

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

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

Выполнил студент группы ИВТ-б-о-21-1

Лысенко И.А. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

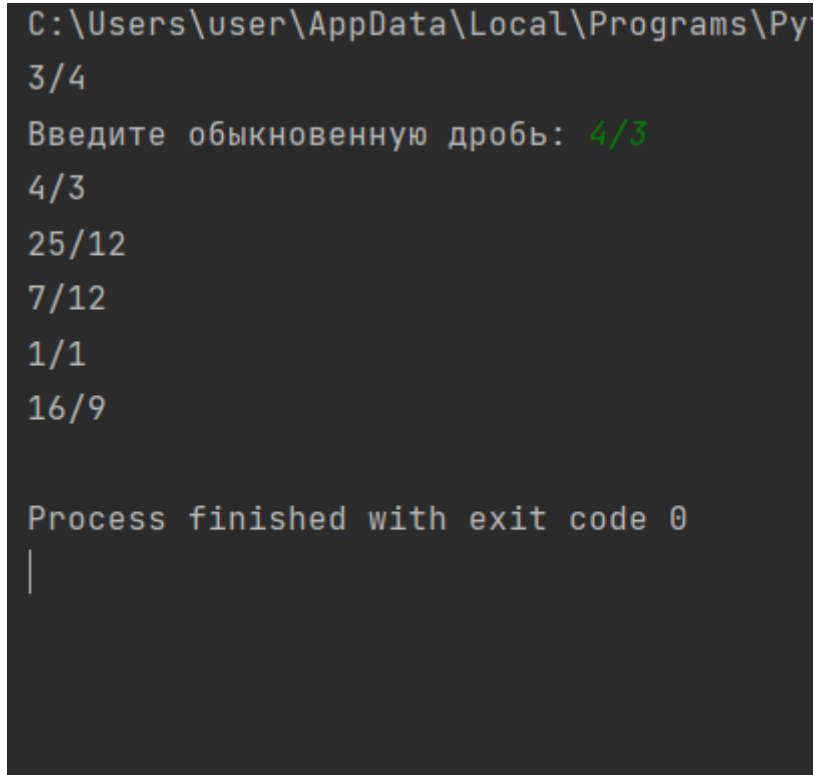
(подпись)

Ставрополь 2023

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

Ход работы:

1. Создал репозиторий на GitHub:
https://github.com/IsSveshuD/OOP_Lab_4.1.git .
2. Проработал пример:



```
C:\Users\user\AppData\Local\Programs\Py
3/4
Введите обыкновенную дробь: 4/3
4/3
25/12
7/12
1/1
16/9

Process finished with exit code 0
|
```

Рисунок 1 – Пример

3. Выполнил индивидуальное задание 1.

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

class Equation:
    def __init__(self, a, b):
        self.first = a
        self.second = b

    def read(self):
        self.first = float(input("Введите значение коэффициента A: "))
        self.second = float(input("Введите значение коэффициента B: "))

    def display(self):
        print(f"Уравнение y = {self.first}x + {self.second}")

    def function(self, x):
        return self.first * x + self.second

def make_equation(a, b):
    return Equation(a, b)

if __name__ == "__main__":
    equation = make_equation(2, 3)
    equation.read()
    equation.display()
    x = float(input("Введите значение x: "))
    result = equation.function(x)
    print(f"Значение функции y для x = {x} равно {result}")
```

Рисунок 2 – Индивидуальное задание1

4. Получил следующий результат работы индивидуального задания

1.

```
C:\Users\user\AppData\Local\Programs\Python\Pyt
Введите значение коэффициента A: 3
Введите значение коэффициента B: 5
Уравнение  $y = 3.0x + 5.0$ 
Введите значение x: 10
Значение функции y для x = 10.0 равно 35.0

Process finished with exit code 0
|
```

Рисунок 3 – Результат работы задания 1

5. Выполнил индивидуальное задание 1.

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

class Money:
    def __init__(self, denominations=None):
        if denominations is None:
            self.denominations = {
                5000: 0,
                1000: 0,
                500: 0,
                100: 0,
                50: 0,
                10: 0,
                5: 0,
                2: 0,
                1: 0,
                0.5: 0,
                0.1: 0,
                0.05: 0,
                0.01: 0
            }
        else:
            self.denominations = denominations

    def read(self):
        print("Введите количество купюр/монет каждого номинала:")
        for denom in self.denominations.keys():
            self.denominations[denom] = int(input(f"{denom} рублей/копеек: "))
```

```

def display(self):
    total = sum(denom * count for denom, count
                in self.denominations.items())
    print(f"Общая сумма: {total:.2f} рублей".replace('.', ','))

def add(self, other):
    result = Money()
    for denom in self.denominations.keys():
        result.denominations[denom] = self.denominations[denom] + \
            other.denominations[denom]
    return result

def sub(self, other):
    result = Money()
    for denom in self.denominations.keys():
        result.denominations[denom] = self.denominations[denom] - \
            other.denominations[denom]
    return result

def div(self, divisor):
    result = Money()
    for denom in self.denominations.keys():
        result.denominations[denom] = self.denominations[denom] / divisor
    return result

def mul(self, multiplier):
    result = Money()
    for denom in self.denominations.keys():
        result.denominations[denom] = self.denominations[denom] \
            * multiplier

def equals(self, other):
    if isinstance(other, Money):
        a = sum(denom * count for denom, count
                in self.denominations.items())
        b = sum(denom * count for denom, count
                in other.denominations.items())

        return a == b
    else:
        return False

def less(self, other):
    if isinstance(other, Money):
        a = sum(denom * count for denom, count
                in self.denominations.items())
        b = sum(denom * count for denom, count
                in other.denominations.items())

        return a < b
    else:
        return False

def greater(self, other):
    if isinstance(other, Money):
        a = sum(denom * count for denom, count in
                self.denominations.items())
        b = sum(denom * count for denom, count in
                other.denominations.items())

```

```

        return a > b
    else:
        return False

if __name__ == '__main__':
    money1 = Money()
    money1.read()
    money1.display()

    money2 = Money()
    money2.read()
    money2.display()

    result = money1.add(money2)
    print("\nСумма:")
    result.display()

    result = money1.sub(money2)
    print("\nРазница:")
    result.display()

    divisor = float(
        input("\nВведите число, на которое хотите разделить сумму: "))
    result = money1.div(divisor)
    print("\nРезультат деления:")
    result.display()

    multiplier = float(
        input("\nВведите число, на которое хотите умножить сумму: "))
    result = money1.mul(multiplier)
    print("\nРезультат умножения:")
    result.display()

```

Рисунок 4 – Индивидуальное задание 2

6. Получил следующий результат работы индивидуального задания

2:

```
C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\user\Do
Введите количество купюр/монет каждого номинала:
5000 рублей/копеек: 5
1000 рублей/копеек: 5
500 рублей/копеек: 5
100 рублей/копеек: 5
50 рублей/копеек: 5
10 рублей/копеек: 5
5 рублей/копеек: 5
2 рублей/копеек: 5
1 рублей/копеек: 5
0.5 рублей/копеек: 5
0.1 рублей/копеек: 5
0.05 рублей/копеек: 5
0.01 рублей/копеек: 5
Общая сумма: 33343,30 рублей
Введите количество купюр/монет каждого номинала:
5000 рублей/копеек: 5
1000 рублей/копеек: 4
500 рублей/копеек: 3
100 рублей/копеек: 2
50 рублей/копеек: 1
10 рублей/копеек: 3
5 рублей/копеек: 3
2 рублей/копеек: 3
1 рублей/копеек: 3
0.5 рублей/копеек: 3
0.1 рублей/копеек: 3
0.05 рублей/копеек: 3
0.01 рублей/копеек: 3
Общая сумма: 30805,98 рублей

Сумма:
Общая сумма: 64149,28 рублей

Разница:
Общая сумма: 2537,32 рублей

Введите число, на которое хотите разделить сумму: 2

Результат деления:
Общая сумма: 16671,65 рублей

Введите число, на которое хотите умножить сумму: 4

Результат умножения:
Общая сумма: 133373,20 рублей

Process finished with exit code 0
```

Рисунок 5 – Результат работы задания 2.

Ответы на вопросы:

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова `class` и имени класса:

```
# class syntax
```

```
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 экземпляра определяются в методах и хранят информацию, специфичную для экземпляра.

3. Каково назначение методов класса?

Методы определяют функциональность объектов, принадлежащих конкретному классу.

Основной синтаксис выглядит так:

```
# basic method syntax
```

```
class MyClass:
```

```
    # the constructor
```

```
    def __init__(self, arg1):
```

```
        self.att = arg1
```

```
    # custom method
```

```
    def do_smt(self):
```

```
        # does something
```

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.

5. Каково назначение `self` ?

Аргумент `self` представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам.

6. Как добавить атрибуты в класс?

Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров этого класса.

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

Отсутствие модификатора (`public`): Атрибуты и методы без явного указания модификатора доступа считаются открытыми (`public`) и могут быть доступны из любого места программы.

Защищенный (`protected`): Атрибуты и методы, которые начинаются с одного подчеркивания (например, `_protected_attribute`), считаются защищенными. Однако, это больше соглашение между программистами, чем строгий модификатор доступа.

Приватный (`private`): Атрибуты и методы, которые начинаются с двух подчеркиваний (например, `__private_attribute`), считаются приватными. Однако, также существует механизм "name mangling" (переименование), который делает их несколько менее прямыми для доступа.

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`

Вывод: в результате выполнения лабораторной работы были получены практические знания и теоретические сведения о методах работы с матрицами и векторами с помощью библиотеки NumPy языка программирования Python.