

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №24
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Элементы объектно-ориентированного программирования в языке Python.

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

Ход работы.

1. Создание репозитория.


Create a new repository

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

Required fields are marked with an asterisk ().*

Owner *

Repository name *

 BigLofanbl4 ▾


/

lab24


✔ lab24 is available.

Great repository names are short and memorable. Need inspiration? How about **fuzzy-octo-fortnight** ?

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.

Create repository

Рисунок 1.1 – Создание репозитория

2. Выполнение примеров из лабораторной работы.

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

class Rational:
    def __init__(self, a=0, b=1):
        a = int(a)
        b = int(b)
        if b == 0:
            raise ValueError()
        self.__numerator = abs(a)
        self.__denominator = abs(b)
        self.__reduce()

    # Сокращение дроби
    def __reduce(self):
        # Функция для нахождения наибольшего общего делителя
        def gcd(a, b):
            if a == 0:
                return b
            elif b == 0:
                return a
            elif a >= b:
                return gcd(a % b, b)
            else:
                return gcd(a, b % a)

        c = gcd(self.__numerator, self.__denominator)
        self.__numerator //= c
        self.__denominator //= c

    @property
    def numerator(self):
        return self.__numerator
```

Рисунок 2.1 – Код примера

3. Выполнение первого задания: поле `first` — дробное положительное число, катет прямоугольного треугольника; поле `second` — дробное положительное число, катет прямоугольного треугольника. Реализовать метод `hypotenuse()` — вычисление гипотенузы.

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

import math

class RightTriangle:
    def __init__(self, first, second):
        if not isinstance(first, (int, float)) or not isinstance(second, (int, float)):
            raise ValueError("The first and second inputs must be numbers")
        if first <= 0 or second <= 0:
            raise ValueError("The numbers must be > 0")
        self.first = first
        self.second = second

    def read(self):
        try:
            first = float(input("Enter first value > 0: "))
            second = float(input("Enter second value > 0: "))
            if first <= 0 or second <= 0:
                raise ValueError("The numbers must be > 0")
            self.first = first
            self.second = second
        except ValueError as error:
            print(error)
            self.read()

    def display(self):
        print(f"The catheter a={self.first}, the catheter b={self.second}")

    def hypotenuse(self):
        return math.sqrt(self.first**2 + self.second**2)

def make_right_triangle(first, second):
    try:
        return RightTriangle(first, second)
    except ValueError as error:
        print(error)

if __name__ == "__main__":
    triangle1 = make_right_triangle(3, 4)
    triangle1.display()
    print(triangle1.hypotenuse())

    triangle2 = make_right_triangle(3, 4)
    triangle2.read()
    triangle2.display()
    print(triangle2.hypotenuse())
```

Рисунок 3.1 – Код программы

```
The catheter a=3, the catheter b=4
5.0
Enter first value > 0: 3
Enter second value > 0: 2
The catheter a=3.0, the catheter b=2.0
3.605551275463989
```

Рисунок 3.2 – Вывод

4. Выполнение второго задания: создать класс Goods (товар). В классе должны быть представлены поля: наименование товара, дата оформления, цена товара, количество единиц товара, номер накладной, по которой товар поступил на склад. Реализовать методы изменения цены товара, изменения количества товара (увеличения и уменьшения), вычисления стоимости товара.

```
class Goods:
    def __init__(self, name, date, price, amount, invoice_number):
        self.name = str(name)
        self.date = self._validate_date(date)
        self.price = self._validate_price(price)
        self.amount = self._validate_amount(amount)
        self.invoice_number = str(invoice_number)

    def _validate_date(self, date_str):
        try:
            return datetime.strptime(date_str, "%Y-%m-%d")
        except ValueError:
            raise ValueError("Incorrect date")

    def _validate_price(self, value):
        value = float(value)
        if value < 0:
            raise ValueError("Price must be positive")
        return value

    def _validate_amount(self, value):
        value = int(value)
        if value < 0:
            raise ValueError("Amount must be positive")
        return value

    def read(self):
        self.name = input("Enter good's name: ")
        self.date = self._validate_date(input("Enter date (YYYY-MM-DD): "))
        self.price = self._validate_price(input("Enter price: "))
        self.amount = self._validate_amount(input("Enter amount: "))
        self.invoice_number = input("Enter invoice number: ")

    def display(self):
        print(f"Good's name: {self.name}")
        print(f>Date: {self.date}")
        print(f"Price: {self.price}")
        print(f"Amount: {self.amount}")
        print(f"Invoice number: {self.invoice_number}")

    def change_price(self, new_price):
        self.price = new_price

    def increase_amount(self, amount):
        self.amount += amount

    def decrease_amount(self, amount):
        if self.amount >= amount:
            self.amount -= amount
        else:
            print("Not enough goods")

    def calc_total_cost(self):
        return self.amount * self.price
```

Рисунок 4.1 – Код программы

```
Good's name: Desktop
Date: 2024-04-26 00:00:00
Price: 50000.99
Amount: 10
Invoice number: INV123
Good's name: Desktop
Date: 2024-04-26 00:00:00
Price: 55000.0
Amount: 12
Invoice number: INV123
660000.0
```

Рисунок 4.2 – Вывод программы

Ответы на контрольные вопросы

1. Объявление класса в Python:

Класс создаётся с использованием ключевого слова «class», за которым следует имя класса и двоеточие. В теле класса определяются методы и атрибуты.

2. Различие между атрибутами класса и атрибутами экземпляра:

- Атрибуты класса объявляются внутри класса, но вне любых методов.

Они являются общими для всех экземпляров класса.

- Атрибуты экземпляра определяются внутри методов класса и принадлежат конкретному экземпляру класса. Для их определения используется «self», который указывает на текущий объект.

3. Назначение методов класса:

Методы класса воздействуют на состояние класса в целом или используются для реализации поведения, характерного для всех экземпляров класса, не требуя создания экземпляра класса. Для того чтобы метод рассматривался как метод класса, его необходимо декорировать декоратором «@classmethod», и его первым параметром должен быть «cls», который ссылается на сам класс.

4. Назначение метода «__init__()»:

Метод «__init__()» называют конструктором класса. Он вызывается автоматически при создании нового экземпляра класса. Этот метод используется для инициализации экземпляра, установки начальных значений атрибутов объекта.

5. Назначение «self»:

«self» в методах класса представляет текущий экземпляр объекта. Он используется для доступа к атрибутам и другим методам этого объекта изнутри класса. «self» аналогичен «this» в других языках программирования.

6. Добавление атрибутов в класс:

Атрибуты класса добавляются путём объявления их в теле класса. Атрибуты экземпляра обычно добавляются в методе «__init__» или других методах экземпляра с использованием «self».

7. Управление доступом к методам и атрибутам:

В Python управление доступом можно осуществлять с помощью указания префиксов для имён атрибутов и методов (одно подчеркивание для "защищенных" и двойное подчеркивание для "приватных"). Однако это больше соглашения, нежели жесткие правила доступа, как в некоторых других языках.

8. Назначение функции «isinstance()»:

Функция «isinstance()» проверяет, принадлежит ли объект к заданному классу или его подклассам. Если объект является экземпляром класса или наследуется от него, функция возвращает «True». Например, «isinstance(obj, MyClass)» вернет «True», если «obj» является экземпляром «MyClass» или его подкласса.