

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
№220 дисциплины «Основы программной инженерии»

Выполнил:

Джараян Арег Александрович
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: Лабораторная работа 4.1 Элементы объектно-ориентированного программирования в языке Python.

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

Ход работы.

1. Создание нового репозитория с лицензией MIT.

Required fields are marked with an asterisk (*).

Owner * / Repository name *
aregdz / 9lab
9lab is available.

Great repository names are short and memorable. Need inspiration? How about [studious-octo-winner](#) ?

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.](#)

i You are creating a public repository in your personal account.

Create repository

Рисунок 1 – создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/9
$ git clone "https://github.com/aregdz/9lab.git"
Cloning into '9lab'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/9
$
```

Рисунок 2 – клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.

```
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
```

Рисунок 4 – Файл .gitignore

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/9/9lab (main)
$ git checkout -b develop
Switched to a new branch 'develop'
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/9/9lab (develop)
$ |
```

Рисунок 4 – организация ветки

```
(venv) PS D:\Рабочий стол\4 сем
Package      Version
-----
black        24.4.0
cfgv         3.4.0
click        8.1.7
colorama     0.4.6
distlib      0.3.8
pyflakes     3.2.0
PyYAML       6.0.1
setuptools   69.5.1
virtualenv   20.25.2
```

Рисунок 5 – создание виртуального окружения

4.Пример 1.

```
#!/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()

    def __reduce(self):
        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

@property
def denominator(self):
    return self.__denominator

def read(self, prompt=None):
    line = input() if prompt is None else input(prompt)
    parts = list(map(int, line.split('/', maxsplit=1)))
    if parts[1] == 0:
        raise ValueError()
    self.__numerator = abs(parts[0])
    self.__denominator = abs(parts[1])
    self.__reduce()

def display(self):
    print(f"{self.__numerator}/{self.__denominator}")

def add(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.denominator + self.denominator * rhs.numerator
        b = self.denominator * rhs.denominator
        return Rational(a, b)
    else:
        raise ValueError()

def sub(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.denominator - self.denominator * rhs.numerator
        b = self.denominator * rhs.denominator
        return Rational(a, b)
    else:
        raise ValueError()

def mul(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.numerator
        b = self.denominator * rhs.denominator
        return Rational(a, b)
    else:
        raise ValueError()

def div(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.denominator
        b = self.denominator * rhs.numerator
        if b == 0:
            raise ValueError("Деление на ноль.")
        return Rational(a, b)
    else:
        raise ValueError()

def equals(self, rhs):
    if isinstance(rhs, Rational):
        return self.numerator == rhs.numerator and self.denominator ==
rhs.denominator
    else:
        return False

def greater(self, rhs):
    if isinstance(rhs, Rational):
        return self.numerator / self.denominator > rhs.numerator / rhs.denominator

```

```

        else:
            return False

    def less(self, rhs):
        if isinstance(rhs, Rational):
            return self.numerator / self.denominator < rhs.numerator / rhs.denominator
        else:
            return False

if __name__ == '__main__':
    r1 = Rational(3, 4)
    r1.display()
    r2 = Rational()
    r2.read("Введите обыкновенную дробь: ")
    r2.display()
    r3 = r2.add(r1)
    r3.display()
    r4 = r2.sub(r1)
    r4.display()
    r5 = r2.mul(r1)
    r5.display()
    r6 = r2.div(r1)
    r6.display()

```

Рисунок 6 – пример 1

```

D:\Рабочий стол\4 семестр\опи\9\9lab\venv\Scripts\python.exe "D:\Рабочий стол\4 семестр\опи\9\9lab\venv\Scripts\python.exe" "D:\Рабочий стол\4 семестр\опи\9\9lab\venv\Scripts\python.exe"
3/4
Введите обыкновенную дробь: 5/6
5/6
19/12
1/12
5/8
10/9

Process finished with exit code 0

```

Рисунок 7 – выполнение примера 1

5. Поле `first` — целое положительное число, калорийность 100 г продукта; поле `second` — дробное положительное число, масса продукта в килограммах. Реализовать метод `power()` — вычисление общей калорийности

продукта.

```
class Pair:
    def __init__(self, first, second):
        self.set_first(first)
        self.set_second(second)

    def set_first(self, value):
        if isinstance(value, int) and value > 0:
            self.first = value
        else:
            raise ValueError("НЕ правильное значение.")

    def set_second(self, value):
        if isinstance(value, float) and value > 0:
            self.second = value
        else:
            raise ValueError("НЕ правильное значение.")

    def read(self):
        first = int(input("Введите целое положительное число для 'first': "))
        second = float(input("Введите дробное положительное число для 'second': "))
        self.set_first(first)
        self.set_second(second)

    def display(self):
        print(f"Калорийность 100 г продукта: {self.first} ккал,\nМасса продукта: {self.second} кг.")

    def power(self):
        return self.first * self.second * 10

def make_pair(first, second):
    try:
        pair = Pair(first, second)
        return pair
    except ValueError as ve:
        print(ve)
        return None

if __name__ == '__main__':
    try:
        first = int(input("Введите калорийность 100 г продукта: "))
        second = float(input("Введите массу продукта в килограммах: "))
        my_pair = make_pair(first, second)
        if my_pair:
            my_pair.display()
            print(f"Общая калорийность продукта: {my_pair.power()} ккал.")
    except Exception as e:
        print("Введены некорректные данные.")

class Pair:
    def __init__(self, first, second):
        self.set_first(first)
        self.set_second(second)

    def set_first(self, value):
        if isinstance(value, int) and value > 0:
            self.first = value
        else:
            raise ValueError("НЕ правильное значение.")

    def set_second(self, value):
        if isinstance(value, float) and value > 0:
            self.second = value
        else:
            raise ValueError("НЕ правильное значение.")
```

```

def read(self):
    first = int(input("Введите целое положительное число для 'first': "))
    second = float(input("Введите дробное положительное число для 'second': "))
    self.set_first(first)
    self.set_second(second)

def display(self):
    print(f"Калорийность 100 г продукта: {self.first} ккал,\nМасса продукта: {self.second} кг.")

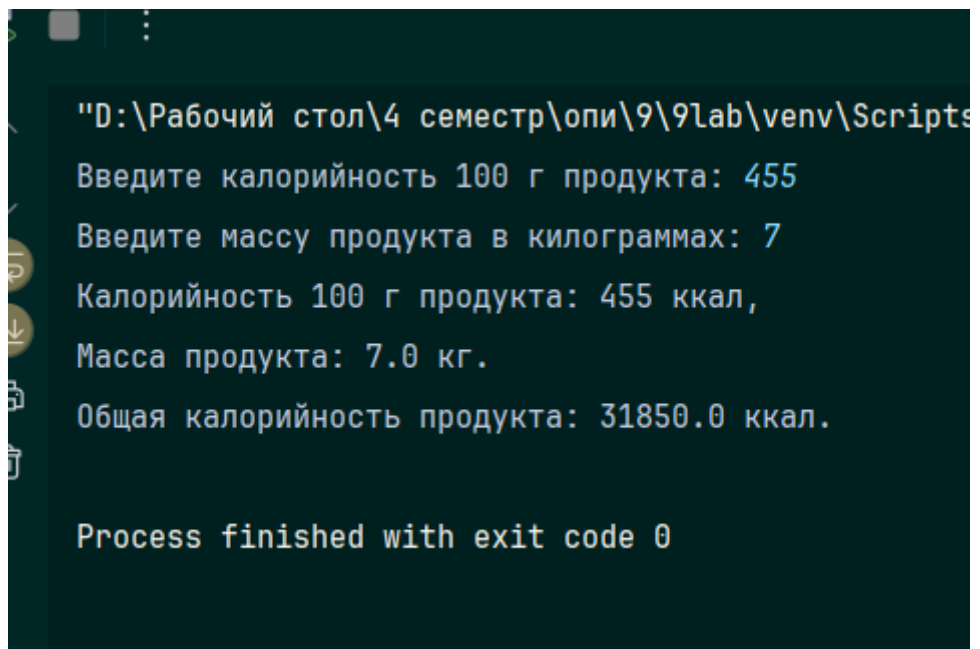
def power(self):
    return self.first * self.second * 10

def make_pair(first, second):
    try:
        pair = Pair(first, second)
        return pair
    except ValueError as ve:
        print(ve)
        return None

if __name__ == '__main__':
    try:
        first = int(input("Введите калорийность 100 г продукта: "))
        second = float(input("Введите массу продукта в килограммах: "))
        my_pair = make_pair(first, second)
        if my_pair:
            my_pair.display()
            print(f"Общая калорийность продукта: {my_pair.power()} ккал.")
    except Exception as e:
        print("Введены некорректные данные.")

```

Рисунок 8 – Выполнение задания 1



```

"D:\Рабочий стол\4 семестр\опи\9\9lab\venv\Scripts
Введите калорийность 100 г продукта: 455
Введите массу продукта в килограммах: 7
Калорийность 100 г продукта: 455 ккал,
Масса продукта: 7.0 кг.
Общая калорийность продукта: 31850.0 ккал.

Process finished with exit code 0

```

Рисунок 9 – пример выполнение задания 1

6. 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.

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

import math

class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def move_x(self, delta_x):
        """Перемещение точки по оси X."""
        self.x += delta_x

    def move_y(self, delta_y):
        """Перемещение точки по оси Y."""
        self.y += delta_y

    def distance_to_origin(self):
        """Определение расстояния до начала координат."""
        return math.sqrt(self.x**2 + self.y**2)

    def distance_to_point(self, other):
        """Расстояние между двумя точками."""
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)

    def to_polar(self):
        """Преобразование в полярные координаты."""
        r = self.distance_to_origin()
        theta = math.atan2(self.y, self.x)
        return (r, theta)

    def __eq__(self, other):
        """Сравнение на совпадение."""
        return self.x == other.x and self.y == other.y

    def __ne__(self, other):
        """Сравнение на несовпадение."""
        return not self.__eq__(other)

    def __str__(self):
        """Строковое представление класса."""
        return f"Point({self.x}, {self.y})"

# Пример использования класса Point
if __name__ == '__main__':
    point1 = Point(3, 4)
    point2 = Point(6, 8)

    print("Первая точка:", point1)
    print("Вторая точка:", point2)

    point1.move_x(2)
    point1.move_y(-1)
    print("Первая точка после перемещения:", point1)

    origin_dist = point1.distance_to_origin()
    print("Расстояние от первой точки до начала координат:", origin_dist)
```

```

points_dist = point1.distance_to_point(point2)
print("Расстояние между первой и второй точками:", points_dist)

polar_coords = point1.to_polar()
print("Полярные координаты первой точки:", polar_coords)

print("Точки совпадают:" if point1 == point2 else "Точки не совпадают:")

```

Рисунок 10 – выполнение задания 2

```

"D:\Рабочий стол\4 семестр\опи\9\9lab\venv\Scripts\python.exe" "D:\Рабочий стол\4 семестр\опи\9
Первая точка: Point(3, 4)
Вторая точка: Point(6, 8)
Первая точка после перемещения: Point(5, 3)
Расстояние от первой точки до начала координат: 5.830951894845301
Расстояние между первой и второй точками: 5.0990195135927845
Полярные координаты первой точки: (5.830951894845301, 0.5404195002705842)
Точки не совпадают:

Process finished with exit code 0

```

Рисунок 11 – пример выполнения задания 2

```

Collecting isort
  Using cached isort-5.13.2-py3-none-any.whl.metadata (12 kB)
create mode 100644 .idea/vcs.xml
create mode 100644 .pre-commit-config.yaml
create mode 100644 1primer.py
create mode 100644 1zadanie.py
create mode 100644 2zadanie.py
create mode 100644 README (1).md
create mode 100644 environment.yml
create mode 100644 pyproject.toml
(venv) PS D:\Рабочий стол\4 семестр\опи\9\9lab> git push origin develop
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (18/18), 4.51 KiB | 1.13 MiB/s, done.
Total 18 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/aregdz/9lab/pull/new/develop
remote:
To https://github.com/aregdz/9lab.git
* [new branch]      develop -> develop

```

Рисунок 12 - фиксация изменений

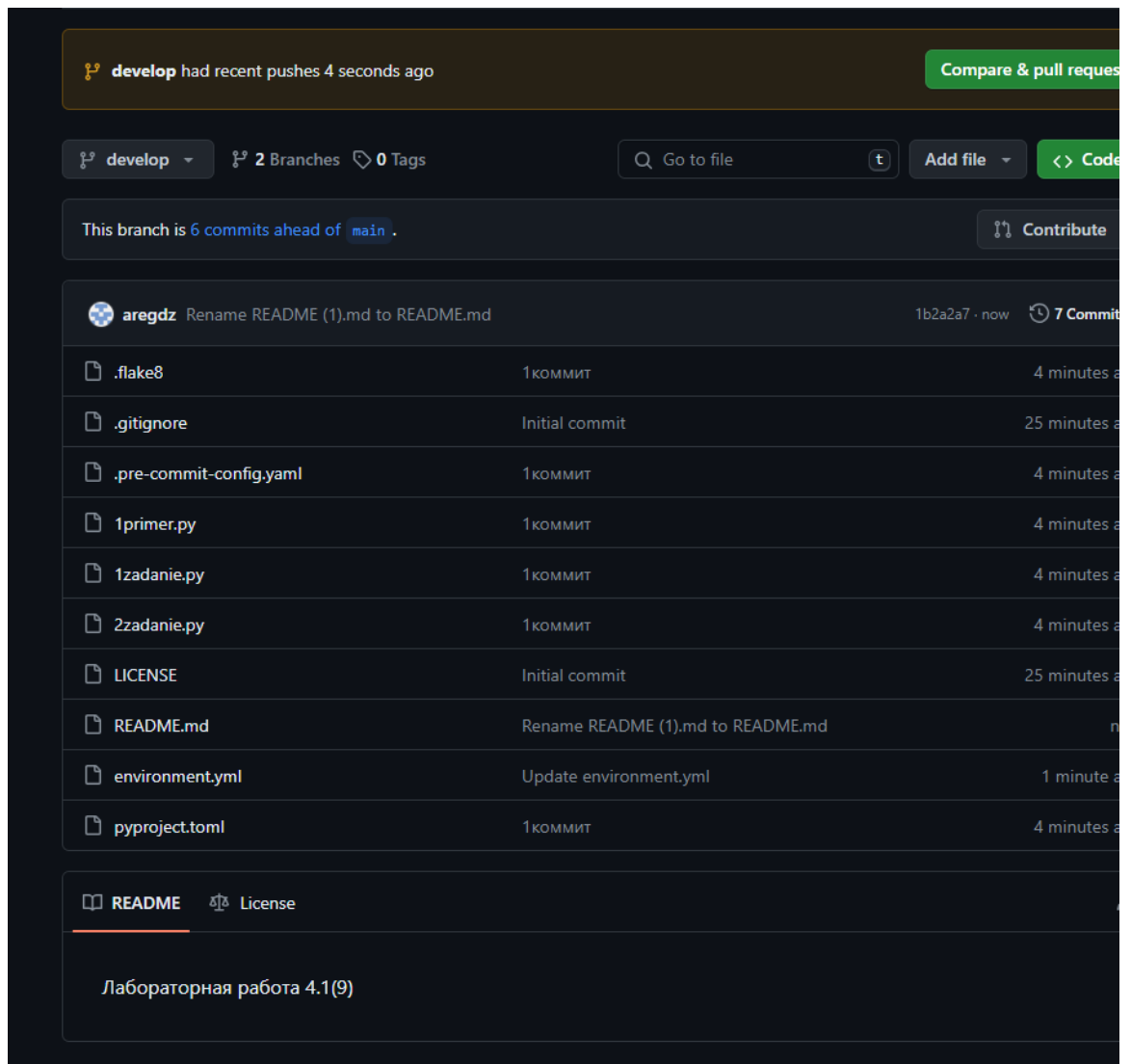


Рисунок 13 – ветка develop в git

Контрольные вопросы:

1.Объявление класса в Python осуществляется с помощью ключевого слова class, за которым следует имя класса и двоеточие. Тело класса содержит определения методов и атрибутов.

2.Атрибуты класса — это переменные, которые связаны с классом, а не с отдельными экземплярами. Они же являются своего рода "статическими" переменными. Атрибуты экземпляра определяются внутри методов класса и привязаны к конкретным объектам класса через self.

3.Методы класса используются для определения поведения экземпляров класса, функциональности класса в целом, а также для взаимодействия экземпляров класса с его атрибутами.

4.Метод `__init__()` является конструктором класса и вызывается при создании нового экземпляра класса. Он используется для инициализации атрибутов экземпляра.

5.`self` – это ссылка на сам объект экземпляра и используется для доступа к его атрибутам и методам изнутри класса. `self` не передается в метод класса явно, он добавляется автоматически при вызове метода объекта.

6.Чтобы добавить атрибуты в класс, вы должны присвоить им значения внутри метода (обычно в `__init__()`) с использованием синтаксиса `self.name = value`, где `name` - это имя атрибута, а `value` - его значение.

7.Управление доступом к методам и атрибутам в Python происходит с помощью соглашений (одно или двойное подчеркивание перед именем атрибута для его объявления как защищенного или приватного), но на уровне языка все атрибуты и методы публичны.

8.Функция `isinstance()` проверяет, принадлежит ли объект к указанному классу или кортежу классов. Это полезно для проверки типов объектов во время выполнения программы.