Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины
«Объектно-ориентированное программирование»
Вариант 4

Выполнил:
Кубанова Ксения Олеговна
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Ассистент департамента цифровых, робототехнических систем и электроники Богданов С. С.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: элементы объектно-ориентированного программирования в языке Python

Цель: приобретение навыков по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

Пример.

Рациональная (несократимая) дробь представляется парой целых чисел (a, b), где a — числитель, b — знаменатель. Создать класс Rational для работы с рациональными дробями. Обязательно должны быть реализованы операции:

- сложения add, $(a, b) + (c, d) = (ad + be, bd)$;
- вычитания sub, $(a, b) - (c, d) = (ad - be, bd)$;
- умножения mul, $(a, b) \times (c, d) = (ac, bd)$;
- деления div, $(a, b)/(c, d) = (ad, be)$;
- сравнения equal, greate, less.

Рисунок 1. Необходимые операции для реализации примера

Должна быть реализована приватная функция сокращения дроби reduce, которая обязательно вызывается при выполнении арифметических операций.

Далее было выполнено задание и отражено в файле *prim1.py* .

```
# Умножение обыкновенных дробей.
 def mul(self, rhs):
    if isinstance(rhs, Rational):
       a = self.numerator * rhs.numerator
       b = self.denominator * rhs.denominator
       return Rational(a, b)
    else:
       raise ValueError()
# Деление обыкновенных дробей.
 def div(self, rhs):
    if isinstance(rhs, Rational):
       a = self.numerator * rhs.denominator
       b = self.denominator * rhs.numerator
       return Rational(a, b)
    else:
       raise ValueError()
# Отношение обыкновенных дробей.
 def equals(self, rhs):
    if isinstance(rhs, Rational):
        return (self.numerator == rhs.numerator) and \
           | | (self.denominator == rhs.denominator)
    else:
       return false
```

Рисунок 2. Одни из реализованных операций примера

ИДЗ 1.

Парой называется класс с двумя полями, которые обычно имеют имена first и second. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать:

- метод инициализации __init__;

- метод должен контролировать значения аргументов на корректность;

- ввод с клавиатуры read ;

- вывод на экран display .

Реализовать внешнюю функцию с именем make_тип() , где тип — тип реализуемой структуры.

Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

Вариант 4.

Поле first — целое положительное число, номинал купюры; номинал может принимать значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000 Поле second — целое положительное число, количество купюр данного достоинства. Реализовать метод summa() — вычисление денежной суммы.

```
Номинал купюры: 5000, Количество купюр: 67
Общая сумма: 335000
Введите номинал купюры (1, 2, 5, 10, 50, 100, 500, 1000, 5000): 50
Введите количество купюр: 78
Номинал купюры: 50, Количество купюр: 78
Общая сумма: 3900
```

Рисунок 3. Вывод программы

```
def read(self):
    self.first = int(input("Введите номинал купюры (1, 2, 5, 10, 50, 100, 500, 1000, 5000): "))
    self.second = int(input("Введите количество купюр: "))
    # Проверка корректности введенных значений
    if self.first not in [1, 2, 5, 10, 50, 100, 500, 1000, 5000] or self.first <= 0:
        raise ValueError("Некорректный номинал купюры.")
    if self.second <= 0:
        raise ValueError("Количество купюр должно быть положительным числом.")

def display(self):
    print(f"Номинал купюры: {self.first}, Количество купюр: {self.second}")

def summa(self):
    return self.first * self.second
```

Рисунок 4. Реализация требований

Готовый код представлен в файле *ind1.py*.

ИДЗ 2.

Составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

метод инициализации __init__ ;

ввод с клавиатуры read ;

вывод на экран display .

В раздел программы, начинающийсяпосле инструкции if __name__ = '__main__': добавить код, демонстрирующий возможности разработанного класса.

Вариант 4.

Создать класс Triangle для представления треугольника. Поля данных должны включать углы и стороны. Требуется реализовать операции: получения и изменения полей данных, вычисления площади, вычисления периметра, вычисления высот, а также определения вида треугольника (равносторонний, равнобедренный или прямоугольный).

Выполнение.

Индивидуальное задание 2 было реализовано в соответствие с заданным условием.

```
def read(self):
    self.a = int(input("Сторона AB треугильника ABC = "))
    self.b = int(input("Сторона AC треугильника ABC = "))
    self.c = int(input("Сторона BC треугильника ABC = "))
    self.angles = self.calculate_angles()

    if self.a < 1 and self.b < 1 and self.c < 1:
        raise ValueError("Некорректные введённые данные, стороны и углы должны быть больше 0")
```

Рисунок 5. Фрагмент кода read()

```
def display(self):
    print(f"Углы: {self.angles}")
    print(f"Стороны AB = {self.a} AC = {self.b} BC = {self.c}")
    print(f"S = {self.FindS()}")
    print(f"P = {self.FindP()}")
    print(f"Type = {self.type()}")
    print(f"Высота относительно стороны a: {self.height(self.a)}")
    print(f"Высота относительно стороны b: {self.height(self.b)}")
    print(f"Высота относительно стороны c: {self.height(self.c)}")
```

Рисунок 6. Фрагмент кода display()

```
if __name__ == '__main__':
    Tria = Triangle()
    Tria.read()
    Tria.display()
```

Рисунок 7. Фрагмент кода, демонстрирующий работу созданного класса

По итогу выполнения в выводе работы:

```
Сторона AB треугильника ABC = 20
Сторона AC треугильника ABC = 20
Сторона BC треугильника ABC = 20
Углы: (60.00000000000001, 60.00000000000001, 59.99999999999999)
Стороны AB = 20 AC = 20 BC = 20
S = 173.20508075688772
P = 60
Type = равносторонний
Высота относительно стороны a: 17.32050807568877
Высота относительно стороны b: 17.32050807568877
Высота относительно стороны c: 17.32050807568877
```

Рисунок 8. Вывод программы

Ответы на контрольные вопросы

1 Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова class и его имени.

2 Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов.

Что касается переменных экземпляра, они хранят данные, уникальные для каждого объекта класса.

3 Каково назначение методов класса?

Методы определяют функциональность объектов, принадлежащих конкретному классу.

4 Для чего предназначен метод __init__() класса?

Метод __init__ указывает, какие атрибуты будут у экземпляров нашего класса.

5 Каково назначение self ?

Аргумент self представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам.

6 Как добавить атрибуты в класс?

Для добавления атрибутов в класс необходимо либо записать информацию в начале сразу после создания класса, либо создать переменную следующим образом: «Pet.all_specs = [tom.spec, avocado.spec, ben.spec]».

7 Как осуществляется управление доступом к методам и атрибутам в языке Python?

Публичные атрибуты и методы: По умолчанию все атрибуты и методы являются публичными и доступны из любого места в коде.

Защищенные атрибуты и методы: Если имя атрибута или метода начинается с одного подчеркивания (например, _protected), это является сигналом для разработчиков, что данный элемент предназначен для внутреннего использования и не должен использоваться за пределами класса или модуля.

Приватные атрибуты и методы: Если имя начинается с двух подчеркиваний (например, __private), Python применяет механизм "имя манглирования", что делает атрибут менее доступным из внешнего кода. Однако это не является строгим ограничением, и доступ к таким атрибутам все еще возможен, но требует использования специального синтаксиса.

8 Каково назначение функции isinstance?

Функция isinstance() в Python используется для проверки, является ли объект экземпляром определенного класса или его подкласса. Она принимает два аргумента: объект и класс (или кортеж классов) и возвращает True, если объект является экземпляром указанного класса, и False в противном случае.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.