Упражнения по программированию главы 7

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

Упражнение по программированию 7.1. Общий объем продаж

```
def main():
    # Переменные
    total_sales = 0.0

    # Инициализировать списки
    daily_sales = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
    days_of_week = ['понедельник', 'вторник', 'среда',
                    'четверг', 'пятница', 'суббота',
                    'воскресенье']

    for i in range(7):
        daily_sales[i] = float(input('Введите продажи за ' \
                                     + days_of_week[i] + ': '))

    for number in daily_sales:
        total_sales += number

    # Показать общий объем продаж
    print ('Общий объем продаж за неделю: $', \
           format(total_sales, ',.2f'), sep='')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.2. Генератор лотерейных чисел

```
import random

def main():
    # Инициализировать список чисел.
    number_list = [0, 0, 0, 0, 0, 0, 0]

    # Присвоить списку случайные числа.
    for i in range(7):
        number_list[i] = random.randint(0, 9)

    # Показать числа в одной строке.
    for i in range(7):
        print (number_list[i], end='')

        # Отделить текущее число от следующего.
        if i < 6:
            print(', ', end='')
```

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

Упражнение по программированию 7.3. Статистика дождевых осадков

```
def main():

    # Локальные переменные
    total = 0.0
    average = 0.0
    highest = 0.0
    lowest = 0.0
    month_lowest = ''
    month_highest = ''

    # Список для данных о дождевых осадках
    month_rain = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                  0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

    # Инициализировать список названиями месяцев.
    month_list = ['январь', 'февраль', 'март',
                  'апрель', 'май', 'июнь', 'июль',
                  'август', 'сентябрь', 'октябрь',
                  'ноябрь', 'декабрь']

    # Получить величину дождевых осадков за каждый месяц.
    for i in range(12):
        month_rain[i] = float(input('Введите дождевые осадки за ' \
                                    + month_list[i] + ": "))

    # Вычислить суммарную величину.
    total = sum(month_rain)

    # Вычислить среднюю величину.
    average = total / 12.0

    # Вычислить максимальную величину.
    highest = max(month_rain)

    # Получить индекс месяца с самой высокой величиной дождевых осадков.
    month_highest = month_rain.index(highest)

    # Вычислить минимум.
    lowest = min(month_rain)

    # Получить индекс месяца с самой низкой величиной дождевых осадков.
    month_lowest = month_rain.index(lowest)

    # Показать результаты
```

```
    print('Суммарная величина дождевых осадков:',
            format(total, '.2f'))
    print('Средняя величина дождевых осадков:',
            format(average, '.2f'))
    print('Максимальная величина дождевых осадков:',
            month_list[month_highest])
    print('Минимальная величина дождевых осадков:',
            month_list[month_lowest])

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.4. Программа анализа чисел

```
def main():
    # Список для хранения чисел
    number_list = []

    # Переменные
    low = 0.0
    high = 0.0
    total = 0.0
    average = 0.0
    number = 0

    # Предложить ввести числа
    for i in range(20):
        number = float(input('Введите число ' + \
                             str(i + 1) + \
                             ' из 20: '))
        number_list.append(number)

    low = min(number_list)
    high = max(number_list)
    total = sum(number_list)
    average = total / 20.0

    print ('Минимальное:', low)
    print ('Максимальное:', high)
    print ('Сумма:', format(total, ',.2f'))
    print ('Среднее:', format(average, ',.2f'))

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.5. Проверка допустимости номера расходного счета

```
def main():
    # Локальные переменные
```

```
    test_account = ''

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

        # Прочитать все строки из файла в список
        accounts = input_file.readlines()

        # Отсечь замыкающий символ '\n' у всех элементов списка
        for i in range(len(accounts)):
            accounts[i] = accounts[i].rstrip('\n')

        # Получить от пользователя входные данные
        test_account = input('Введите номер счета для проверки: ')

        # Применить оператор in для поиска указанного
              # пользователем номера счета
        if test_account in accounts:
            print('Номер счета', test_account, 'допустимый.')
        else:
            print('Номер счета', test_account, 'недопустимый.')

    except IOError:
        print('Файл не найден')
    except:
        print('Произошла ошибка')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.6. Больше числа n

```
def main():
    # Объявить локальные переменные
    number = 5
    number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    # Показать число.
    print('Число:', number)

    # Показать список чисел.
    print('Список чисел:\n', number_list, sep='')

    # Показать список чисел, которые больше этого числа.
    print('Список чисел, которые больше числа ', \
          number, ':', sep='')

    # Вызвать функцию display_larger_than_n_list, передав
```

```
    # число и список чисел в качестве аргументов.
    display_larger_than_n_list(number, number_list)

# Функция display_larger_than_n_list принимает два аргумента:
# список и число. Эта функция показывает все числа в списке,
# которые больше указанного числа.
def display_larger_than_n_list(n, n_list):
    # Объявить пустой список.
    larger_than_n_list = []

    # Перебрать значения в списке.
    for value in n_list:

        # Определить, является ли значение больше n.
        if value > n:

            # Если да, то добавить это значение в список.
            larger_than_n_list.append(value)

    # Display the list.
    print(larger_than_n_list)

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.7. Экзамен на получение водительских прав

```
# Программа исходит из того, что решения испытуемого перечислены
# построчно, т. е. каждая строка содержит ответ испытуемого на вопрос,
# но без предваряющего ответ номера вопроса. Предполагается, что
# ответы испытуемого расположены в порядке следования вопросов.

def main():
    # Определить переменные
    solution = ['A', 'C', 'A', 'A', 'D',
                'B', 'C', 'A', 'C', 'B',
                'A', 'D', 'C', 'A', 'D',
                'C', 'B', 'B', 'D', 'A']
    correct_count = 0
    incorrect_count = 0
    incorrect_questions = []
    counter = 0

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

        # Прочитать все строки из файла в список.
```

```
        student_solutions = input_file.readlines()

        # Отсечь замыкающий символ '\n' у всех элементов списка.
        for i in range(len(student_solutions)):
            student_solutions[i] = student_solutions[i].rstrip('\n')

        # Сравнить решение испытуемого с правильным решением и
        # обновить соответствующие счетчики.
        for i in range(len(student_solutions)):
            if student_solutions[i] == solution[i]:
                correct_count += 1
            else:
                incorrect_count += 1
                incorrect_questions.append(str(i + 1))

        # Определить, прошел ли испытуемый экзамен, и показать результат.
        if correct_count >= 15:
            print('Поздравляем!! Вы прошли экзамен.')
        else:
            print('Сожалеем, но Вы не прошли экзамен.')

        # Показать подробную информацию об экзамене.
        print('Число вопросов, на которые Вы ответили правильно:', \
              correct_count)
        print('Число вопросов, на которые Вы ответили не правильно:', \
              incorrect_count)

        # Определить, ответил ли испытуемый хоть один раз неправильно.
        if incorrect_count > 0:
            # Показать номера вопросов, на которые испытуемый не ответил.
            print('Вопросы, на которые Вы ответили неправильно:', end='')
            while counter < incorrect_count:
                print(incorrect_questions[counter], end='')
                if counter + 1 < incorrect_count:
                    print (', ', end='')
                counter += 1

    # Обработать любые ошибки, которые могут произойти.
    except IOError:
        print('Файл не найден')
    except IndexError:
        print('Ошибка индексации')
    except:
        print('Произошла ошибка')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.8. Поиск имени

```
def main():
    # Определить переменные
    boy = ''
    girl = ''

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

        # Прочитать все строки из файла в список.
        popular_boys = boy_input.readlines()

        # Отсечь замыкающий символ '\n' у всех элементов списка.
        for i in range(len(popular_boys)):
            popular_boys[i] = popular_boys[i].rstrip('\n')

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

        # Прочитать все строки из файла в список.
        popular_girls = girl_input.readlines()

        # Отсечь замыкающий символ '\n' у всех элементов списка.
        for i in range(len(popular_girls)):
            popular_girls[i] = popular_girls[i].rstrip('\n')

        # Получить от пользователя входные данные.
        boy = input("Введите мужское имя или Н, если Вы" \
                    " не хотите вводить мужское имя: ")
        girl = input("Введите женское имя или Н, если Вы" \
                     " не хотите вводить женское имя: ")

        # Показать результат для введенного пользователем мужского имени
        if boy=='Н':
            print("Вы решили не вводить мужское имя.")
        elif boy in popular_boys:
            print(boy, "является одним из самых популярных мужских имен.")
        else:
            print(boy, "не является одним из самых популярных мужских имен.")

        # Показать результат для введенного пользователем женского имени
        if girl=='Н':
            print("Вы решили не вводить женское имя.")
        elif girl in popular_girls:
            print(girl, "является одним из самых популярных женских имен.")
        else:
```

```
            print(girl, "не является одним из самых популярных женских имен.")

    # Обработать любые ошибки, которые могут произойти.
    except IOError:
        print('Файл не найден')
    except IndexError:
        print('Ошибка индексации')
    except:
        print('Произошла ошибка')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.9. Данные о населении

```
# Программа исходит из того, что все изменения численности
# населения являются положительными. Иными словами, каждый
# год численность населения была больше, чем в
# предыдущем году.
def main():

    # Определить переменные
    yearly_change = []
    change = 0.0
    total_change = 0.0
    average_change = 0.0
    greatest_increase = 0.0
    smallest_increase = 0.0
    greatest_year = 0
    smallest_year = 0

    # Константа для базового года
    BASE_YEAR = 1950

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

        # Прочитать все строки из файла в список.
        yearly_population = input_file.readlines()

        # Преобразовать все строковые значения в числа.
        for i in range(len(yearly_population)):
            yearly_population[i] = float(yearly_population[i])

        # Вычислить изменение численности населения
              # по каждым двум годам.
        for i in range(1, len(yearly_population)):
            change = yearly_population[i] - yearly_population[i-1]
```

```
            yearly_change.append(change)

            # Если это первый год, то присвоить отслеживающим
                    # переменным значение этого года.
            if i==1:
                greatest_increase = change
                smallest_increase = change
                greatest_year = 1
                smallest_year = 1

            # Это не первое изменение численности населения.
            # При необходимости обновить отслеживающие переменные.
            else:
                if change > greatest_increase:
                    greatest_increase = change
                    greatest_year = i

                elif change < smallest_increase:
                    smallest_increase = change
                    smallest_year = i

        total_change = float(sum(yearly_change))
        average_change = total_change / len(yearly_change)

        print('Среднегодовое изменение в численности ' \
              'во время этого периода времени составило',
                      format(average_change, ',.2f'))
        print('Год с максимальным ростом численности', \
              BASE_YEAR + greatest_year)
        print('Год с минимальным ростом численности', \
              BASE_YEAR + smallest_year)

    # Обработать любые ошибки, которые могут произойти.
    except IOError:
        print('Файл не найден')
    except IndexError:
        print('Ошибка индексации')
    except:
        print('Произошла ошибка')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.10. Чемпионы Мировой серии

```
def main():

    # Определить переменные
    team = ''
    win_count = 0
```

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

        # Прочитать все строки из файла в список.
        winners = input_file.readlines()

        # Отсечь все замыкающие символы новой строки.
        for i in range(len(winners)):
            winners[i] = winners[i].rstrip('\n')

        # Получить от пользователя входные данные.
        team = input('Введиет название команды: ')

        # Проверить, не выигрывала ли эта команда Мировую Серию.
        if team not in winners:
            print('Команда', team,
                          'никогда не выигрывала Мировую серию.')

        # Подсчитать количество побед команды в Мировой Серии.
        else:
            for item in winners:
                if item == team:
                    win_count += 1
            print('Команда', team, 'выигрывала Мировую серию', \
                  win_count, 'раз между 1903 и 2009 годами.')

    # Обработать любые ошибки, которые могут произойти.
    except IOError:
        print('Файл не найден')
    except IndexError:
        print('Ошибка индексации')
    except:
        print('Произошла ошибка')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.11. Магический квадрат Ло Шу

```
# Глобальные константы
ROWS = 3    # Количество строк
COLS = 3    # Количество столбцов
MIN = 1     # Значение наименьшего числа
MAX = 9     # Значение наибольшего числа

def main():
    # Создать двумерный список.
```

```
    test_list = [ [4, 9, 2],
                  [3, 5, 7],
                  [8, 1, 6] ]

    # Показать список в строковом и постолбцовом формате.
    display_square_list(test_list)

    # Определить, является ли список магическим квадратом Ло Шу.
    if is_magic_square(test_list):
        print('Это магический квадрат Ло Шу.')
    else:
        print('Это не магический квадрат Ло Шу.')

# Функция display_square_list принимает двумерный список
# в качестве аргумента и показывает значения списка в строковом
# и постолбцовом формате.
def display_square_list(value_list):
    for r in range(ROWS):
        for c in range(COLS):
            print(value_list[r][c], end=' ')
        print()

# Функция is_magic_square принимает двумерный список
# в качестве аргумента и возвращает True, если список удовлетворяет
# всем требованиям, предъявляемым к магическому квадрату.
# В противном случае она возвращает False.
def is_magic_square(value_list):
    # Исходно назначить состоянию значение False.
    status = False

    # Вызвать функции и сохранить возвращаемые из них значения.
    is_in_range = check_range(value_list)      # в диапазоне
    is_unique = check_unique(value_list)       # уникальные
    is_equal_rows = check_row_sum(value_list)  # равные строки
    is_equal_cols = check_col_sum(value_list)  # равные столбцы
    is_equal_diag = check_diag_sum(value_list) # равные диагонали

    # Оределить, удовлетворяет ли список всем требованиям.
    if is_in_range and \
       is_unique and \
       is_equal_rows and \
       is_equal_cols and \
       is_equal_diag:

        # Если да, то назначить переменной status значение True.
        status = True

    # Вернуть status.
    return status
```

```
# Функция check_range принимает двумерный список
# в качестве аргумента и возвращает True, если значения
# в списке лежат в пределах указанного диапазона.
# В противном случае она возвращает False.
def check_range(value_list):
    # Исходно назначить состоянию значение True.
    status = True

    # Перебрать все значения в списке.
    for r in range(ROWS):
        for c in range(COLS):
            # Определить, есть ли какие-либо значения,
            # которые лежат за пределами диапазона.
            if value_list[r][c] < MIN or \
               value_list[r][c] > MAX:
                # Если да, назначить status значение False.
                status = False

    # Вернуть status.
    return status

# Функция check_unique принимает двумерный список
# в качестве аргумента и возвращает True, если значения
# в списке уникальные.
# В противном случае она возвращает False.
def check_unique(value_list):
    # Исходно назначить состоянию значение True.
    status = True
    # Инициализировать переменную искомым значением.
    search_value = MIN
    # Инициализировать счетчик нулем.
    count = 0

    # Выполнять поиск до тех пор, пока не будет достигнуто
    # минимальное значение, и при этом значение является уникальным.
    while search_value <= MAX and status == True:
        # Перебрать все значения в списке.
        for r in range(ROWS):
            for c in range(COLS):
                # Определить, равняется ли текущее
                # значение искомому значению.
                if value_list[r][c] == search_value:
                    # Если да, нарастить счетчик.
                    count += 1
                # Определить, счетчик больше единицы?
                if count > 1:
                    # Если да, то значение не уникальное.
                    # Присвоить переменной status значение False.
```

```
                    status = False
        # Нарастить значение переменной search_value.
        search_value += 1
        # Обнулить переменную count.
        count = 0

    # Вернуть status.
    return status

# Функция check_row_sum принимает двумерный список
# в качестве аргумента и возвращает True, если сумма
# значений в каждой строке списка одинаковая.
# В противном случае она возвращает False.
def check_row_sum(value_list):
    # Исходно назначить состоянию значение True.
    status = True

    # Вычислить сумму значений в первой строке.
    sum_row_0 = value_list[0][0] + \
                value_list[0][1] + \
                value_list[0][2]

    # Вычислить сумму значений во второй строке.
    sum_row_1 = value_list[1][0] + \
                value_list[1][1] + \
                value_list[1][2]

    # Вычислить сумму значений в третьей строке.
    sum_row_2 = value_list[2][0] + \
                value_list[2][1] + \
                value_list[2][2]

    # Определить, не является ли сумма любой строки одинаковой.
    if (sum_row_0 != sum_row_1) or \
       (sum_row_0 != sum_row_2) or \
       (sum_row_1 != sum_row_2):
        # Если да, то присвоить переменной status значение False
        status = False

    # Вернуть status.
    return status

# Функция check_col_sum принимает двумерный список
# в качестве аргумента и возвращает True, если сумма
# значений в каждом столбце списка одинаковая.
# В противном случае она возвращает False.
def check_col_sum(value_list):
    # Исходно назначить состоянию значение True.
    status = True
```

```
    # Вычислить сумму значений в первом столбце.
    sum_col_0 = value_list[0][0] + \
                value_list[1][0] + \
                value_list[2][0]

    # Вычислить сумму значений во втором столбце.
    sum_col_1 = value_list[0][1] + \
                value_list[1][1] + \
                value_list[2][1]

    # Вычислить сумму значений в третьем столбце.
    sum_col_2 = value_list[0][2] + \
                value_list[1][2] + \
                value_list[2][2]

    # Определить, не является ли сумма любого столбца одинаковой
    if (sum_col_0 != sum_col_1) or \
       (sum_col_0 != sum_col_2) or \
       (sum_col_1 != sum_col_2):
        # Если да, то присвоить переменной status значение False
        status = False

    # Вернуть status.
    return status

# Функция check_diag_sum принимает двумерный список
# в качестве аргумента и возвращает True, если сумма
# значений в каждой диагонали списка одинаковая.
# В противном случае она возвращает False.
def check_diag_sum(value_list):
    # Исходно назначить состоянию значение True.
    status = True

    # Вычислить сумму значений в первой диагонали.
    sum_diag_0 = value_list[0][0] + \
                value_list[1][1] + \
                value_list[2][2]

    # Вычислить сумму значений во второй диагонали.
    sum_diag_1 = value_list[2][0] + \
                value_list[1][1] + \
                value_list[0][2]

    # Определить, не является ли сумма диагоналей одинаковой
    if sum_diag_0 != sum_diag_1:
        status = False

    # Вернуть status.
```

```
    return status

# Вызвать главную функцию.
main()
```

Упражнение по программированию 7.12. Генерация простого числа

```
# Функция prime_or_composite принимает целое число и
# показывает сообщение о том, является полученное значение
# простым или составным числом.

def prime_or_composite(n):
    has_divisor = False

    for i in range(2, n):
        if n % i == 0:
            has_divisor = True

    if has_divisor:
        print(n, 'является составным.')
    else:
        print(n, 'является простым.')

def main():
    # Получить от пользователя целое число.
    user_num = int(input('Введите целое число больше 1: '))

    # Создать пустой список.
    numbers = []

    # Заполнить список числами.
    for count in range(2, user_num + 1):
        numbers.append(count)

    # Определить, является ли каждый элемент простым или составным.
    for i in range(len(numbers)):
        prime_or_composite(numbers[i])

# Вызвать главную функцию
main()
```

Упражнение по программированию 7.13. Магический шар

```
import random

def main():
    # Открыть файл ответов 8_Ball_Response_ru.
       # Файл находится в подпапке data.
    response_file = open(r'data\8_ball_responses_ru.txt', 'r')
```

```
    # Прочитать содержимое файла в список.
    responses = response_file.readlines()

    # Закрыть файл.
    response_file.close()

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

    # Получить вопрос пользователя.
    question = input('Введите свой вопрос: ')

    # Показать произвольный ответ.
    print(responses[random.randint(0, len(responses))])

# Вызвать главную функцию
main()
```

Упражнение по программированию 7.14. Круговая диаграмма расходов

```
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.family']     = 'sans-serif'
rcParams['font.sans-serif'] = ['Ubuntu Condensed']
rcParams['figure.figsize']  = (4, 3.8)
rcParams['legend.fontsize'] = 10
rcParams['xtick.labelsize'] = 9
rcParams['ytick.labelsize'] = 9

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

    # Прочитать содержимое файла в список.
    expenses = expense_file.readlines()

    # Закрыть файл.
    expense_file.close()

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

    # Создать метки долей.
    slice_labels = ['Аренда', 'Топливо', 'Еда', 'Одежда', 'Техобслуживание авто', 'Прочее']

    # Создать круговую диаграмму из этих значений.
    plt.pie(expenses, labels=slice_labels)
```

```
    # Добавить заголовок.
    plt.title('Месячные расходы')

    # Показать круговую диаграмму.
    plt.show()

# Вызвать главную функцию
main()
```

Упражнение по программированию 7.15. График еженедельных цен на бензин за 1994 год

```
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.family']     = 'sans-serif'
rcParams['font.sans-serif'] = ['Ubuntu Condensed']
rcParams['figure.figsize']  = (7, 6)
rcParams['legend.fontsize'] = 10
rcParams['xtick.labelsize'] = 9
rcParams['ytick.labelsize'] = 9

# Именованная константа
NUM_WEEKS = 52

def main():
    # Открыть файл цен на бензин.
    # Файл находится в подпапке data
    gas_file = open(r'data\1994_Weekly_Gas_Averages.txt', 'r')

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

    # Закрыть файл.
    gas_file.close()

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

    # Создать список с номерами недель
       # (чтобы использовать в качестве координат X).
    x_coords = []
    for i in range(1, NUM_WEEKS + 1):
        x_coords.append(i)

    # Построить линейный график.
    plt.plot(x_coords, gas)

    # Ограничить ось x диапазоном от 1 до 52.
```

```
    plt.xlim(xmin=1, xmax=NUM_WEEKS)

    # Добавить заголовок.
    plt.title('Средненедельные цены на бензин в 1994 г.')

    # Добавить метки к осям.
    plt.xlabel('Недели (по номеру)')
    plt.ylabel('Средние цены')

    # Показать график.
    plt.show()

# Вызвать главную функцию
main()
```