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 №3
дисциплины «Объектно-ориентированное программирование»

Выполнил:
Кожуховский Виктор Андреевич
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:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Наследование и полиморфизм в языке Python

Цель: приобретение навыков по созданию иерархии классов при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Изучил теоретический материал работы.

2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал примеры лабораторной работы.

9. Разработайте программу по следующему описанию.

В некой игре-стратегии есть солдаты и герои. У всех есть свойство, содержащее уникальный номер объекта, и свойство, в котором хранится принадлежность команде. У солдат есть метод "иду за героем", который в качестве аргумента принимает объект типа "герой". У героев есть метод увеличения собственного уровня. В основной ветке программы создается по одному герою для каждой команды. В цикле генерируются объекты-солдаты. Их принадлежность команде определяется случайно. Солдаты разных команд добавляются в разные списки. Измеряется длина списков солдат противоборствующих команд и выводится на экран. У героя, принадлежащего команде с более длинным списком, увеличивается уровень. Отправьте одного из солдат первого героя следовать за ним. Выведите на экран идентификационные номера этих двух юнитов.

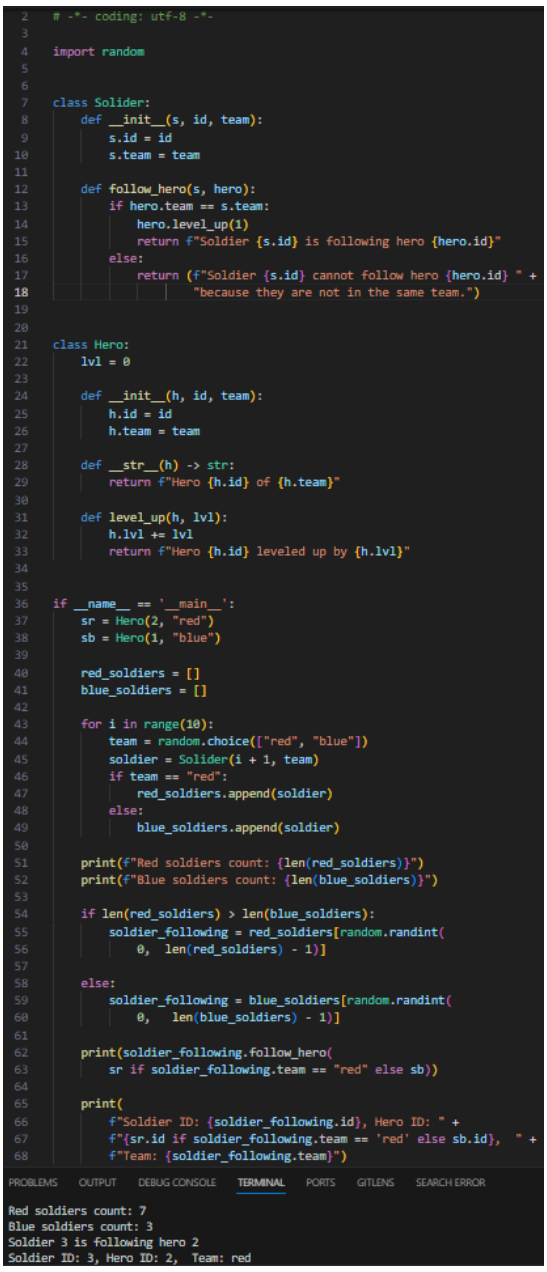


Рисунок 1. Выполнение общего задания

8. Выполнил индивидуальное задание для варианта 14.

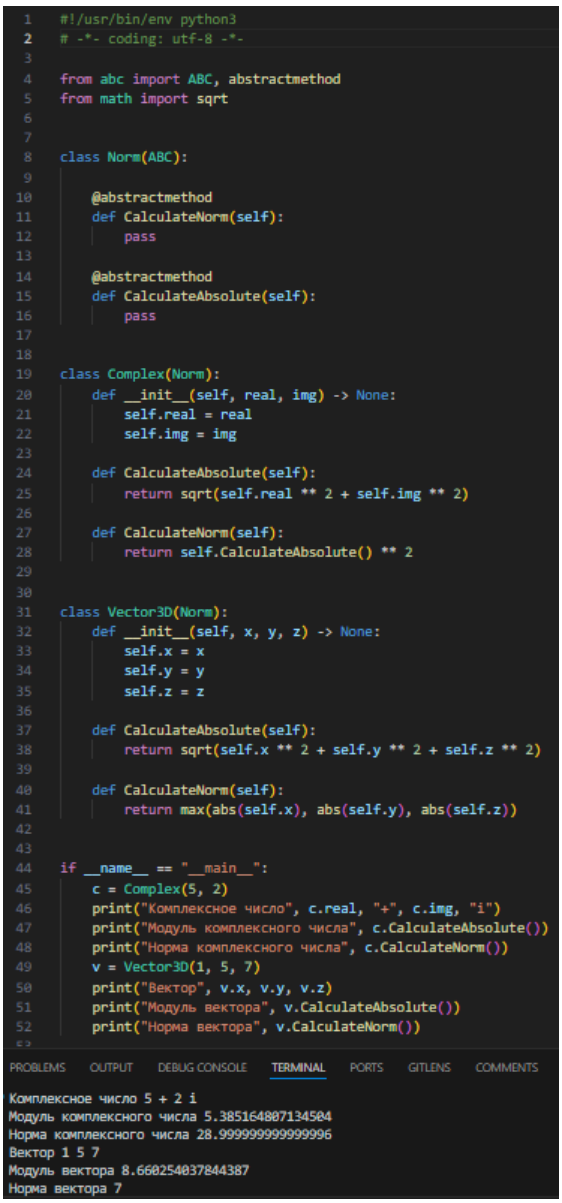
Задание 1

Реализовать класс-оболочку Number для числового типа float. Реализовать методы умножения и вычитания. Создать производный класс Real, в котором реализовать метод, вычисляющий корень произвольной степени, и метод для вычисления числа в данной степени.

Рисунок 2. Код решения индивидуального задания 1 и его выполнение

Задание 2

Создать абстрактный класс Norm с виртуальной функцией вычисления нормы и модуля. Определить производные классы Complex, Vector3D с собственными функциями вычисления нормы и модуля. (Модуль для комплексного числа вычисляется как корень из суммы квадратов действительной и мнимой частей; норма для комплексных чисел вычисляется как модуль в квадрате. Модуль вектора вычисляется как корень квадратный из суммы квадратов координат; норма вектора вычисляется как максимальное из абсолютных значений координат.)



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from abc import ABC, abstractmethod
from math import sqrt


class Norm(ABC):

    @abstractmethod
    def CalculateNorm(self):
        pass

    @abstractmethod
    def CalculateAbsolute(self):
        pass


class Complex(Norm):
    def __init__(self, real, img) -> None:
        self.real = real
        self.img = img

    def CalculateAbsolute(self):
        return sqrt(self.real ** 2 + self.img ** 2)

    def CalculateNorm(self):
        return self.CalculateAbsolute() ** 2


class Vector3D(Norm):
    def __init__(self, x, y, z) -> None:
        self.x = x
        self.y = y
        self.z = z

    def CalculateAbsolute(self):
        return sqrt(self.x ** 2 + self.y ** 2 + self.z ** 2)

    def CalculateNorm(self):
        return max(abs(self.x), abs(self.y), abs(self.z))


if __name__ == "__main__":
    c = Complex(5, 2)
    print("Комплексное число", c.real, "+", c.img, "i")
    print("Модуль комплексного числа", c.CalculateAbsolute())
    print("Норма комплексного числа", c.CalculateNorm())
    v = Vector3D(1, 5, 7)
    print("Вектор", v.x, v.y, v.z)
    print("Модуль вектора", v.CalculateAbsolute())
    print("Норма вектора", v.CalculateNorm())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

```
Комплексное число 5 + 2 i
Модуль комплексного числа 5.385164807134504
Норма комплексного числа 28.999999999999996
Вектор 1 5 7
Модуль вектора 8.660254037844387
Норма вектора 7
```

Рисунок 3. Код решения индивидуального задания 2 и его выполнение

9. Зафиксировал сделанные изменения в репозитории.

10. Выполнил слияние ветки для разработки с веткой master/main.

11. Отправил сделанные изменения на сервер GitHub.

Ссылка: https://github.com/Viktorkozh/OOP-3

Контрольные вопросы:

1. Как осуществляется объявление класса в языке Python?

class MyClass:

pass

Ключевое слово class используется для объявления класса, за которым следует имя класса и двоеточие.

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса: Общие для всех экземпляров класса. Объявляются внутри класса, но вне методов.

Атрибуты экземпляра: Уникальны для каждого экземпляра класса. Объявляются внутри метода __init__.

3. Каково назначение методов класса?

Методы класса определяют поведение объектов класса. Они могут изменять состояние объекта или выполнять действия, связанные с объектом.

4. Для чего предназначен метод __init__() класса?

Это инициализатор, который вызывается при создании нового экземпляра класса. Он используется для инициализации атрибутов экземпляра.

```
class MyClass:
    def __init__(self, value):
        self.value = value
```

5. Каково назначение self ?

self представляет собой ссылку на текущий экземпляр класса и используется для доступа к атрибутам и методам класса изнутри.

6. Как добавить атрибуты в класс?

Атрибуты можно добавлять в методе __init__ или в любом другом методе класса.

```
class MyClass:
    def __init__(self, value):
        self.value = value
```

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

В Python используется соглашение об именовании для управления доступом:

Приватные атрибуты и методы начинаются с одного или двух подчеркиваний (например, _private или __very_private).

Публичные атрибуты и методы не имеют подчеркиваний.

8. Каково назначение функции isinstance ?

Функция isinstance используется для проверки, является ли объект экземпляром определенного класса или кортежа классов.

isinstance(obj, MyClass)

Вывод: приобрел навыки по созданию иерархии классов при написании программ с помощью языка программирования Python версии 3.x.