

**Учреждение образования  
«Брестский государственный технический университет»  
Кафедра ИИТ**

**Лабораторная работа №2  
По дисциплине «Современные платформы программирования»**

**Выполнил:**

Студент 3 курса

Группы ПО-12

Авдей Д.Б.

**Проверил:**

Кулик А.Д.

**Брест 2026**

**Цель:** закрепить навыки объектно-ориентированного программирования на языке Python.

## Вариант №1

### Задание 1

Реализовать простой класс.

Требования к выполнению:

- Реализовать пользовательский класс по варианту.

Для каждого класса

- Создать атрибуты (поля) классов
- Создать методы классов
- Добавить необходимые свойства и сеттеры (по необходимости)
- Переопределить магические методы `__str__` и `__eq__`

1) Равнобедренный треугольник, заданный длинами сторон – Предусмотреть возможность определения площади и периметра, а также логический метод, определяющий существует ли такой треугольник. Конструктор должен позволять создавать объекты с начальной инициализацией. Переопределить метод `__eq__`, выполняющий сравнение объектов данного типа.

### Результат:

```
t1 = IsoscelesTriangle(equal_side: 5, base: 6)
t2 = IsoscelesTriangle(equal_side: 5, base: 6)

print(t1)
print("Существует:", t1.exists())
print("Периметр:", t1.perimeter())
print("Площадь:", t1.area())
print("Треугольники равны:", t1 == t2)
```

```
D:\Python\python.exe E:\program\spp_po12\reports\Avdey\Lab_2\sr
Равнобедренный треугольник: боковая сторона = 5, основание = 6
Существует: True
Периметр: 16
Площадь: 12.0
Треугольники равны: True
```

### Код:

```
import math

class IsoscelesTriangle:
    def __init__(self, equal_side: float = 1, base: float = 1):
        self.equal_side = equal_side
        self.base = base

    @property
    def equal_side(self):
        return self._equal_side

    @equal_side.setter
    def equal_side(self, value):
        if value <= 0:
            raise ValueError("Длина стороны должна быть положительной")
        self._equal_side = value

    @property
    def base(self):
        return self._base

    @base.setter
    def base(self, value):
        if value <= 0:
            raise ValueError("Длина основания должна быть положительной")
        self._base = value

    def exists(self) -> bool:
        return 2 * self.equal_side > self.base

    def perimeter(self) -> float:
        if not self.exists():
            raise ValueError("Треугольник не существует")
        return 2 * self.equal_side + self.base

    def area(self) -> float:
        if not self.exists():
            raise ValueError("Треугольник не существует")
        height = math.sqrt(self.equal_side**2 - (self.base / 2) ** 2)
        return 0.5 * self.base * height

    def __str__(self):
        return f"Равнобедренный треугольник: " f"боковая сторона =
```

```

{self.equal_side}, " f"основание = {self.base}"

    def __eq__(self, other):
        if not isinstance(other, IsoscelesTriangle):
            return NotImplemented
        return self.equal_side == other.equal_side and self.base == other.base

t1 = IsoscelesTriangle(5, 6)
t2 = IsoscelesTriangle(5, 6)

print(t1)
print("Существует:", t1.exists())
print("Периметр:", t1.perimeter())
print("Площадь:", t1.area())
print("Треугольники равны:", t1 == t2)

```

## Задание 2

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы

1) Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

## Результат:

```

Преподаватель Иванов объявил курс «Python»
Студент Алексей записался на курс «Python»
Студент Мария записался на курс «Python»
Студент Алексей обучается
Студент Мария обучается
Преподаватель Иванов поставил студенту Алексей оценку 5 по курсу «Python»
Преподаватель Иванов поставил студенту Мария оценку 4 по курсу «Python»

Архив оценок:
Студент: Алексей, Курс: Python, Оценка: 5
Студент: Мария, Курс: Python, Оценка: 4

Process finished with exit code 0

```

## Код:

```
from abc import ABC, abstractmethod

class Grading(ABC):
    @abstractmethod
    def set_grade(self, student, course, grade):
        pass

class Person:
    def __init__(self, name):
        self.name = name

class Student(Person):
    def __init__(self, name):
        super().__init__(name)
        self.courses = []

    def enroll(self, course):
        self.courses.append(course)
        course.add_student(self)

    def study(self):
        print(f"Студент {self.name} обучается")

class Teacher(Person, Grading):
    def announce_course(self, course):
        print(f"Преподаватель {self.name} объявил курс «{course.title}»")

    def set_grade(self, student, course, grade):
        Archive.save_grade(student, course, grade)
        print(
            f"Преподаватель {self.name} поставил студенту {student.name} "
            f"оценку {grade} по курсу «{course.title}»"
        )

class Course:
    def __init__(self, title):
        self.title = title
        self.students = []

    def add_student(self, student):
        self.students.append(student)
        print(f"Студент {student.name} записался на курс «{self.title}»")

class Archive:
    grades = []

    @classmethod
    def save_grade(cls, student, course, grade):
        cls.grades.append({"student": student.name, "course": course.title,
                           "grade": grade})
```

```

    @classmethod
    def show_archive(cls):
        print("\nАрхив оценок:")
        for record in cls.grades:
            print(f"Студент: {record['student']}, " f"Курс: {record['course']}, " f"Оценка: {record['grade']}")

if __name__ == "__main__":

    teacher = Teacher("Иванов")

    student1 = Student("Алексей")
    student2 = Student("Мария")

    python_course = Course("Python")

    teacher.announce_course(python_course)

    student1.enroll(python_course)
    student2.enroll(python_course)

    student1.study()
    student2.study()

    teacher.set_grade(student1, python_course, 5)
    teacher.set_grade(student2, python_course, 4)

    Archive.show_archive()

```

**Вывод:** закрепил навыки объектно-ориентированного программирования на языке Python.