Упражнения по программированию главы 9

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

Упражнение по программированию 9.1. Информация об учебных курсах

```
def main():
    # Инициализировать словари
    rooms = {'CS101':3004, 'CS102':4501, 'CS103':6755,
             'NT110':1244, 'CM241':1411}

    instructors = {'CS101':'Хайнс', 'CS102':'Альварадо',
                   'CS103':'Рич', 'NT110':'Берк',
                   'CM241':'Ли'}

    times = {'CS101':'8:00', 'CS102':'9:00',
             'CS103':'10:00', 'NT110':'11:00',
             'CM241':'12:00'}

    course = input('Введите номер курса: ')

    if course not in rooms:
        print(course, 'не является допустимым номером курса.')
    else:
        print(f'Подробная информация о курсе {course}:')
        print('Аудитория:', rooms[course])
        print('Преподаватель:', instructors[course])
        print('Время:', times[course])

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.2. Викторина со столицами

```
import random

def main():
    # Инициализировать словари
    capitals = {'Алабама':'Монтгомери', 'Аляска':'Джуно',
                'Аризона':'Феникс', 'Арканзас':'Литтл Рок',
                'Калифорния':'Сакраменто', 'Колорадо':'Денвер',
                'Коннектикут':'Хартфорд', 'Делавэр':'Довер',
                'Флорида':'Таллахасси', 'Джорджия':'Атланта',
                'Гавайи':'Гонолулу', 'Айдахо':'Бойсе',
                'Иллинойс':'Спрингфилд', 'Индиана':'Индианаполис',
                'Айова':'Де-Мойн', 'Канзас':'Топика',
                'Кентукки':'Франкфорт', 'Луизиана':'Батон-Руж',
                'Мэн':'Батон-Руж', 'Мэриленд':'Аннаполис',
                'Аннаполис':'Бостон', 'Мичиган':'Лансинг',
                'Миннесота':'Сен-Пол', 'Миссисипи':'Джэксон',
```

```
            'Миссури':'Джефферсон-Сити', 'Монтана':'Хелена',
            'Небраска':'Линкольн', 'Невада':'Карсон-Сити',
            'Нью-Гэмпшир':'Конкорд', 'Нью-Джерси':'Трентон',
            'Нью-Мексико':'Санта-Фе', 'Нью-Йорк':'Олбани',
            'Серерная Каролина':'Роли', 'Северная Дакота':'Бисмарк',
            'Огайо':'Колумбус', 'Оклахома':'Оклахома-сити',
            'Орегон':'Сейлем', 'Пенсильвания':'Гаррисберг',
            'Род-Айленд':'Провиденс', 'Южная Каролина':'Колумбия',
            'Южная Дакота':'Пирр', 'Теннеси':'Нэшвилл',
            'Техас':'Остин', 'Юта':'Солт-Лейк-Сити',
            'Вермонт':'Монтпилиер', 'Виргиния':'Ричмонд',
            'Вашингтон':'Олимпия', 'Западная Виргиния':'Чарлстон',
            'Висконсин':'Мэдисон', 'Вайоминг':'Шайен'}

# Локальные переменные
correct = 0
wrong = 0
next_question = True
index = 0
user_solution = ''

# Продолжать до тех пор, пока пользователь не выйдет из игры.
while next_question:

    # Получить доступ к списку названий штатов.
    state_iterator = iter(capitals)

    # Получить произвольное название штата для вопроса.
    index = (random.randint(1,50) - 1)
    for i in range (index-1):
        temp = state_iterator.__next__()
    current_state = str(state_iterator.__next__())

    # Получить решение пользователя.
    user_solution = input('Назовите столицу штата ' + \
                          current_state + \
                          ' (либо введите 0, чтобы выйти): ')
    # Пользователь желает выйти из игры.
    if user_solution == '0':
        next_question = False
        print('Вы дали', correct, 'правильных и',\
              wrong, 'неправильных ответов.')
    # Решение пользователя правильное.
    elif user_solution == capitals[current_state]:
        correct = correct + 1
        print('Правильно.')
    # Решение пользователя неправильное.
    else:
        wrong = wrong + 1
```

```
            print('Неправильно.')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.3 (часть 1). Шифрование и дешифрование файлов

```
# Шифрование и дешифрование - это обратные задачи
CODE = {'A':')','a':'0','B':'(','b':'9','C':'*','c':'8',\
        'D':'&','d':'7','E':'^','e':'6','F':'%','f':'5',\
        'G':'$','g':'4','H':'#','h':'3','I':'@','i':'2',\
        'J':'!','j':'1','K':'Z','k':'z','L':'Y','l':'y',\
        'M':'X','m':'x','N':'W','n':'w','O':'V','o':'v',\
        'P':'U','p':'u','Q':'T','q':'t','R':'S','r':'s',\
        'S':'R','s':'r','T':'Q','t':'q','U':'P','u':'p',\
        'V':'O','v':'o','W':'N','w':'n','X':'M','x':'m',\
        'Y':'L','y':'l','Z':'K','z':'k','!':'J','1':'j',\
        '@':'I','2':'i','#':'H','3':'h','$':'G','4':'g',\
        '%':'F','5':'f','^':'E','6':'e','&':'D','7':'d',\
        '*':'C','8':'c','(':'B','9':'b',')':'A','0':'a',\
        ':':',',',':':',','?':'.','.':'?','<':'>','>':'<',\
        "'":'"','"':"'",'+':'-','-':'+','=':';',';':'=',\
        '{':'[','[':'{','}':']',']':'}'}

def main():
    # Получить строковое значение преобразованного текста.
    result = convert()

    # Открыть файл и записать в него.
    output_name = input('Введите имя выходного файла: ')

       # Файл будет находиться в подпапке data
    output_file = open('data\\' + output_name, 'w')
    output_file.write(result)
    output_file.close()

# Функция convert запрашивает у пользователя имя файла, открывает
# файл и преобразует его содержимое, используя словарь CODE.
# Затем она возвращает строку преобразованного текста.
def convert():
    input_name = input('Введите имя входного файла: ')

    # Файл находится в подпапке data
    input_file = open('data\\' + input_name, 'r')

    result = ''
    text = input_file.read()

    # Если символ является пробелом, он не преобразовывается;
```

```
    # в противном случае преобразовать.
    for ch in text:
        if ch.isspace():
            result = result + ch
        else:
            result = result + CODE[ch]

    return result

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.3 (часть 2). Шифрование и дешифрование файлов

```
# Шифрование и дешифрование - это обратные задачи
CODE = {'A':')','a':'0','B':'(','b':'9','C':'*','c':'8',\
        'D':'&','d':'7','E':'^','e':'6','F':'%','f':'5',\
        'G':'$','g':'4','H':'#','h':'3','I':'@','i':'2',\
        'J':'!','j':'1','K':'Z','k':'z','L':'Y','l':'y',\
        'M':'X','m':'x','N':'W','n':'w','O':'V','o':'v',\
        'P':'U','p':'u','Q':'T','q':'t','R':'S','r':'s',\
        'S':'R','s':'r','T':'Q','t':'q','U':'P','u':'p',\
        'V':'O','v':'o','W':'N','w':'n','X':'M','x':'m',\
        'Y':'L','y':'l','Z':'K','z':'k','!':'J','1':'j',\
        '@':'I','2':'i','#':'H','3':'h','$':'G','4':'g',\
        '%':'F','5':'f','^':'E','6':'e','&':'D','7':'d',\
        '*':'C','8':'c','(':'B','9':'b',')':'A','0':'a',\
        ':':',',',':':',','?':'.','.':'?','<':'>','>':'<',\
        "'":'"','"':"'",'+':'-','-':'+','=':';',';':'=',\
        '{':'[','[':'{','}':']',']':'}'}

def main():
    # Получить строковое значение преобразованного текста.
    result = convert()

    # Написать преобразованный текст на экран.
    print(result)

# Функция convert запрашивает у пользователя имя файла, открывает
# файл и преобразует его содержимое, используя словарь CODE.
# Она затем возвращает строку преобразованного текста.
def convert():
    input_name = input('Введите имя входного файла: ')

    # Файл находится в подпапке data
    input_file = open('data\\' + input_name, 'r')

    result = ''
    text = input_file.read()
```

```
    # Если символ является пробелом, он не преобразовывается;
    # в противном случае преобразовать.
    for ch in text:

        if ch.isspace():
            result = result + ch
        else:
            result = result + CODE[ch]

    return result

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.4. Уникальные слова

```
def main():
    # Получить имя входного файла.
    input_name = input('Введите имя входного файла: ')

    # Открыть входной файл и прочитать текст.
    # Файл находится в подпапке data
    input_file = open('data\\' + input_name, 'r')

    text = input_file.read()
    words = text.split()

    # Создать множество уникальных слов.
    unique_words = set(words)

    # Напечатать результаты.
    print('Это уникальные слова в тексте:')
    for word in unique_words:
        print(word)

    # Закрыть файл.
    input_file.close()

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.5. Частота слов

```
def main():
    # Задать пустой словарь
    counter = {}

    # Получить входной текст
    input_name = input('Введите имя входного файла: ')
```

```
    # Файл находится в подпапке data
    input_file = open('data\\' + input_name, 'r')

    text = input_file.read()
    words = text.split()

    # Добавить каждое уникальное слово в словарь со счетчиком 0
    unique_words = set(words)
    for word in unique_words:
        counter[word] = 0

    # По каждому слову в тексте увеличить его счетчик в словаре
    for item in words:
        counter[item] += 1

    # Показать результаты
    print(format('слово', '15'),'\t',format('появления','15'))
    print('-----------------------------------------------')
    while len(counter)>0:

        pair = counter.popitem()
        print(format(pair[0],'15'), format(pair[1], '15'))

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.6. Анализ файла

```
def main():
    # Получить входной текст первого файла и создать множество,
    # содержащее его уникальные слова
    input_name = input('Введите имя первого входного файла: ')
    # Файл находится в подпапке data
    file1 = open('data\\' + input_name, 'r')
    text1 = file1.read()
    file1.close()
    words1 = text1.split()
    set1 = set(words1)

    # Получить входной текст второго файла и создать множество,
    # содержащее его уникальные слова
    input_name = input('Введите имя второго входного файла: ')
    # Файл находится в подпапке data
    file2 = open('data\\' + input_name, 'r')
    text2 = file2.read()
    file2.close()
    words2 = text2.split()
    set2 = set(words2)

    # Получить объединение множеств и напечатать его значения.
```

```
    union = set1.union(set2)
    print('Эти уникальные слова теперь содержатся в обоих файлах:')
    for item in union:
        print(item)
    print()

    # Получить пересечение множеств и напечатать его значения
    intersection = set1.intersection(set2)
    print('Эти слова встречаются в обоих файлах:')
    for item in intersection:
        print(item)
    print()

    # Получить разность между множеством set1 и множеством set2 и
    # напечатать его значения
    difference1 = set1.difference(set2)
    print('Эти слова встречаются в первом файле и ' \
          'не встречаются во втором файле:')
    for item in difference1:
        print(item)
    print()

    # Получить разность между множеством set2 и множеством set1 и
    # напечатать его значения
    difference2 = set2.difference(set1)
    print('Эти слова встречаются во втором файле и ' \
          'не встречаются во первом файле:')
    for item in difference2:
        print(item)
    print()

    # Получить симметрическую разность между множеством set1 и
    # множеством set2 и напечатать его значения
    sym_diff = set1.symmetric_difference(set2)
    print('Эти слова встречаются в первом файле или ' \
          'втором файле и не встречаются в ' \
          'обоих файлах:')
    for item in sym_diff:
        print(item)
    print()

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.7. Победители Мировой серии

```
# Глобальная константа для базового года.
BASE_YEAR = 1903

def main():
```

```
    # Локальные словарные переменные.
    year_dict = {}
    count_dict = {}

    # 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()

    # Заполнить словари информацией о командах.
    for i in range(len(winners)):
        team = winners[i].rstrip('\n')

        # Выяснить, в каком году команда стала победителем
        # (приняв в расчет пропущенные годы).
        year = BASE_YEAR + i
        if year >= 1904:
            year += 1
        if year >= 1994:
            year += 1

        # Добавить информацию в словарь year_dict.
        year_dict[str(year)] = team

        # Обновить словарь count_dict.
        if team in count_dict:
            count_dict[team] += 1
        else:
            count_dict[team] = 1

    # Получить от пользователя входные данные.
    year = input('Введите год в диапазоне 1903-2009: ')

    # Напечатать результаты.
    if year == '1904' or year == '1994':
        print(f'Мировая серия не проводилась в {year} году.')
    elif year<'1903' or year > '2009':
        print(f'Данные за {year} год не включены в базу данных.')
    else:
        winner = year_dict[year]
        wins = count_dict[winner]
        print('Командой, которая стала победителем Мировой серии в ', \
              year, ' году, была ', winner, '.', sep='')
        print('Они побеждали в Мировой серии', wins, 'раз.')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.8. Имена и адреса электронной почты

```
import pickle

# Глобальные константы для элементов меню.
LOOK_UP = 1
ADD = 2
CHANGE = 3
DELETE = 4
QUIT = 5

# Глобальная константа с именем файла.
# Файл находится в подпапке data.
FILENAME = r'data\emails.dat'

# Главная функция
def main():
    # Получить словарь email.
    emails = load_emails()

    # Инициализировать переменную для выбора пользователя.
    choice = 0

    # Обрабатывать запросы пользователя до тех пор,
       # пока пользователь не завершит работу.
    while choice != QUIT:

        choice = get_user_choice()

        if choice== LOOK_UP:
            look_up(emails)
        elif choice == ADD:
            add(emails)
        elif choice == CHANGE:
            change(emails)
        elif choice == DELETE:
            delete(emails)

    # Законсервировать результирующий словарь.
    save_emails(emails)

    print('Информация сохранена.')

def load_emails():
    try:
        # Открыть файл.
        input_file = open(FILENAME, 'rb')

        # Расконсервировать словарь.
```

```
        email_dict = pickle.load(input_file)

        # Закрыть словарь.
        input_file.close()

    # Не получилось открыть словарь.
    except IOError:
        # Создать пустой словарь.
        email_dict = {}

    # Вернуть словарь.
    return email_dict

def get_user_choice():
    # Показать меню, получить выбор пользователя и
    # проверить его на допустимость.
    print()
    print('Меню')
    print('-----------------------------------------')
    print('1. Найти электронный адрес')
    print('2. Добавить новое имя и электронный адрес')
    print('3. Изменить существующий электронный адрес')
    print('4. Удалить имя и электронный адрес')
    print('5. Выйти из программы')
    print()

    choice = int(input('Введите свой вариант: '))

    # Проверить выбранный вариант.
    while choice < LOOK_UP or choice>QUIT:
        choice = int(input('Введенный вами вариант неправильный. ' \
                           'Пожалуйста, введите правильный вариант: '))

    # Вернуть выбранный пользователем вариант.
    return choice

def look_up(emails):
    # Получить искомое имя.
    name = input('Введите имя: ')

    # Нацти имя в словаре.
    if name in emails:
        print('Имя:', name)
        print('Электронная почта:', emails[name])
    else:
        print('Указанное имя не найдено.')

def add(emails):
    # Получить имя и электронный адрес.
```

```
    name = input('Введите имя: ')
    address = input('Введите электронный адрес: ')

    # Добавить новый адрес, если имя не существует.
    # В противном случае уведомить пользователя, что имя существует.
    if name not in emails:
        emails[name] = address
        print('Имя и электронный адрес были добавлены.')
    else:
        print('Это имя уже существует.')

def change(emails):
    # Получить имя для обновления информации.
    name = input('Введите имя: ')

    # Изменить адрес, если имя существует.
    # В противном случае уведомить пользователя, что не имя существует.
    if name in emails:
        address = input('Введите новый адрес: ')
        emails[name] = address
        print('Информация обновлена.')
    else: # name not found
        print('Указанное имя не найдено.')

def delete(emails):
    # Получить имя для удаления.
    name = input('Введите имя: ')

    if name in emails:

        del emails[name]
        print('Информация удалена.')

    else: # имя не найдено
        print('Указанное имя не найдено.')

# Функция консервирует указанный словарь и
# сохраняет его в файле emails.
def save_emails(emails):
    # Открыть файл для записи.
    output_file = open(FILENAME, 'wb')

    # Законсервировать словарь и его сохранить.
    pickle.dump(emails, output_file)

    # Закрыть файл.
    output_file.close()

# Вызвать главную функцию.
```

```
main()
```

Упражнение по программированию 9.9. Имитация игры в блек-джек

```
# Глобальная константа для выигрышного количества карт.
MAX = 21

# Главная функция.
def main():
    # Локальные переменные.
    hand1 = 0
    hand2 = 0
    deck = create_deck()

    while hand1 <= MAX and hand2 <= MAX:

        # Раздать карты каждому игроку и вычислить стоимость
            # комбинации карт на руках.
        card1, value1 = deck.popitem()
        hand1 = update_hand_value(hand1, value1, card1)

        card2, value2 = deck.popitem()
        hand2 = update_hand_value(hand2, value2, card2)

        print('Игроку 1 сдана карта', card1)
        print('Игроку 2 сдана карта', card2)
        print()

    # Determine the winner.
    if hand1 > MAX and hand2 > MAX:
        print("Победителя нет.")
    elif hand1 > 21:
        print("Игрок 2 выиграл.")
    else:
        print("Игрок 1 выиграл.")

# Функция create_deck создает колоду карт и возвращает колоду.
def create_deck():
    # Задать локальные переменные.
    suits = ['пик','червей','крестей','бубей']
    special_values = {'туз':1, 'король':10, 'дама':10, 'валет':10}

    # Создать список всех достоинств карт.
    numbers = ['туз', 'король', 'дама', 'валет']
    for i in range(2,11):
        numbers.append(str(i))

    # Инициализировать колоду.
    deck = {}
    for suit in suits:
```

```
        for num in numbers:
            # Значения 2-10.
            if num.isnumeric():
                deck[num + ' ' + suit] = int(num)
            # Туз, король, дама или валет.
            else:
                deck[num + ' ' + suit] = special_values[num]
    return deck

def update_hand_value(hand, value, card):
    if not card.startswith('Туз'):
        return hand+value
    # Добавление 11 вызовет превышение максимума.
    elif hand > 10:
        # По умолчанию значение 1.
        return hand + value
    else:
        return hand + 11

# Вызвать главную функцию.
main()
```

Упражнение по программированию 9.10. Словарный индекс

```
# Функция get_word_dict возвращает словарь, содержащий
# слова из списка line_list в качестве ключей и
# их номера строк в качестве значений.
def get_word_dict(line_list):
    # Создать счетчик строк.
    count = 0

    # Создать словарь для слов.
    word_dict = {}

    # Обойти в цикле список строк.
    for e in line_list:
        # Разбить строку на слова.
        words = e.split(' ')

        # Обойти в цикле список слов.
        for w in words:
            # Если слов есть словаре...
            if w in word_dict:
                # Обновить существующий элемент.
                word_dict[w].add(count + 1)
            else:
                # В противном случае сохранить слово в словаре.
                word_dict[w] = set([count + 1])

        # Обновить счетчик.
```

```
        count += 1

    # Вернуть словарь.
    return word_dict

# Функция write_index_file записывает индексный файл,
# содержащий элементы словаря word_dict.
def write_index_file(word_dict):
    # Открыть файл.
       # Файл находится в подпапке data.
    outputfile = open(r'data\index.txt', 'w')

    # Записать элементы словаря.
    for key in word_dict:
        # Записать слово.
        outputfile.write(key + ': ')
        # Записать номера строк.
        for val in word_dict[key]:
            outputfile.write(str(val) + ' ')
        # Записать символ новой строки.
        outputfile.write('\n')

    # Закрыть файл.
    outputfile.close()

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

    # Прочитать содержимое файла в список.
    line_list = inputfile.readlines()

    # Закрыть файл.
    inputfile.close()

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

    # Получить словарь, содержащий слова и их номера строк.
    word_dict = get_word_dict(line_list)

    # Записать индексный файл.
    write_index_file(word_dict)

# Вызвать главную функцию.
main()
```