

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций  
«Элементы объектно-ориентированного программирования в  
языке Python»**

**Отчет по лабораторной работе № 4.1  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Трушева В. О. \_\_\_\_ .« » 2023г.

Подпись студента \_\_\_\_\_

Работа защищена « \_\_\_\_\_ 20 \_\_ г.

Проверила Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2023

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

## Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

Repository name \*

pynikcmd

 / 

2.26\_LR\_OPI

2.26\_LR\_OPI is available.

Great repository names are short and memorable. Need inspiration? How about [fluffy-adventure?](#)

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)


This will set main as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

3. Выполните клонирование созданного репозитория.

```
D:\fgit>git clone https://github.com/pynikcmd/4.1_LR_OPI.git
Cloning into '4.1_LR_OPI'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 5.12 KiB | 873.00 KiB/s, done.
```

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

 .gitignore Update .gitignore

5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
D:\fgit\4.1_LR_OPI>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/fgit/4.1_LR_OPI/.git/hooks]
```

6. Создайте проект PyCharm в папке репозитория.

7. Проработайте примеры лабораторной работы.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  class Book:
6      material = "paper"
7      cover = "paperback"
8      all_books = []
9
10
11  if __name__ == '__main__':
12      print(Book.material)
13      print(Book.cover)
14      print(Book.all_books)
15
```

Run: Primer\_1 x

D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe

paper  
paperback  
[]

Пример 1

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  class River:
6      # список всех рек
7      all_rivers = []
8
9      def __init__(self, name, length):
10         self.name = name
11         self.length = length
12         # добавляем текущую реку в список всех рек
13         River.all_rivers.append(self)
14
15
16  if __name__ == '__main__':
17      volga = River("Волга", 3530)
18      seine = River("Сена", 776)
19      nile = River("Нил", 6852)
20      # далее печатаем все названия рек
21      for river in River.all_rivers:
22          print(river.name)
```

River

Run: Primer\_2 x

D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe

Волга  
Сена  
Нил

Пример 2

```
1  #!/usr/bin/env python3
2  #- coding: utf-8 -*-
3
4
5  class River:
6      all_rivers = []
7
8      def __init__(self, name, length):
9          self.name = name
10         self.length = length
11         River.all_rivers.append(self)
12
13     def get_info(self):
14         print("Длина {0} равна {1} км".format(self.name, self.length))
15
16
17 if __name__ == '__main__':
18     volga = River("Волга", 3530)
19     seine = River("Сена", 776)
20     nile = River("Нил", 6852)
21     volga.get_info()
22     seine.get_info()
23     nile.get_info()
```

Run: Primer\_3 x

D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe D:\fgit\4.1\_LR\_OPI\Tas

Длина Волга равна 3530 км

Длина Сена равна 776 км

Длина Нил равна 6852 км

Пример 3

```
16         print("Cannot load that much")
17
18     def unload_cargo(self, weight):
19         if self.cargo - weight >= 0:
20             self.cargo -= weight
21             print("Unloaded {} tons".format(weight))
22         else:
23             print("Cannot unload that much")
24
25     def name_captain(self, cap):
26         self.captain = cap
27         print("{} is the captain of the {}".format(self.captain, self.name))
28
29
30 if __name__ == '__main__':
31     black_pearl = Ship("Black Pearl", 800)
32     black_pearl.name_captain("Jack Sparrow")
33     print(black_pearl.captain)
34     black_pearl.load_cargo(600)
35     black_pearl.unload_cargo(400)
36     black_pearl.load_cargo(700)
37     black_pearl.unload_cargo(300)
38
```

Ship > load\_cargo() > if self.cargo + weight <= self....

Run: Primer\_4 ×

Loaded 600 tons  
Unloaded 400 tons  
Cannot load that much  
Cannot unload that much

Пример 4

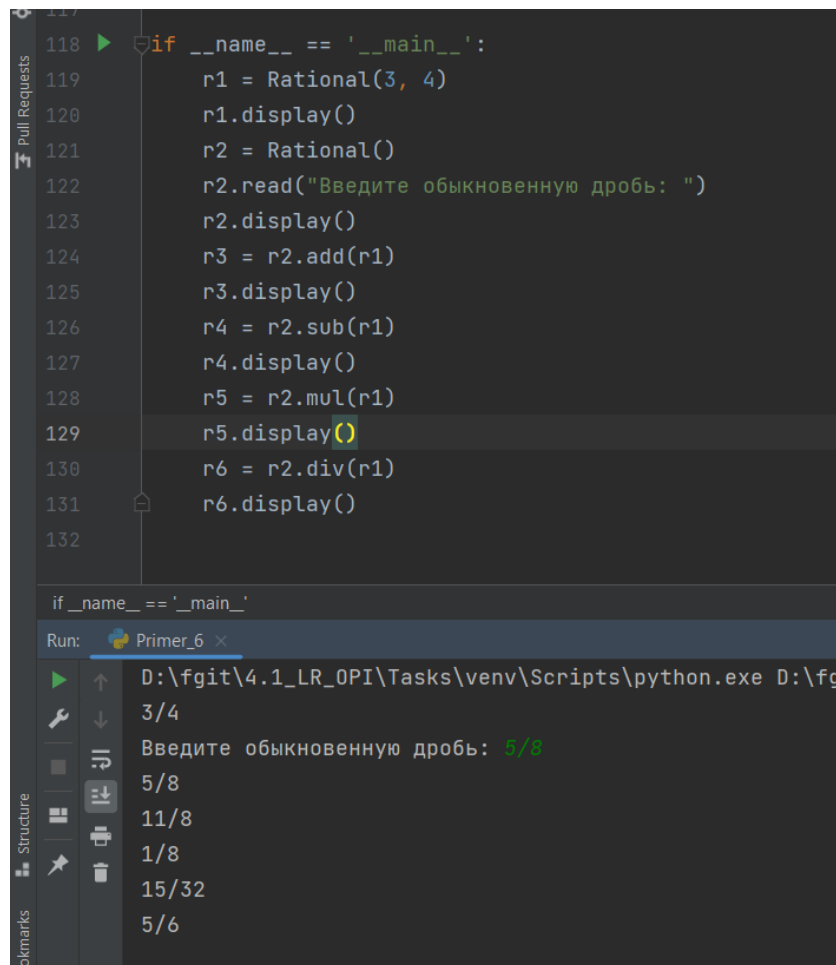
```
23         return self.__height
24
25     @height.setter
26     def height(self, h):
27         if h > 0:
28             self.__height = h
29         else:
30             raise ValueError
31
32     def area(self):
33         return self.__width * self.__height
34
35
36 if __name__ == '__main__':
37     rect = Rectangle(10, 20)
38     print(rect.width)
39     print(rect.height)
40     rect.width = 50
41     print(rect.width)
42     rect.height = 70
43     print(rect.height)
44
```

if \_\_name\_\_ == '\_\_main\_\_'

Run: Primer\_5 x

D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe D  
10  
20  
50  
70

Пример 5



```
117
118 ▶ if __name__ == '__main__':
119     r1 = Rational(3, 4)
120     r1.display()
121     r2 = Rational()
122     r2.read("Введите обыкновенную дробь: ")
123     r2.display()
124     r3 = r2.add(r1)
125     r3.display()
126     r4 = r2.sub(r1)
127     r4.display()
128     r5 = r2.mul(r1)
129     r5.display()
130     r6 = r2.div(r1)
131     r6.display()
132
```

Run: Primer\_6 ×

D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe

3/4

Введите обыкновенную дробь: 5/8

5/8

11/8

1/8

15/32

5/6

Пример 6

8. Выполните индивидуальные задания. Приведите в отчете скриншоты работы программ решения индивидуального задания.

#### Задание 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.

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанного класса.

### Вариант – 7 (27)

Условие. Поле `first` — дробное число, левая граница диапазона; поле `second` — дробное число, правая граница диапазона. Реализовать метод `rangecheck()` — проверку заданного числа на принадлежность диапазону.

```
25     def rangecheck(self, number):
26         # Проверка принадлежности числа к диапазону
27         if self.first <= number <= self.second:
28             print(f"{number} находится в диапазоне.")
29         else:
30             print(f"{number} вне диапазона.")
31
32
33     def make_pair(first, second):
34         # Создание объекта Pair и возврат его экземпляра
35         return Pair(first, second)
36
37
38 if __name__ == '__main__':
39     pair1 = Pair(1.5, 3.7)
40     pair1.display()
41
42     pair2 = make_pair(2.0, 4.5)
43     pair2.display()
44
45     number = float(input("Введите число для проверки диапазона: "))
46     pair2.rangecheck(number)
47
```

Pair > rangecheck() > else

Run: Ind\_1 x

D:\fgit\4.1\_LR\_OPI\Tasks\venv\Scripts\python.exe D:\fgit\4.1\_LR\_OPI\Ta  
Pair: 1.5, 3.7  
Pair: 2.0, 4.5  
Введите число для проверки диапазона: 5  
5.0 вне диапазона.

## Задание 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` .

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанного класса.

### Вариант – 12 (27)

Создать класс `Fraction` для работы с дробными числами. Число должно быть представлено двумя целочисленными полями: целая часть и дробная часть. Реализовать арифметические операции сложения, вычитания, умножения и операции сравнения.

```
6 class Fraction:
7     def __init__(self, whole_part, fractional_part):
8         self.whole_part = int(whole_part)
9         self.fractional_part = int(fractional_part)
10
11     def read(self):
12         self.whole_part = int(input("Введите целую часть: "))
13         self.fractional_part = int(input("Введите дробную часть: "))
14
15     def display(self):
16         print(f"Fraction: {self.whole_part}.{self.fractional_part}")
17
18     def add(self, other):
19         if isinstance(other, Fraction):
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Run: Ind\_2 x

```
D:\fgit\4.1_LR_OPI\Tasks\venv\Scripts\python.exe D:\fgit\4.1_LR_OPI\Ta
Fraction: 1.50
Введите целую часть: 10
Введите дробную часть: 2
Fraction: 10.2
Сложение:
Fraction: 11.52
Вычитание:
Fraction: -9.48
Умножение:
Fraction: 11.0
Fraction 1 меньше, чем Fraction 2
```

9. Зафиксируйте сделанные изменения в репозитории.

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

11. Отправьте сделанные изменения на сервер GitHub.

## Контрольные вопросы

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова `class` и имени класса:

```
class MyClass:
```

```
var = ... # некоторая переменная
```

```
def do_smt(self): # какой-то метод
```

## 2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибут класса - это атрибут, общий для всех экземпляров класса. 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. Атрибуты экземпляра определяются в методах и хранят информацию, специфичную для экземпляра.

## 3. Каково назначение методов класса?

Методы определяют функциональность объектов, принадлежащих конкретному классу.

## 4. Для чего предназначен метод `__init__()` класса?

Метод `__init__` является конструктором. Конструкторы - это концепция объектно-ориентированного программирования. Класс может иметь один и только один конструктор. Если `__init__` определен внутри класса, он автоматически вызывается при создании нового экземпляра 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. В примере с `__init__` мы создаем атрибуты для конкретного экземпляра и присваиваем им значения аргументов метода. Важно использовать параметр `self` внутри метода, если мы хотим сохранить значения экземпляра для последующего использования.

В большинстве случаев нам также необходимо использовать параметр `self` в других методах, потому что при вызове метода первым аргументом, который ему передается, является сам объект.

#### 6. Как добавить атрибуты в класс?

Новый атрибут класса указывается через точку после названия класса, затем ему присваивается определенное значение.

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

Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются `getter/setter`, их можно реализовать, но ничего не мешает 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).

Если же атрибут или метод начинается с двух подчеркиваний, то тут напрямую вы к нему уже не обратитесь (простым образом).

#### 8. Каково назначение функции `isinstance` ?

Встроенная функция `isinstance(obj, Cls)` , используемая при реализации методов арифметических операций и операций отношения, позволяет узнать что некоторый объект `obj` является либо экземпляром класса `Cls` либо экземпляром одного из потомков класса `Cls`.