

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ  
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет по лабораторной работе № 4.1  
«Элементы объектно-ориентированного  
программирования в языке Python»**

**по дисциплине «Основы программной инженерии»**

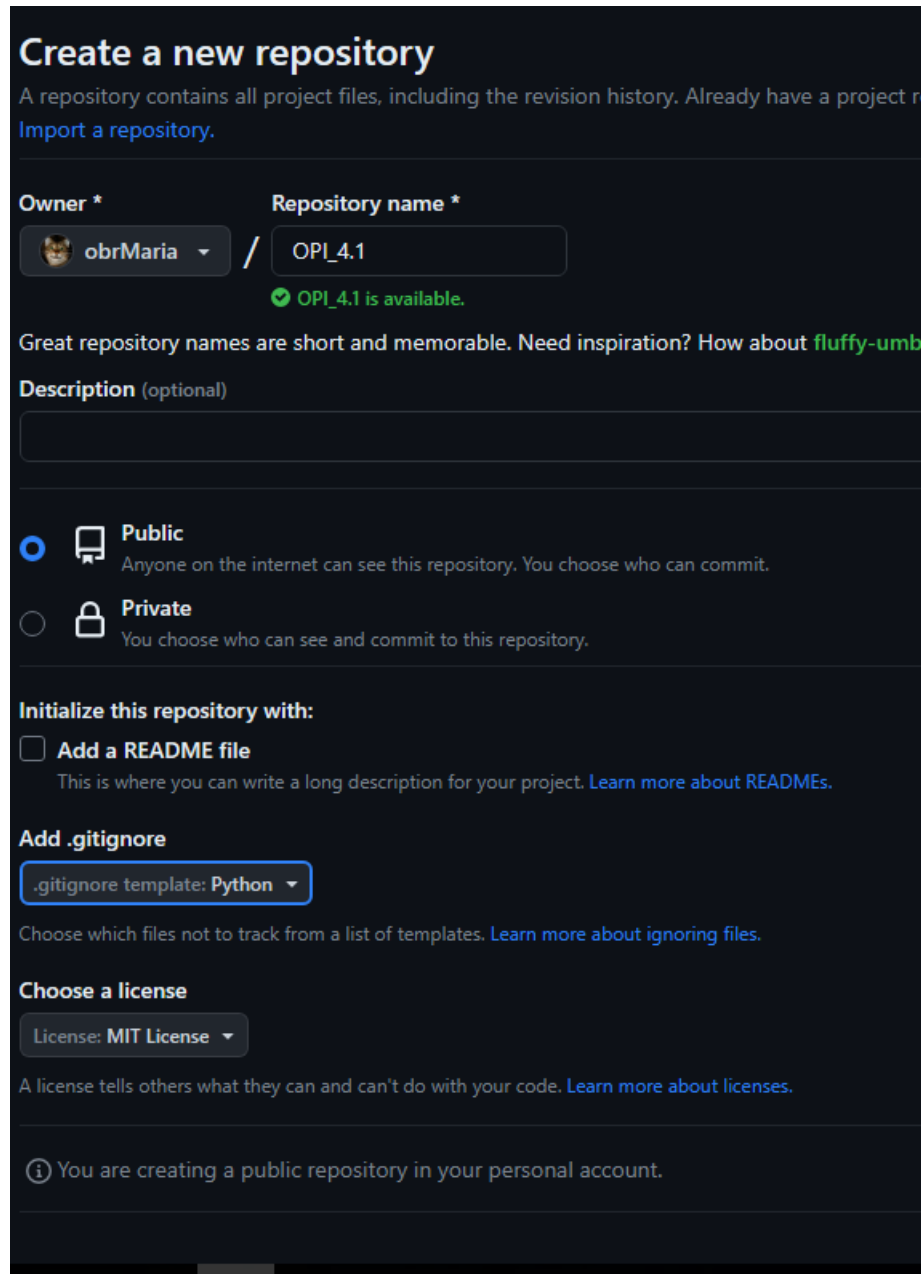
Выполнила:  
Образцова Мария Дмитриевна,  
2 курс, группа ПИЖ-б-о-21-1,  
Проверил:  
Доцент кафедры инфокоммуникаций,  
Воронкин Р.А.

Ставрополь, 2023 г.

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? [Import a repository.](#)

**Owner \*** obrMaria / **Repository name \*** OPI\_4.1

✔ OPI\_4.1 is available.

Great repository names are short and memorable. Need inspiration? How about fluffy-umb

**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.](#)

**Info** You are creating a public repository in your personal account.

3. Выполните клонирование созданного репозитория. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```

M@DESKTOP-UVM9NOL MINGW64 ~/Desktop (master)
$ git clone https://github.com/obrMaria/OPI_4.1.git
Cloning into 'OPI_4.1'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.

M@DESKTOP-UVM9NOL MINGW64 ~/Desktop (master)
$ cd OPI_4.1

```

Рисунок – клонирование созданного репозитория

#### 4. Проработать примеры лабораторной работы.

The screenshot shows a code editor with four tabs labeled 1.py, 2.py, 3.py, and 4.py. The active tab is 1.py, which contains the following Python code:

```

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

```

Below the code editor, a terminal window is open, showing the output of the script:

```

C:\Users\M\Desktop\OPI_4.1\PY\ind\venv\Scripts\
paper
paperback
[]
Process finished with exit code 0

```

Рисунок – Атрибут класса

```
2.py x 3.py 4.py 5.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  usages new *
6  class River:
7      # список всех рек
8      all_rivers = []
9
10     new *
11     def __init__(self, name, length):
12         self.name = name
13         self.length = length
14         # добавляем текущую реку в список
15         River.all_rivers.append(self)
16
17     if __name__ == '__main__':
18         volga = River("Волга", 3530)
19         seine = River("Сена", 776)
20         nile = River("Нил", 6852)
21         # далее печатаем все названия
22         for river in River.all_rivers:
23             print(river.name)
24
25     if __name__ == '__main__' > for river in River.all_rive
```

2 x

C:\Users\M\Desktop\OPI\_4.1\PY\ind\venv\Script

Волга  
Сена  
Нил

Рисунок – Атрибуты и элементы класса

```
4.py x 3.py 5.py ind_1.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  1 usage
6  class Ship:
7      def __init__(self, name, capacity):
8          self.name = name
9          self.capacity = capacity
10         self.cargo = 0
11
12     2 usages
13     def load_cargo(self, weight):
14         if self.cargo + weight <= self.capacity:
15             self.cargo += weight
16             print("Loaded {} tons".format(weight))
17         else:
18             print("Cannot load that much")
19
20     2 usages
21     def unload_cargo(self, weight):
22         if self.cargo - weight >= 0:
23             self.cargo -= weight
24             print("Unloaded {} tons".format(weight))
25         else:
26             print("Cannot unload that much")
27
28 Ship > __init__()
29
30 C:\Users\M\Desktop\OPI_4.1\PY\ind\venv\Scripts>python ind_1.py
31 Jack Sparrow is the captain of the Black Pearl
32 Jack Sparrow
33 Loaded 600 tons
34 Unloaded 400 tons
35 Cannot load that much
36 Cannot unload that much
37
38 Process finished with exit code 0
```

Рисунок – Изменение атрибутов с помощью методов

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  1 usage
6  class Rectangle:
7      def __init__(self,
8          self.__width =
9          self.__height
10
11      3 usages
12      @property
13      def width(self):
14          return self.__
15
16      1 usage
17      @width.setter
18      def width(self, w)
19          if w > 0:
20              self.__wic
21          else:
22              raise Valu
23
24      3 usages
25      @property

```

Python 5 x

C:\Users\M\Desktop\OPI\_4.1\P

10

20

50

70

Process finished with exit c

Рисунок – Свойства

The image shows a Python IDE with a file explorer on the left and a code editor. The code editor displays a Python script with the following content:

```
102
103     def greater(self, rhs):
104         if isinstance(rhs, Rational):
105             v1 = self.numerator / self.den
106             v2 = rhs.numerator / rhs.denom
107             return v1 > v2
108         else:
109             return False
110
111     def less(self, rhs):
112         if isinstance(rhs, Rational):
113             v1 = self.numerator / self.den
114             v2 = rhs.numerator / rhs.denom
115             return v1 < v2
116         else:
117             return False
118
119
120 ► if __name__ == '__main__':
121     r1 = Rational(3, 4)
122     r1.display()
123     r2 = Rational()
124     r2.read("Введите обыкновенную дробь: ")
125     r2.display()
126     r3 = r2.add(r1)
```

Below the code editor, there is a 'Run' button and a '6' icon. To the right of the 'Run' button is a vertical toolbar with icons for running, stepping through, and other debugging actions. The output window shows the following text:

```
3/4
Введите обыкновенную дробь: 5/6
5/6
19/12
1/12
5/8
10/9
```

Рисунок – пример

## 5. Выполнить индивидуальные 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.

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

1.(21) Поле `first` — дробное число; поле `second` — целое число, показатель степени. Реализовать метод `power()` — возведение числа `first` в степень `second`. Метод должен правильно работать при любых допустимых значениях `first` и `second`.



The screenshot shows a Python IDE with two tabs: '6.py' and 'ind\_1.py'. The 'ind\_1.py' tab is active, displaying a class named 'Number' with two methods: '\_\_init\_\_' and 'read'. The '\_\_init\_\_' method takes 'first' and 'second' as arguments and assigns them to 'self.first' and 'self.second'. The 'read' method prompts the user for a floating-point number and an integer, storing them in 'self.first' and 'self.second' respectively. Below the code, the Python interpreter window shows the execution of 'ind\_1.py'. It displays the calculation  $3^4 = 81$ , prompts for a floating-point number (1.2) and an integer (5), and then shows the calculation  $1.2^5 = 2.4883199999999994$ . A traceback follows, indicating a 'ValueError' was raised in the 'make\_power' function at line 38.

```
20
21
22 class Number:
23     def __init__(self, first=0.0, second=0):
24         self.first = first
25         self.second = second
26
27     def read(self):
28         self.first = float(input("Введите дробное число >> "))
29         self.second = int(input("Введите целое число >> "))
30
Number > __init__()

C:\Users\M\Desktop\OPI_4.1\PY\ind\venv\Scripts\python.exe C:\Users\M\Desktop\OPI_4.1\PY\ind\ind_1.py
3 ^ 4 = 81
Введите дробное число >> 1.2
Введите целое число >> 5
1.2 ^ 5 = 2.4883199999999994
Traceback (most recent call last):
  File "C:\Users\M\Desktop\OPI_4.1\PY\ind\ind_1.py", line 50, in <module>
    test = make_power(0, 1.2)
           ^^^^^^^^^^^^^^^^^
  File "C:\Users\M\Desktop\OPI_4.1\PY\ind\ind_1.py", line 38, in make_power
    raise ValueError
ValueError
```

Рисунок – Индивидуальное задание №1

## 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` .

В раздел программы, начинающийся после инструкции `if __name__ =`

'\_\_main\_\_': добавить код, демонстрирующий возможности разработанного класса.

6.(21) Создать класс Point для работы с точками на плоскости. Координаты точки — декартовы. Обязательно должны быть реализованы: перемещение точки по оси  $x$ , перемещение по оси  $y$ , определение расстояния до начала координат, расстояния между двумя точками, преобразование в полярные координаты, сравнение на совпадение и несовпадение.

```
6.py x ind_1.py ind_2.py x
48 point1.move_y(-1) # перемещение точки
49
50 print("новые координаты первой точки: ")
51 point1.display() # Вывод новых координат
52
53 distance_center = point1.distance_to_origin
54 print("Расстояние до центра от первой точки")
55
56 distance = point1.distance_to_point(point2)
57 print("Расстояние между точками:", distance)
58
59 polar_coords = point1.to_polar_coordinates
60 print("Полярные координаты первой точки:", polar_coords)
61
62 are_equal = point1.compare(point2) # Сравнение
63 if are_equal:
64     print("Точки совпадают")
65 else:
66     print("Точки не совпадают")
__name__ == '__main__':
```

ind\_2 x ind\_2 x

⋮

C:\Users\M\Desktop\OPI\_4.1\PY\ind\venv\Scripts\python.exe ind\_2.py

Введите значение координаты x: -2

Введите значение координаты y: 1

Координаты точки: (-2.0, 1.0)

Введите значение координаты x: 0

Введите значение координаты y: 0

Координаты точки: (0.0, 0.0)

новые координаты первой точки:

Координаты точки: (0.0, 0.0)

Расстояние до центра от первой точки: 0.0

Расстояние между точками: 0.0

Полярные координаты первой точки: (0.0, 0.0)

Точки совпадают

Process finished with exit code 0

Рисунок – Индивидуальное задание №2

## 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`.