Лабораторное занятие 13
Разработка классов

1 Цель работы

1.1 Приобрести навыки по составлению рекурсивных функций.

2 Литература

2.1 Прохоренок, Н.А. Python 3. Самое необходимое / Н.А. Прохоренок, В.А. Дронов. – Санкт-Петербург: БХВ-Петербург, 2016. – с.18-50.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание практической работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Создать класс "Работник" с закрытыми атрибутами:

– номер;

– ФИО;

– возраст;

– должность.

5.2 Создать конструктор для создания объекта класса «Работник». Создать метод с названием «Работать», выводящий «Сотрудник имя_сотрудника работает».

5.3 Написать программу, выполняющую следующие действия:

– ввод с клавиатуры данных о работниках. Количество работников задается пользователем. Сохранить данные о работниках в текстовой файл;

– вывод на экран информацию обо всех работниках;

– вывод на экран информации о работниках, возраст которых больше двадцати пяти. Информация должна быть упорядочена по возрасту работников. Если таких работников нет, то программа должна выдать сообщение об этом.

6 Порядок выполнения работы

6.1 Запустить Python IDLE и выполнить все задания из п.5.

6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Что такое ООП?
8.2 Для чего используются атрибуты у класса?
8.3 Для чего используются методы?
8.4 Что такое конструктор и что будет, если его не создать?
8.5 Какие есть принципы ООП и что они значат?

9. Приложение

В языке Python класс определяется с помощью ключевого слова class:

```
class название_класса:
    атрибуты_класса
    методы_класса
```

Для создания объекта класса используется конструктор. Мы можем явным образом определить в классах конструктор с помощью специального метода, который называется __init__() (по два прочерка с каждой стороны).

```
class Person:
    # конструктор
    def __init__(self):
        print("Создание объекта Person")

tom = Person()      # Создание объекта Person
```

Атрибуты хранят состояние объекта. Для определения и установки атрибутов внутри класса можно применять слово self. Например, определим следующий класс Person:

```
class Person:

    def __init__(self, name, age):
        self.name = name        # имя человека
        self.age = age          # возраст человека


tom = Person("Tom", 22)

tom.company = "Microsoft"
print(tom.company)  # Microsoft
```

Методы класса фактически представляют функции, которые определенны внутри класса и которые определяют его поведение. Например, определим класс Person с одним методом:

```
class Person:

    def __init__(self, name, age):
        self.name = name        # имя человека
        self.age = age          # возраст человека

    def display_info(self):
        print(f"Name: {self.name}  Age: {self.age}")


tom = Person("Tom", 22)
tom.display_info()      # Name: Tom  Age: 22

bob = Person("Bob", 43)
bob.display_info()      # Name: Bob  Age: 43
```

Язык программирования Python позволяет определить приватные или закрытые атрибуты. Для этого имя атрибута должно начинаться с двойного подчеркивания - `__name`. Например, перепишем предыдущую программу, сделав оба атрибута - name и age приватными:

```
class Person:
    def __init__(self, name, age):
        self.__name = name    # устанавливаем имя
        self.__age = age       # устанавливаем возраст

    def print_person(self):
        print(f"Имя: {self.__name}\tВозраст: {self.__age}")


tom = Person("Tom", 39)
tom.__name = "Человек-паук"     # пытаемся изменить атрибут __name
tom.__age = -129                # пытаемся изменить атрибут __
tom.print_person()              # Имя: Tom        Возраст: 39
```