Упражнения по программированию главы 10

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

Упражнение по программированию 10.1. Класс *Pet*

```
# import pet
from objects import pet  # класс хранится в папке objects

def main():
    # Локальные переменные
    pet_name = ""
    pet_type = ""
    pet_age = 0

    # Получить данные о домашнем животном.
    pet_name = input('Введите кличку животного: ')
    pet_type = input('Введите вид животного: ')
    pet_age = int(input('Введите возраст животного: '))

    # Создать экземпляр класса Pet.
    mypet = pet.Pet(pet_name, pet_type, pet_age)

    # Показать введенные данные.
    print ('\nВот данные, которые Вы ввели: ')
    print (' - кличка животного: ', mypet.get_name())
    print (' - вид животного: ', mypet.get_animal_type())
    print (' - возраст животного: ', mypet.get_age())

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.2. Класс *Car*

```
# import car
from objects import car  # класс хранится в папке objects

def main():
    # Создать экземпляр класса Car.
    my_car = car.Car('2008', 'Honda Accord')

    # Ускориться 5 раз.
    print('Автомобиль ускоряется: ')
    for i in range(5):
        my_car.accelerate()
        print ('Текущая скорость: ', my_car.get_speed())
    print()

    # Притормозить 5 раз.
```

```
    print ('Автомобиль замедляется: ')
    for i in range(5):
        my_car.brake()
        print ('Текущая скорость: ', my_car.get_speed())

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.3. Класс персональных данных *Information*

```
# import info
from objects import info  # класс хранится в папке objects

def main():
    # Создать три экземпляра класса Information.
    my_info = info.Information('Джон Доу', '111 Моя улица', 22, '555-555-1281')
    mom_info = info.Information('Мать', '222 Мамина улица', 52, '555-555-1234')
    sister_info = info.Information('Джейн Доу', '333 Ее улица', 20, '555-555-4444')

    print('Информация обо мне:')
    display_info(my_info)
    print()

    print("Информация о матери:")
    display_info(mom_info)
    print()

    print ("Информация о сестре:")
    display_info(sister_info)

def display_info(info):
    print(' Имя:     ', info.get_name())
    print(' Адрес:   ', info.get_address())
    print(' Возраст: ', info.get_age())
    print(' Телефон: ', info.get_phone_number())

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.4. Класс *Employee*

```
# import emp
from objects import emp  # класс хранится в папке objects

def main():
    # Создать три экземпляра класса Employee
    emp1 = emp.Employee('Сьюзан Мейерс', '47899', 'Бухгалтерия', 'Вице-президент')
```

```
    emp2 = emp.Employee('Марк Джоунс', '39119', 'IT', 'Программист')
    emp3 = emp.Employee('Джой Роджерс', '81774', 'Производственный', 'Инженер')

    print('Сотрудник 1:')
    print(emp1)
    print()
    print('Сотрудник 2:')
    print(emp2)
    print()
    print('Сотрудник 3:')
    print(emp3)

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.5. Класс *RetailItem*

```
# import retail
from objects import retail  # класс хранится в папке objects

def main():
    # Создать три экземпляра класса RetailItem.
    item1 = retail.RetailItem('Пиджак', 12, 59.95)
    item2 = retail.RetailItem('Дизайнерские джинсы', 40, 34.95)
    item3 = retail.RetailItem('Рубашка', 20, 24.95)



    # Показать информацию.
    print ('Товарная позиция 1: ')
    print (item1)
    print()
    print ('Товарная позиция2:')
    print (item2)
    print()
    print ('Товарная позиция 3:')
    print (item3)

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.6. Расходы на лечение

```
# import procedure
# import patient
from objects import procedure  # классы хранятся в папке objects
from objects import patient

def main():
```

```
    procedure_1 = procedure.Procedure('Врачебный осмотр', '8-24-2022', 'Ирвин', 250.0)
    procedure_2 = procedure.Procedure('Рентгенография', '8-24-2022', 'Джемисон', 500.0)
    procedure_3 = procedure.Procedure('Анализ крови', '8-24-2022', 'Смит', 200.0)

    pat = patient.Patient('Джеймс', 'Эдвард', 'Джоунс', '123 Мэйн стрит',
                          'Биллингс', 'Монтана', '59000', '406-555-1212',
                          'Дженни Джоунс Jones', '406-555-1213')

    print(pat)
    print(procedure_1)
    print(procedure_2)
    print(procedure_3)

# Вызвать главную функцию
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.7. Система управления персоналом

```
# import emp
from objects import emp  # класс хранится в папке objects
import pickle

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

# Глобальная константа для имени файла.
# Файл находится в подпапке data.
FILENAME = r'data\employees.dat'

# Главная функция
def main():

    # Получить словарь сотрудников.
    employees = load_employees()

    # Инициализировать переменную для выбора пользователя.
    choice = 0

    # Обрабатывать запросы пользователя до тех пор,
    # пока пользователь не выйдет из программы.
    while choice != QUIT:

        choice = get_user_choice()

        if choice== LOOK_UP:
```

```
            look_up(employees)
        elif choice == ADD:
            add(employees)
        elif choice == CHANGE:
            change(employees)
        elif choice == DELETE:
            delete(employees)

    # Законсервировать результирующий словарь.
    save_employees(employees)

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

        # Расконсервировать словарь.
        employee_dict = pickle.load(input_file)

        # Закрыть файл.
        input_file.close()
    except IOError:
        # Не получилось открыть файл.
        # Создать пустой словарь.
        employee_dict = {}

    return employee_dict

def get_user_choice():
    # Показать меню, получить выбор пользователя и
       # проверить его допустимость.
    print()
    print('Меню')
    print('-------------------------------------')
    print('1. Найти сотрудника')
    print('2. Добавить нового сотрудника')
    print('3. Изменить существующего сотрудника')
    print('4. Удалить сотрудника')
    print('5. Выйти из программы')
    print()

    choice = int(input('Введите выбранный пункт меню: '))

    # Проерить выбор.
    while choice < LOOK_UP or choice > QUIT:
        choice = int(input('Выбранный Вами пункт меню недопустимый.' \
                           ' Пожалуйста, введите пункт меню: '))

    # Вернуть выбор пользователя.
```

```
    return choice

def look_up(employees):
    # Получить идентификационный номер сотрудника для поиска.
    ID = input('Введите идентификационный номер сотрудника: ')

    # Отыскать идентификатор в словаре. Если найден, то
    # данные будут распечатаны с помощью метода employee __str__;
    # в противном случае распечатать указанное сообщение.
    print(employees.get(ID, "Указанный идентификатор не найден"))

def add(employees):
    # Получить информацию о сотруднике.
    name = input('Введите имя сотрудника: ')
    ID = input('Введите идентификатор сотрудника: ')
    department = input('Введите отдел сотрудника: ')
    title = input('Введите должность сотрудника: ')

    new_emp = emp.Employee(name, ID, department, title)

    # Добавить нового сотрудника, если идентификатор не существует.
    # В противном случае уведомить пользователя, что
       # идентификатор существует.
    if ID not in employees:
        employees[ID] = new_emp
        print('Новый сотрудник был добавлен.')
    else:
        print('Сотрудник с этим идентификатором уже существует.')

def change(employees):
    # Получить обновленную информацию о сотруднике.
    ID = input('Введите идентификатор сотрудника: ')

    # Изменить информацию о сотруднике, если идентификатор существует.
    # В противном случае уведомить пользователя, что идентификатор
       # не существует.
    if ID in employees:
        name = input('Введите новое имя: ')
        department = input('Введите новый отдел: ')
        title = input('Введите новую должность: ')

        new_emp = emp.Employee(name, ID, department, title)

        employees[ID] = new_emp
        print('Информация о сотруднике обновлена.')
    # Идентификатор не найден.
    else:
        print('Указанный идентификатор не найден.')
```

```
def delete(employees):
    # Получить обновленную информацию о сотруднике.
    ID = input('Введите идентификатор сотрудника: ')

    # Изменить информацию о сотруднике, если идентификатор существует.
    # В противном случае уведомить пользователя, что идентификатор
       # не существует.
    if ID in employees:
        del employees[ID]
        print('Информация о сотруднике удалена.')
    # Идентификатор не найден.
    else:
        print('Указанный идентификатор не найден.')

# Функция консервирует указанный словарь и
# сохраняет его в файле с данными о сотрудниках.
def save_employees(employees):
    # Открыть файл для записи.
    output_file = open(FILENAME, 'wb')

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

    # Закрыть файл.
    output_file.close()

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.8. Класс *CashRegister*

```
# import cashRegister
from objects import cashRegister  # классы хранятся в папке objects

# Константы для вариантов покупаемых товаров
PANTS = 1
SHIRT = 2
DRESS = 3
SOCKS = 4
SWEATER = 5

# Главный метод
def main():

    # Создать продаваемые товары.
    pants = retail.RetailItem('Брюки', 10, 19.99)
    shirt = retail.RetailItem('Рубашка', 15, 12.50)
    dress = retail.RetailItem('Платье', 3, 79.00)
    socks = retail.RetailItem('Носки', 50, 1.00)
```

```
    sweater = retail.RetailItem('Свитер', 5, 49.99)

    # Создать словарь продаваемых товаров.
    sale_items = {PANTS:pants, SHIRT:shirt,
                  DRESS:dress, SOCKS:socks,
                  SWEATER:sweater}

    # Создать кассовый аппарат.
    register = cashRegister.CashRegister()

    # Инициализировать проверку цикла.
    checkout = 'Н'

    # Пользователь хочет приобрести дополнительные товары:
    while checkout == 'Н':

        user_choice = get_user_choice()
        item = sale_items[user_choice]
        if item.get_inventory() == 0:
            print('Этого товара нет в наличии.')
        else:
            register.purchase_item(item)

            # Обновить товарную позицию
            new_item = retail.RetailItem(item.get_description(), \
                                         item.get_inventory()-1, \
                                         item.get_price())
            sale_items[user_choice] = new_item

            checkout = input('Вы готовы оформить покупку (Д/Н)? ')

    print()
    print('Сумма Вашей покупки составляет: ', \
          format(register.get_total(), '.2f'))
    print()
    register.show_items()
    register.clear()

def get_user_choice():
    print('Menu')
    print('---------------------------------')
    print('1. Брюки')
    print('2. Рубашка')
    print('3. Платье')
    print('4. Носки')
    print('5. Свитер')
    print()

    choice = int(input('Введите пункт меню товара, ' + \
```

```
                    'который вы хотели бы приобрести: '))
    print()

    while choice > SWEATER or choice < PANTS:

        choice = int(input('Введите допустимый номер товара: '))
        print()

    return choice

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```

Упражнение по программированию 10.9. Викторина

```
# import question
from objects import question  # класс хранится в папке objects

def main():
    # Локальные переменные
    first_points = 0
    second_points = 0
    player = ''

    # Создать список вопросов.
    questions = get_questions()

    for i in range(10):

        if i % 2 == 0:
            player = 'Один'
        else:
            player = 'Два'
        print('Вопрос для игрока ', player, ':')

        current = questions[i]
        print(current)
        user_answer = int(input('Введите ваше решение (номер между 1 и 4): '))
        if current.isCorrect(user_answer):
            if player == 'Один':
                first_points += 1
            else:
                second_points += 1
            print('Это правильный ответ.')
            print()
        else:
            print('Неправильно. Правильный отет', current.get_solution())
            print()
```

```
    print('Первый игрок заработал', first_points, 'очков.')
    print('Второй игрок заработал', second_points, 'очков.')
    if first_points == second_points:
        print('Ничья.')
    elif first_points > second_points:
        print('Первый игрок побеждает в игре.')
    else:
        print('Второй игрок побеждает в игре.')

def get_questions():

    questions = []

    # Создать словарь вопросов и добавить в список.
    question1 = question.Question('Сколько дней в лунном ' + \
                                  'году?', '354', '365', \
                                  '243', '379', 1)
    questions.append(question1)
    question2 = question.Question('Какая самая большая планета?', \
                                  'Марс', 'Юпитер', 'Земля', \
                                  'Плутон', 2)
    questions.append(question2)
    question3 = question.Question('Какой кит самый большой?', \
                                  'Косатка', 'Горбатый кит', \
                                  'Белуга', 'Синий кит', 4)
    questions.append(question3)
    question4 = question.Question('Какой динозавр мог летать?', \
                                  'Трицератопс', 'Тираннозавр', \
                                  'Птеранодон', 'Диплодок', 3)
    questions.append(question4)
    question5 = question.Question('Какой из этих героев книги о Винни Пухе является осликом?', \
                                  'Пух', 'Иа-Иа', 'Пятачок', 'Кенга', 2)
    questions.append(question5)
    question6 = question.Question('Какая из перечисленных планет самая жаркая?', \
                                  'Марс', 'Плутон', 'Земля', \
                                  'Венера', 4)
    questions.append(question6)
    question7 = question.Question('У какого динозавра был самый ' + \
                                  'большой мозг по сравнению с телом?',  \
                                                'Троодон', 'Стегозавр', 'Ихтиозавр',
'Гиганторапор', 1)
    questions.append(question7)
    question8 = question.Question('Какой из пингвинов самый крупный?', \
                                  'Антарктический пингвин', \
                                  'Золотоволосый пингвин', \
                                  'Императорский пингвин', \
                                  'Белокрылый пингвин', 3)
    questions.append(question8)
    question9 = question.Question('В какой сказке героем является обезъянка?', \
```

```
                                    'Винни Пух', 'Любопытный Джордж', 'Хортон', \
                                    'Гуфи', 2)
    questions.append(question9)
    question10 = question.Question('Сколько длится год на Марсе?', \
                                    '550 земных дней', \
                                    '498 земных дней', \
                                    '126 земных дней', \
                                    '687 земных дней', 4)
    questions.append(question10)

    return questions

# Вызвать главную функцию.
if __name__ == '__main__':
    main()
```