

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Лабораторная работа 3

Наследование и полиморфизм в языке Python

Выполнил студент группы ИВТ-б-о-20-1

Пентухов С. А. « » _____ 2022г.

Подпись студента _____

Работа защищена « » _____ 2022г.

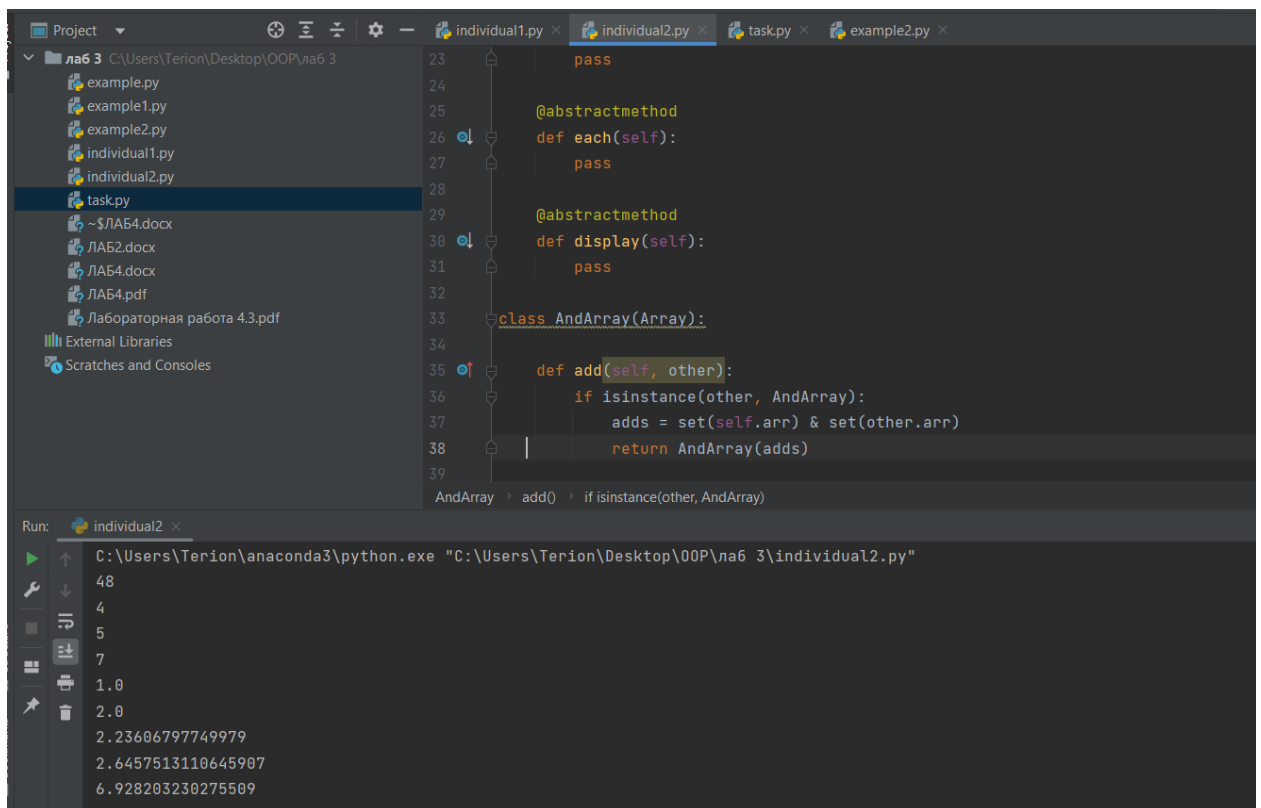
Проверил Воронкин Р.А. _____

(подпись)

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

Индивидуальное задание 1

Создать класс Pair (пара чисел); определить метод перемножения полей и операцию сложения пар $(a,b) + (c,d) = (a + b, c + d)$. Определить производный класс Complex с полями: действительная и мнимая части числа. Определить методы умножения $(a,b) * (c,d) = (ac - bd, ad + bc)$ и вычитания $(a,b) - (c,d) = (a - b, c - d)$. Результаты выполнения отображены на рисунке 2.



The screenshot shows an IDE with a project named 'лаб 3' containing files 'example.py', 'example1.py', 'example2.py', 'individual1.py', 'individual2.py', 'task.py', and 'example2.py'. The 'individual2.py' file is open, showing the following code:

```
23 pass
24
25 @abstractmethod
26 def each(self):
27     pass
28
29 @abstractmethod
30 def display(self):
31     pass
32
33 class AndArray(Array):
34
35     def add(self, other):
36         if isinstance(other, AndArray):
37             adds = set(self.arr) & set(other.arr)
38             return AndArray(adds)
39
```

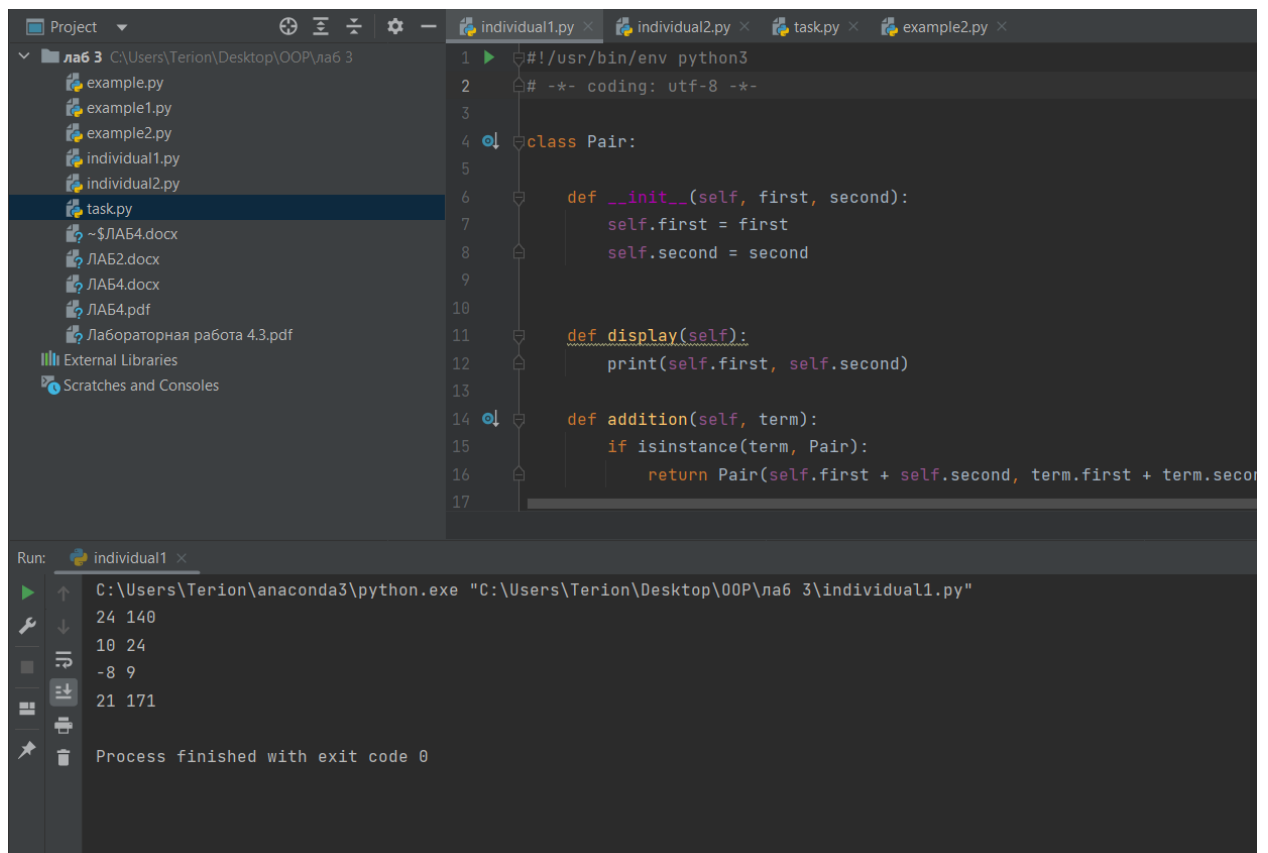
The Run console shows the execution of 'individual2.py' with the following output:

```
Run: individual2
C:\Users\Terion\anaconda3\python.exe "C:\Users\Terion\Desktop\OOP\лаб 3\individual2.py"
48
4
5
7
1.0
2.0
2.23606797749979
2.6457513110645907
6.928203230275509
```

Рисунок 1 – Результат выполнения

Индивидуальное задание 2

Создать абстрактный базовый класс `Array` с виртуальными методами сложения и поэлементной обработки массива `foreach()`. Разработать производные классы `AndArray` и `OrArray` (выбор). В первом классе операция сложения реализуется как пересечение множеств, а поэлементная обработка представляет собой извлечение квадратного корня. Во втором классе операция сложения реализуется как объединение, а поэлементная обработка — вычисление логарифма. Результат выполнения отображен на рисунке 3.



The screenshot shows an IDE with a project named 'лаб 3' (lab 3) containing several files. The active file is 'individual1.py', which defines a `Pair` class. The class has an `__init__` method to initialize `first` and `second` attributes, a `display` method to print these values, and an `addition` method that takes a `term` (another `Pair` object) and returns a new `Pair` object with the sum of corresponding attributes. The `Run` window at the bottom shows the execution of the script, displaying the output of the `display` method for three different `Pair` objects: (24, 140), (10, 24), and (-8, 9). The process finished with exit code 0.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 class Pair:
5
6     def __init__(self, first, second):
7         self.first = first
8         self.second = second
9
10
11     def display(self):
12         print(self.first, self.second)
13
14     def addition(self, term):
15         if isinstance(term, Pair):
16             return Pair(self.first + self.second, term.first + term.second)
17
```

Run: individual1

C:\Users\Terion\anaconda3\python.exe "C:\Users\Terion\Desktop\00P\лаб 3\individual1.py"

24 140
10 24
-8 9
21 171

Process finished with exit code 0

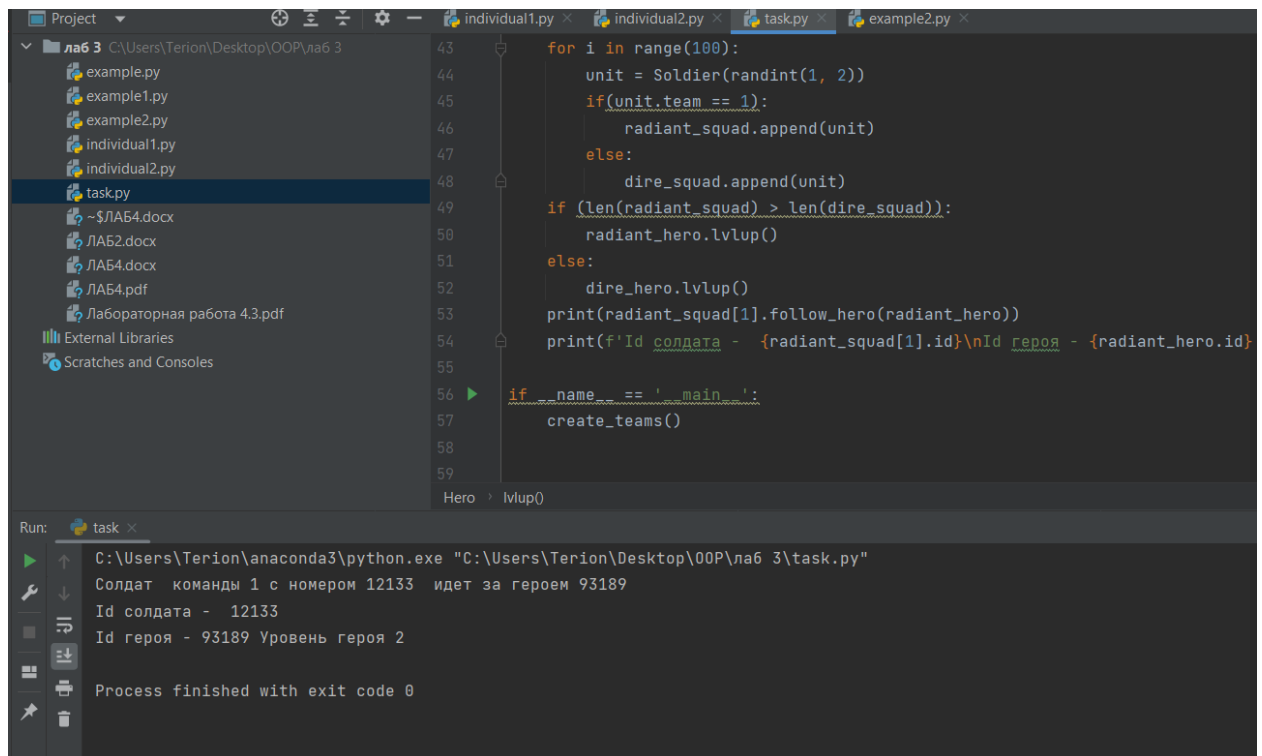
Рисунок 2 – Выполнения задания

Индивидуальное задание 3

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. Результат выполнения отображен на рисунке 1.



The screenshot shows an IDE with a project named 'лаб 3' (lab 3) on the left. The main editor displays the code in 'task.py'. The code generates two squads of soldiers, compares their lengths, levels up the hero of the larger squad, and prints the IDs of a soldier and a hero. The 'Run' console at the bottom shows the execution output.

```
43 for i in range(100):
44     unit = Soldier(randint(1, 2))
45     if unit.team == 1:
46         radiant_squad.append(unit)
47     else:
48         dire_squad.append(unit)
49 if (len(radiant_squad) > len(dire_squad)):
50     radiant_hero.lvup()
51 else:
52     dire_hero.lvup()
53 print(radiant_squad[1].follow_hero(radiant_hero))
54 print(f'Id солдата - {radiant_squad[1].id}\nId героя - {radiant_hero.id}')
55
56 if __name__ == '__main__':
57     create_teams()
58
59 Hero > lvup()
```

Run: task

```
C:\Users\Terion\anaconda3\python.exe "C:\Users\Terion\Desktop\00P\лаб 3\task.py"
Солдат команды 1 с номером 12133 идет за героем 93189
Id солдата - 12133
Id героя - 93189 Уровень героя 2
Process finished with exit code 0
```

Рисунок 3 – Выполнения задания

Ответы на контрольные вопросы

1. Как осуществляется объявление класса в языке Python? - Классы объявляются с помощью ключевого слова `class` и имени класса.

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. В этой теме мы рассмотрим только атрибуты класса, но не волнуйтесь, у вас будет достаточно времени, чтобы узнать больше и об атрибутах экземпляра.

3. Каково назначение методов класса? - Методы определяют функциональность объектов, принадлежащих конкретному классу.

4. Для чего предназначен метод `__init__()` класса? – Для объявления конструктора класса.

5. Каково назначение `self`? – Указание объекта на самого себя.

6. Как добавить атрибуты в класс? - Атрибуты экземпляра - это как раз те, которые мы определяем в методах, поэтому по определению мы можем создавать новые атрибуты внутри наших пользовательских методов.

Как осуществляется управление доступом к методам и атрибутам в языке Python? - т.д.) - В Python таких возможностей нет, и любой может обратиться к атрибутам и методам вашего класса, если возникнет такая необходимость. Это существенный недостаток этого языка, т.к. нарушается один из ключевых принципов ООП – инкапсуляция. Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются `getter/setter`, их можно реализовать, но ничего не помешает изменить атрибут напрямую. При этом есть соглашение, что метод или атрибут, который начинается с нижнего подчеркивания, является скрытым, и снаружи класса трогать его

не нужно (хотя сделать это можно).

7. В Python есть встроенная функция `instance ()`, которая сравнивает значение с указанным типом. Если данное значение и тип соответствуют, он вернет `true`, иначе `false`. Используя `isinstance ()`, вы можете проверить строку, число с плавающей точкой, `int`, список, кортеж, `dict`, `set`, `class` и т. Д.