Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Филиал «Минский радиотехнический колледж»

Учебная дисциплина

«Инструментальное программное обеспечение»

**Инструкция**

по выполнению лабораторной работы

«Разработка и проектирование классов.

Разработка, отладка и испытание программ создания классов

и экземпляров классов»

Минск 2020 г.

# **Лабораторная работа №11 (2 часа)**

**Тема работы: «Разработка и проектирование классов»**

**1 Цель работы**

Научиться проектировать и создавать классы для решения задач.

**2 Задание**

Спроектировать и реализовать класс, который будет содержать не менее 3 полей, одного конструктора и трех методов. Предметную область для создания класса определяете сами.

**3 Оснащение работы**

Задание по варианту, ЭВМ, среда разработки **Python 3.7, IDLE**.

**4 Основные теоретические сведения**

**Объектно-ориентированное программирование**

**Классы и объекты**

Python поддерживает объектно-ориентированную парадигму программирования, а это значит, что мы можем определить компоненты программы в виде классов.

Класс является шаблоном или формальным описанием объекта, а объект представляет экземпляр этого класса, его реальное воплощение. Можно провести следующую аналогию: у всех у нас есть некоторое представление о человеке - наличие двух рук, двух ног, головы, пищеварительной, нервной системы, головного мозга и т.д. Есть некоторый шаблон – этот шаблон можно назвать классом. Реально же существующий человек (фактически экземпляр данного класса) является объектом этого класса.

С точки зрения кода класс объединяет набор функций и переменных, которые выполняют определенную задачу. Функции класса еще называют методами. Они определяют поведение класса. А переменные класса называют атрибутами – они хранят состояние класса/

Класс определяется с помощью ключевого слова class:

|  |  |
| --- | --- |
| 1  2 | class название\_класса:      методы\_класса |

Для создания объекта класса используется следующий синтаксис:

|  |  |
| --- | --- |
| 1 | название\_объекта = название\_класса([параметры]) |

Например, определим простейший класс Person, который будет представлять человека:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class Person:      name = "Tom"        def display\_info(self):          print("Привет, меня зовут", self.name)    person1 = Person()  person1.display\_info()         # Привет, меня зовут Tom    person2 = Person()  person2.name = "Sam"  person2.display\_info()         # Привет, меня зовут Sam |

Класс Person определяет атрибут name, который хранит имя человека, и метод display\_info, с помощью которого выводится информация о человеке.

При определении методов любого класса следует учитывать, что все они должны принимать в качестве первого параметра ссылку на текущий объект, который. согласно условностям, называется self (в ряде языков программирования есть своего рода аналог – ключевое слово this). Через эту ссылку внутри класса мы можем обратиться к методам или атрибутам этого же класса. В частности, через выражение self.name можно получить имя пользователя.

После определения класс Person создаем пару его объектов – person1 и person2. Используя имя объекта, мы можем обратиться к его методам и атрибутам. В данном случае у каждого из объектов вызываем метод display\_info(), который выводит строку на консоль, и у второго объекта также изменяем атрибут name. При этом при вызове метода display\_info не надо передавать значение для параметра self.

**Конструкторы**

Для создания объекта класса используется конструктор. Так, выше, когда мы создавали объекты класса Person, мы использовали конструктор по умолчанию, который неявно имеют все классы:

|  |  |
| --- | --- |
| 1  2 | person1 = Person()  person2 = Person() |

Однако мы можем явным образом определить в классах конструктор с помощью специального метода, который называется \_\_init(). К примеру, изменим класс Person, добавив в него конструктор:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class Person:        # конструктор      def \_\_init\_\_(self, name):          self.name = name  # устанавливаем имя        def display\_info(self):          print("Привет, меня зовут", self.name)  person1 = Person("Tom")  person1.display\_info()         # Привет, меня зовут Tom  person2 = Person("Sam")  person2.display\_info()         # Привет, меня зовут Sam |

В качестве первого параметра конструктор также принимает ссылку на текущий объект – self. Нередко в конструкторах устанавливаются атрибуты класса. Так, в данном случае в качестве второго параметра в конструктор передается имя пользователя, которое устанавливается для атрибута self.name. Причем для атрибута необязательно определять в классе переменную name, как это было в предыдущей версии класса Person. Установка значения self.name = name уже неявно создает атрибут name.

|  |  |
| --- | --- |
| 1  2 | person1 = Person("Tom")  person2 = Person("Sam") |

В итоге мы получим следующий консольный вывод:

Привет, меня зовут Tom

Привет, меня зовут Sam

**Деструктор**

После окончания работы с объектом мы можем использовать оператор del для удаления его из памяти:

|  |  |
| --- | --- |
| 1  2  3 | person1 = Person("Tom")  del person1     # удаление из памяти  # person1.display\_info()  # Этот метод работать не будет, так как person1 уже удален из памяти |

Стоит отметить, что в принципе это необязательно делать, так как после окончания работы скрипта все объекты автоматически удаляются из памяти.

Кроме того, мы можем определить в классе деструктор, реализовав встроенную функцию \_\_del\_\_, который будет вызываться либо в результате вызова оператора del, либо при автоматическом удалении объекта. Например:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  1 | class Person:      # конструктор      def \_\_init\_\_(self, name):          self.name = name  # устанавливаем имя        def \_\_del\_\_(self):          print(self.name,"удален из памяти")      def display\_info(self):          print("Привет, меня зовут", self.name)      person1 = Person("Tom")  person1.display\_info()  # Привет, меня зовут Tom  del person1     # удаление из памяти  person2 = Person("Sam")  person2.display\_info()  # Привет, меня зовут Sam |

Консольный вывод:

Привет, меня зовут Tom

Tom удален из памяти

Привет, меня зовут Sam

Sam удален из памяти

Определение классов в модулях и подключение

Как правило, классы размещаются в отдельных модулях и затем уже импортируются в основной скрипт программы. Пусть у нас будет в проекте два файла: файл main.py (основной скрипт программы) и classes.py (скрипт с определением классов).

В файле classes.py определим два класса:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class Person:        # конструктор      def \_\_init\_\_(self, name):          self.name = name  # устанавливаем имя        def display\_info(self):          print("Привет, меня зовут", self.name)      class Auto:      def \_\_init\_\_(self, name):          self.name = name        def move(self, speed):          print(self.name, "едет со скоростью", speed, "км/ч") |

В дополнение к классу Person здесь также определен класс Auto, который представляет машину и который имеет метод move и атрибут name. Подключим эти классы и используем их в скрипте main.py:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | from classes import Person, Auto    tom = Person("Tom")  tom.display\_info()    bmw = Auto("BMW")  bmw.move(65) |

Подключение классов происходит точно также, как и функций из модуля. Мы можем подключить весь модуль выражением:

|  |  |
| --- | --- |
| 1 | import classes |

Либо подключить отдельные классы, как в примере выше.

В итоге мы получим следующий консольный вывод:

Привет, меня зовут Tom

BMW едет со скоростью 65 км/ч

**5 Порядок выполнения работы**

1. Выделить ключевые моменты задачи.

2. Построить алгоритм решения задачи.

3. Запрограммировать полученный алгоритм.

4. Провести тестирование полученной программы.

**6 Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7 Контрольные вопросы и задания**

1. Перечислите составные части класса?
2. Как создать объект класса?

**8 Рекомендуемая литература**

**Прохоренок, Н.А.** Python 3. Самое необходимое / Н.А Прохоренок, В.А. Дронов – СПб.: БВХ-Петербург, 2016.

**Федоров, Д.Ю.** Основы программирования на примере языка Python / Д.Ю. Федоров. – М.: Юрайт, 2018.