы списка gas_list,
# отсортированные в порядке убывания цены.
def create_high_to_low_file(gas_list):
    # Сделать копию списка gas_list.
    temp_list = []
    for e in gas_list:
        temp_list.append(e)

    # Открыть файл для записи.
    # Файл будет находиться в подпапке data
    outputfile = open(r'data\high_to_low.txt', 'w')

    while (len(temp_list) > 0):
        # Получить индекс элемента с самой высокой ценой.
        highest_index = highest_element_position(temp_list)

        # Получить этот элемент.
        highest_line = temp_list[highest_index]

        # Записать этот элемент в файл.
        outputfile.write(highest_line)

        # Удалить этот элемент из списка.
        del temp_list[highest_index]

    # Закрыть файл.
    outputfile.close()

def main():
    # Открыть файл.
    # Файл находится в подпапке data
    gas_file = open(r'data\GasPrices.txt', 'r')

    # Прочитать содержимое файла в список.
    gas_list = gas_file.readlines()
```

```
    # Создать файл с элементами, отсортированными
    # по цене в порядке убывания.
    create_high_to_low_file(gas_list)

# Вызвать главную функцию
main()
```