Институт цифрового развития
Кафедра инфокоммуникаций

«Элементы объектно-ориентированного программирования в языке Python»

ОТЧЕТ
по лабораторной работе №26
дисциплины
«Основы программной инженерии»

Выполнил:
Мизин Глеб Егорович
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

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


class Book:
    material = "paper"
    cover = "paperback"
    all_books = []


if __name__ == '__main__':
    print(Book.material)
    print(Book.cover)
    print(Book.all_books)
```

Рисунок 1 – Пример 1

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


class River:
    # список всех рек
    all_rivers = []

    def __init__(self, name, length):
        self.name = name
        self.length = length
        # добавляем текущую реку в список всех рек
        River.all_rivers.append(self)


if __name__ == '__main__':
    volga = River("Волга", 3530)
    seine = River("Сена", 776)
    nile = River("Нил", 6852)
    # далее печатаем все названия рек
    for river in River.all_rivers:
        print(river.name)
```

Рисунок 2 – Пример 2

Рисунок 3 – Пример 3

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


class Ship:
    def __init__(self, name, capacity):
        self.name = name
        self.capacity = capacity
        self.cargo = 0

    def load_cargo(self, weight):
        if self.cargo + weight <= self.capacity:
            self.cargo += weight
            print("Loaded {} tons".format(weight))
        else:
            print("Cannot load that much")

    def unload_cargo(self, weight):
        if self.cargo - weight >= 0:
            self.cargo -= weight
            print("Unloaded {} tons".format(weight))
        else:
            print("Cannot unload that much")

    def name_captain(self, cap):
        self.captain = cap
        print("{} is the captain of the {}".format(self.captain, self.name))
```

Рисунок 4 – Пример 4

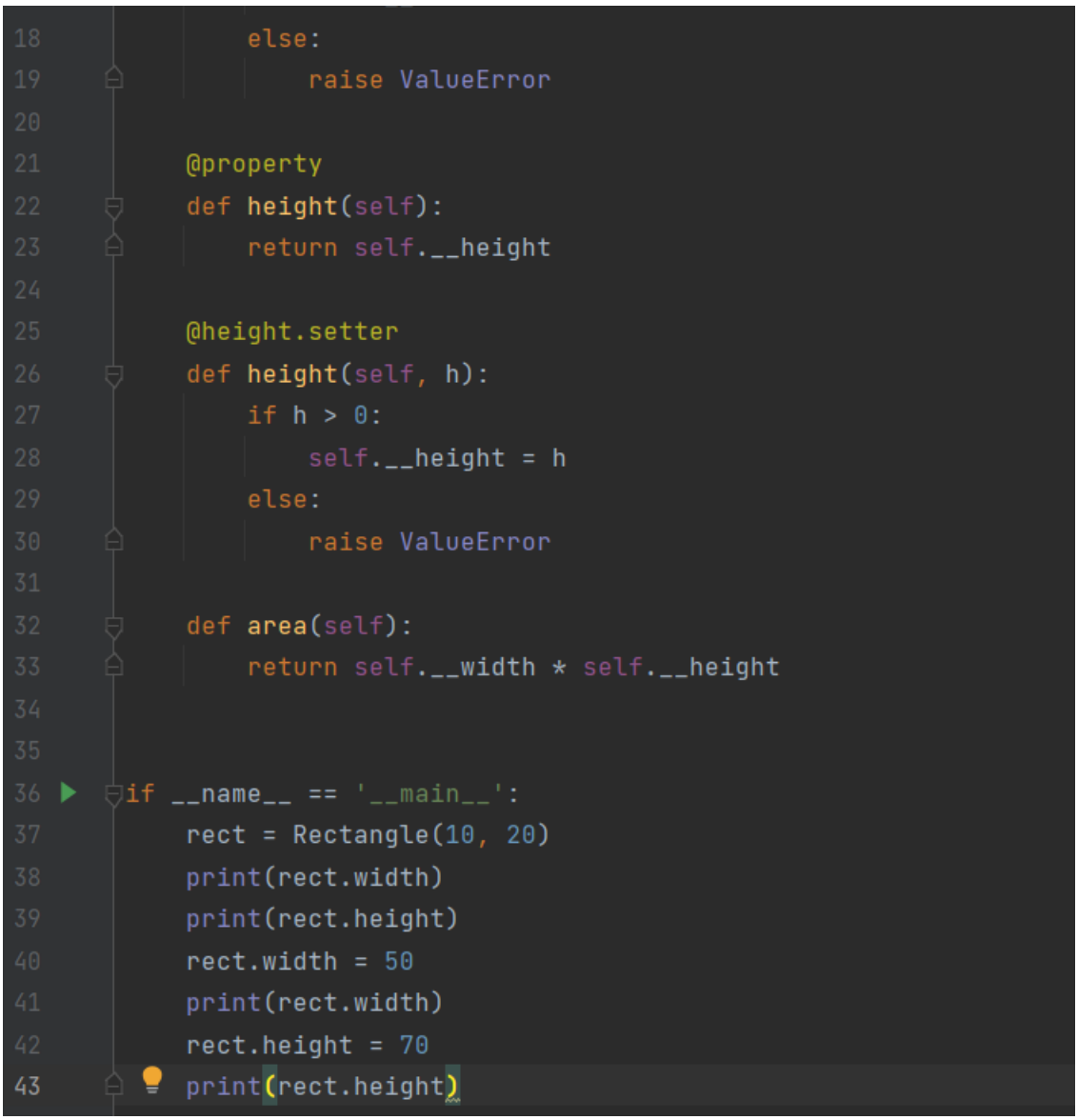

Рисунок 5 – Пример 5

```python
#!/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)
```

Рисунок 6 – Пример 6

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

import math


def is_number(a):
    try:
        float(a)
    except ValueError:
        return False
    return True


def make_cords(first, second):
    if is_number(first) and is_number(second) and first > 0 and second > 0:
        coordinates = Coordinates(first, second)
        return coordinates
    else:
        raise ValueError


class Coordinates:
    def __init__(self, first=0.0, second=0.0):
        if is_number(first) and is_number(second):
            if first > 0 and second > 0:
                self.__first = first
```

Рисунок 7 – Индивидуальное задание №1

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


class Fraction:
    def __init__(self, integer_part=0, fractional_part=0):
        self.integer_part = integer_part
        self.fractional_part = fractional_part

    def read(self):
        self.integer_part = float(input("Enter the integer part: "))
        self.fractional_part = int(input("Enter the fractional part: "))

    def add(self, other):
        integer_part = self.integer_part + other.integer_part
        fractional_part = self.fractional_part + other.fractional_part
        carry = fractional_part // 100
        fractional_part %= 100
        integer_part += carry
        self.__add = f'{integer_part}.{fractional_part}'
        return self.__add

    def sub(self, other):
        integer_part = self.integer_part - other.integer_part
        fractional_part = self.fractional_part - other.fractional_part
        while fractional_part < 0:
            integer_part -= 1
```

Рисунок 8 – Индивидуальное задание №2

Контрольные вопросы

1. Как осуществляется объявление класса в языке Python?

Объявление класса в Python осуществляется с помощью ключевого слова class, за которым следует имя класса и двоеточие. Например, class MyClass:

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса принадлежат классу в целом и доступны всем его экземплярам. Атрибуты экземпляра, наоборот, принадлежат конкретному экземпляру класса и могут быть разными у разных объектов.

3. Каково назначение методов класса?

Методы класса являются функциями, которые связаны с классом и могут быть вызваны у его экземпляров. Они часто используются для работы с атрибутами экземпляра, их изменения или чтения.

4. Для чего предназначен метод __init__() класса?

__init__() - это метод класса, который вызывается при создании экземпляра объекта. Он используется для установки начальных значений атрибутов объекта.

5. Каково назначение self ?

self - это первый параметр всех методов класса в Python. Данный параметр используется для доступа и изменения атрибутов объекта, от которого вызывается метод.

6. Как добавить атрибуты в класс?

Для добавления атрибутов в класс нужно просто создать новые атрибуты в методе класса или за его пределами, используя имя класса, например: MyClass.new_attribute = "value".

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

Управление доступом к методам и атрибутам в Python осуществляется с помощью модификаторов доступа: public, private и protected. В Python все атрибуты и методы класса по умолчанию являются публичными, т.е. доступ к ним возможен из любого места программы. Для создания приватных атрибутов и методов используется синтаксис с двойным подчеркиванием (__), а для создания защищенных атрибутов и методов одинарным подчеркиванием (_).

8. Каково назначение функции isinstance ?

Функция isinstance() возвращает True, если переданный объект является экземпляром указанного класса или его потомком. Она используется для проверки типа объекта в Python.