ЛАБОРАТОРНАЯ РАБОТА №11

По дисциплине: Язык программирования Python
Тема занятия: Пользовательский класс
Цель занятия: Научиться создавать и описывать класс
Количество часов: 6

Содержание работы:

Задание: Создать свой класс, который будет наследовать 2 свойства и 4 метода другого класса (родителя) и будет содержать по одному своему свойству и методу. Тематика выбирается студентом самостоятельно.

Методические указания по выполнению:

В языке Python для определения класса используется оператор class:

```
class имя_класса(класс1, класс2, ...):
    # определения методов
```

В языке Python не считается зазорным получить доступ к некоторому атрибуту (не методу) напрямую, если, конечно, этот атрибут описан в документации как часть интерфейса класса. Такие атрибуты называются свойствами (properties). В других языках программирования принято для доступа к свойствам создавать специальные методы (вместо того чтобы напрямую обращаться к общедоступным членам-данным). В Python достаточно использовать ссылку на атрибут, если свойство ни на что в объекте не влияет (то есть другие объекты могут его произвольно менять). Если же свойство менее тривиально и требует каких-то действий в самом объекте, его можно описать как свойство (пример взят из документации к Python):

```
class C(object):
  def getx(self): return self.__x
  def setx(self, value): self.__x = value
  def delx(self): del self.__x
  x = property(getx, setx, delx, "I'm the 'x' property.")
```

Синтаксически доступ к свойству x будет обычной ссылкой на атрибут:

```
>>> c = C()
>>> c.x = 1
>>> print c.x
1
>>> del c.x
```

А на самом деле будут вызываться соответствующие методы: setx(), getx(), delx().

Например, классы определяют поведение своих экземпляров с помощью функций, создаваемых инструкциями def внутри инструкции class. Поскольку такие вложенные инструкции def выполняют присваивание именам внутри класса, они присоединяются к объектам классов в виде атрибутов и будут унаследованы всеми экземплярами и подклассами:

```
class C1(C2, C3): # Создать и связать класс C1
  def setname(self, who): # Присвоить: Cl.setname
      self.name = who # Self - либо 11, либо 12
```

```
11 = C1() # Создать два экземпляра
12 = C1()
I1.setname( "bob") # Записать 'bob' в 11. name
I2.setname('mel') # Записать 'meГ в 12. name
print 11. name # Выведет 'bob'
```

Если в классе потребуется гарантировать, что атрибут, такой как name, всегда будет присутствовать в экземплярах, то такой атрибут должен создаваться на этапе создания класса, как показано ниже:

```
class C1(C2, C3):
        def init (self, who): # Создать имя при создании класса
                self, name = who # Self - либо II, либо 12
11 = CI ('bob") # Записать 'bob' в 11. паше
12 = C1('теГ) # Записать 'теГ в 12. пате
print П. name # Выведет 'bob'
```

В этом случае интерпретатор Python автоматически будет вызывать метод с именем init каждый раз при создании экземпляра класса. Новый экземпляр будет передаваться методу init в виде первого аргумента self, а любые значения, перечисленные в круглых скобках при вызове класса, будут передаваться во втором и последующих за ним аргументах. В результате инициализация экземпляров будет выполняться в момент их создания, без необходимости вызывать дополнительные методы.Метод init известен как конструктор, так как он запускается на этапе конструирования экземпляра. Этот метод является типичным представителем большого класса методов, которые называются методами перегрузки операторов.

Вопросы для защиты лабораторной работы:

1. Вопросы по листингу программы