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, отладка и испытание программ с применением методов, статических методов и абстрактных методов»

Минск 2020 г.

# **Лабораторная работа №13**

**Тема работы: «Разработка, отладка и испытание программ с применением методов, статических методов и абстрактных методов»**

**1 Цель работы**

Научить использовать статические и абстрактные методы.

**2 Задание**

В класс лабораторной работы № 11 необходимо добавить по два статических и абстрактных метода, реализовать перегрузку 2 методов.

**3 Оснащение работы**

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

**4 Основные теоретические сведения**

**Методы**

Как мы выяснили ранее, в объектно-ориентированном программировании, методы используются для реализации функционалов объекта. В предыдущем разделе мы создали методы start() и stop() для класса Car. До этих пор, мы использовали объекты класса для вызова методов. Однако, есть тип методов, который может быть вызван напрямую при помощи имени класса. Такой метод называется **статичным методом**.

Статичные методы

Для объявления статического метода, вам нужно указать дескриптор @staticmethod перед названием метода, как показано ниже:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class Car:        @staticmethod      def get\_class\_details():          print ("Это класс Car")    Car.get\_class\_details() |

В коде выше мы создали класс Car с одним статичным методом get\_class\_details(). Давайте вызовем этот метод, используя название класса.

|  |  |
| --- | --- |
| 1 | Car.get\_class\_details() |

Вы можете видеть что нам не нужно создавать экземпляр класса Car для вызова метода get\_class\_details(), вместо этого мы просто использовали название класса. Стоит упомянуть, что статические методы могут иметь доступ только к атрибутам класса в Python, вы не сможете обратиться к методам через self.

**Возврат множественных значений из метода**

Одна из лучших особенностей языка Python заключается в том, что **методы класса** могут возвращать множественные значения. Взгляните на следующий пример:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class Square:        @staticmethod      def get\_squares(a, b):          return a\*a, b\*b    print(Square.get\_squares(3, 5)) |

В скрипте выше мы создали класс под названием Square со статичным методом get\_squares(). Метод принимает два параметра. Он умножает каждый параметр на себя и возвращает оба результата при помощи оператора return. В выдаче указанного выше скрипта вы увидите квадраты 3 и 5.

**Метод str**

До этого момента мы выводили атрибуты при помощи метода print(). Посмотрим, что случится, если мы выведем объект класса.

Для этого нам нужно создать простой класс Car с одним методом и попытаться вывести объект класса в консоль. Выполним следующий скрипт:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class Car:        # создание методов класса      def start(self):          print ("Двигатель заведен")    car\_a = Car()  print(car\_a) |

В скрипте выше мы создали объект car\_a класса Car и вывели его значение на экран. По сути мы относимся к объекту car\_a как к строке. Выдача выглядит следующим образом:

|  |  |
| --- | --- |
| 1 | <\_\_main\_\_.Car object at 0x000001CCCF4335C0> |

Выдача показывает локацию памяти, где хранится наш объект. Каждый **объект Python** по умолчанию содержит метод \_\_str\_\_. Когда вы используете объект в качестве строки, вызывается метод \_\_str\_\_, который по умолчанию выводит локацию памяти объекта. Однако, вы также можете предоставить собственное **определение метода** \_\_str\_\_. Например, как в следующем примере:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # создание класса Car  class Car:        # создание методов класса      def \_\_str\_\_(self):          return "Car class Object"        def start(self):          print ("Двигатель заведен")    car\_a = Car()  print(car\_a) |

В скрипте выше, мы переопределили метод \_\_str\_\_, предоставив наше собственное определение метода. Теперь, если вы выведите объект car\_a, вы увидите сообщение «**Car class Object**» в консоли. Это сообщение, которое мы внесли в наш пользовательский метод \_\_str\_\_.

Использование этого метода позволяет вам создавать пользовательские и более осмысленные описания, когда объект выводится. Вы можете даже отобразить кое-какие данные внутри класса, такие как название класса Car.

**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. Для чего используются методы?
2. В чем особенность статических и абстрактных методов?

**8 Рекомендуемая литература**

**Плас, Дж. В.** Python для сложных задач. Наука о данных и машинное обучение / Дж.В. Плас. – СПб: Питер, 2018.

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