

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Кафедра инфокоммуникаций**

**Отчет  
по лабораторной работе №8  
«Элементы объектно-ориентированного  
программирования в языке Python»  
по дисциплине:  
«Введение в системы искусственного интеллекта»**

**Вариант 3**

Выполнил: студент группы ИВТ-б-о-18-1  
Данченко Максим Игоревич

\_\_\_\_\_ (подпись)

Проверил:  
Воронкин Роман Александрович

\_\_\_\_\_ (подпись)

Ставрополь, 2022 г.

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

### Задание №1

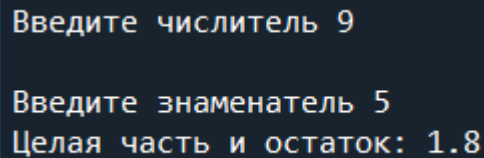
**3. Поле first — целое положительное число, числитель; поле second — целое положительное число, знаменатель. Реализовать метод ipart() — выделение целой части дроби first/second. Метод должен проверять неравенство знаменателя нулю.**

Был создан класс Neravenstvo с полями first(числитель) и second(знаменатель), для доступа к переменным были реализованы get и set методы. Для выделение целой части дроби, был создан специальный метод def ipart () (рисунок 1)

```
1  class Neravenstvo:
2
3      def __init__(self, first = 0, second = 0):
4          self.__first = int(first)
5          self.__second = int(second)
6
7      def get_first(self):
8          return self.__first
9
10     def set_first(self, f):
11         self.__first = f
12
13     def get_second(self):
14         return self.__second
15
16     def set_second(self, s):
17         self.__second = s
18
19     def ipart(self):
20         return self.__first / self.__second
21
22     def read(self):
23         self.set_first(int(input("Введите числитель ")))
24         self.set_second(int(input("Введите знаменатель ")))
25
26     def display(self):
27         print(f'Целая часть и остаток: {self.ipart()}')
28
29 if __name__ == '__main__':
30     neravenstvo = Neravenstvo()
31     neravenstvo.read()
32     neravenstvo.display()
```

Рисунок 1 – Листинг программы

Результат работы программы изображен на рисунке 2



```
Введите числитель 9
Введите знаменатель 5
Целая часть и остаток: 1.8
```

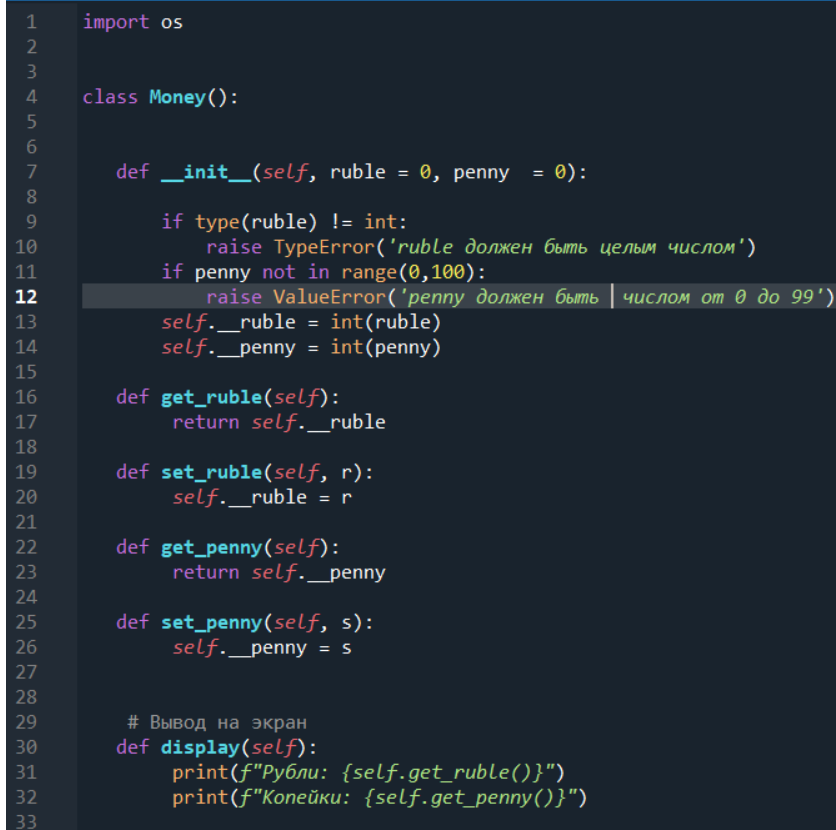
Рисунок 2 – Результат программы

## Задание №2

Создать класс Money для работы с денежными суммами.

Число должно быть представлено двумя полями: типа int для рублей и копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения.

Код программы показан на рисунках 1-4



```
1  import os
2
3
4  class Money():
5
6
7      def __init__(self, ruble = 0, penny = 0):
8
9          if type(ruble) != int:
10             raise TypeError('ruble должен быть целым числом')
11          if penny not in range(0,100):
12             raise ValueError('penny должен быть | числом от 0 до 99')
13          self.__ruble = int(ruble)
14          self.__penny = int(penny)
15
16      def get_ruble(self):
17          return self.__ruble
18
19      def set_ruble(self, r):
20          self.__ruble = r
21
22      def get_penny(self):
23          return self.__penny
24
25      def set_penny(self, s):
26          self.__penny = s
27
28
29      # Вывод на экран
30      def display(self):
31          print(f"Рубли: {self.get_ruble()}")
32          print(f"Копейки: {self.get_penny()}")
33
```

Рисунок 1 – листинг программы

```

34
35     def vvodruble(self):
36         self.set_ruble(int(input(f"Введите рубль:")))
37         self.set_penny(int(input(f"Введите копейки:")))
38
39
40
41
42     def __str__(self):
43         return f'{self.get_ruble()} {self.get_penny()}'
44
45     def __add__(self, other):
46         kopSum = self.get_penny() + other.get_penny()
47         rub = self.get_ruble() + other.get_ruble() + (kopSum // 100)
48         kop = (kopSum % 100)
49         return Money(rub, kop)
50
51     def __sub__(self, other):
52         kopSum = self.get_penny() - other.get_penny()
53         rub = self.get_ruble() - other.get_ruble() + (kopSum // 100)
54         kop = (kopSum % 100)
55         return Money(rub, kop)
56
57     def __mul__(self, other):
58         kopSum = self.get_penny() * other.get_penny()
59         rub = self.get_ruble() * other.get_ruble() + (kopSum // 100)
60         kop = (kopSum % 100)
61         return Money(rub, kop)
62
63     def __truediv__(self, other):
64         kopSum = self.get_penny() / other.get_penny()
65         rub = self.get_ruble() / other.get_ruble() + (kopSum // 100)
66         kop = (kopSum % 100)
67         return Money(rub, kop)
68

```

Рисунок 2 – листинг программы

```

68
69     def sravnenie(self, a, b):
70         if a > b :
71             return 1
72         elif a == b :
73             return 2
74         else:
75             return 3
76
77     # Ввод данных
78
79     if __name__ == '__main__':
80         acc = Money()
81         acc1=Money()
82         acc2=Money()
83         while True:
84             os.system('cls')
85             acc.display()
86
87             print("Ввод>> [1]")
88             print("сумма>> [2]")
89             print("вычитание>> [3]")
90             print("произведение>> [4]")
91             print("разность>> [5]")
92             print("сравнение>> [6]")
93             print("Выход >> [7]")
94
95             command = int(input(">>"))
96
97             if command == 1:
98
99                 print("Ввод первого числа ")
100                 acc1.vvodruble()
101                 print("Ввод второго числа ")
102                 acc2.vvodruble()
103

```

Рисунок 3 – листинг программы

```

91     print("разность>> [5]")
92     print("сравнение>> [6]")
93     print("Выход >> [7]")
94
95     command = int(input(">>"))
96
97     if command == 1:
98
99         print("Ввод первого числа ")
100        acc1.vvodruble()
101        print("Ввод второго числа ")
102        acc2.vvodruble()
103
104
105    elif command == 2:
106        acc=acc1+acc2
107        print(acc)
108
109    elif command == 3:
110        acc=acc1-acc2
111        print(acc)
112
113    elif command == 4:
114        acc=acc1*acc2
115        print(acc)
116
117    elif command == 5:
118        acc=acc1/acc2
119        print(acc)
120
121    elif command == 6:
122        # print (acc.sravnenie(acc1.get_ruble()+acc1.get_penny(),acc2.get_ruble()+acc2.get_penny()))
123        if(acc.sravnenie(acc1.get_ruble()+acc1.get_penny(),acc2.get_ruble()+acc2.get_penny())) == 1:
124            print(f"{acc1.get_ruble()} руб, {acc1.get_penny()} коп больше {acc2.get_ruble()} руб, {acc2.get_penny()} коп" )
125        elif(acc.sravnenie(acc1.get_ruble()+acc1.get_penny(),acc2.get_ruble()+acc2.get_penny())) == 2:
126            print(f"{acc1.get_ruble()} руб, {acc1.get_penny()} коп равны {acc2.get_ruble()} руб, {acc2.get_penny()} коп" )
127        elif(acc.sravnenie(acc1.get_ruble()+acc1.get_penny(),acc2.get_ruble()+acc2.get_penny())) == 3:
128            print(f"{acc1.get_ruble()} руб, {acc1.get_penny()} коп меньше {acc2.get_ruble()} руб, {acc2.get_penny()} коп" )
129
130    elif command == 7:
131        break
132    else:
133        print(f"Неизвестная команда: {command}\n")
134        input("Нажмите 'Enter' для продолжения")

```

Рисунок 4 – листинг программы

Результат работы программы изображен на рисунках 5-10

```
Рубли: 0
Копейки: 0
ввод>> [1]
сумма>> [2]
вычитание>> [3]
произведение>> [4]
разность>> [5]
сравнение>> [6]
Выход >> [7]

>>1
ввод первого числа

введите рубли:2

введите копейки:20
ввод второго числа

введите рубли:4
```

Рисунок 5 – Результат выполнения ввода чисел

```
>>2
6,50
Рубли: 6
Копейки: 50
ввод>> [1]
сумма>> [2]
вычитание>> [3]
произведение>> [4]
разность>> [5]
сравнение>> [6]
Выход >> [7]
```

Рисунок 6 – Результат выполнения суммирования

```
>>3
-3,90
Рубли: -3
Копейки: 90
ввод>> [1]
сумма>> [2]
вычитание>> [3]
произведение>> [4]
разность>> [5]
сравнение>> [6]
Выход >> [7]

>>|
```

Рисунок 7 – Результат выполнения вычитания

```
>>4
14,0
Рубли: 14
Копейки: 0
ввод>> [1]
сумма>> [2]
вычитание>> [3]
произведение>> [4]
разность>> [5]
сравнение>> [6]
Выход >> [7]
```

Рисунок 8 – Результат выполнения произведения

```
>>6
2 руб, 20 коп меньше 3 руб, 20 коп
Рубли: 0
Копейки: 0
ввод>> [1]
сумма>> [2]
вычитание>> [3]
произведение>> [4]
разность>> [5]
сравнение>> [6]
Выход >> [7]
```

Рисунок 9 – Результат выполнения сравнения в первом случае

```
>>6
2 руб, 20 коп равны 2 руб, 20 коп
Рубли: 0
Копейки: 0
ввод>> [1]
сумма>> [2]
вычитание>> [3]
произведение>> [4]
разность>> [5]
сравнение>> [6]
Выход >> [7]
```

Рисунок 10 – Результат выполнения сравнения во втором случае

```
>>6
4 руб, 20 коп больше 3 руб, 20 коп
Рубли: 0
Копейки: 0
ввод>> [1]
сумма>> [2]
вычитание>> [3]
произведение>> [4]
разность>> [5]
сравнение>> [6]
Выход >> [7]
```

Рисунок 11 – Результат выполнения сравнения в третьем случае

**Вывод:** в процессе выполнения лабораторной работы, были приобретены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.

### 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?

Атрибуты класса являются общими для всех объектов класса, а атрибуты экземпляра специфическими для каждого экземпляра. Более того, атрибуты класса определяются внутри класса, но вне каких-либо методов, а атрибуты экземпляра обычно определяются в методах, чаще всего в `__init__`.

#### 3. Каково назначение методов класса?



Методы определяют функциональность объектов, принадлежащих конкретному классу.

#### **4. Для чего предназначен метод `__init__()` класса?**

Метод `__init__` является конструктором. Конструкторы - это концепция объектноориентированного программирования. Класс может иметь один и только один конструктор. Если `__init__` определен внутри класса, он автоматически вызывается при создании нового экземпляра класса.

#### **5. Каково назначение `self` ?**

Аргумент `self` представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам. Важно использовать параметр `self` внутри метода, если мы хотим сохранить значения экземпляра для последующего использования.

В большинстве случаев нам также необходимо использовать параметр `self` в других методах, потому что при вызове метода первым аргументом, который ему передается, является сам объект. Давайте добавим метод к нашему классу `River` и посмотрим, как он будет работать.

#### **6. Как добавить атрибуты в класс?**

Атрибуты созданного экземпляра класса можно добавлять, изменять или удалять в любое время, используя для доступа к ним точечную запись. Если построить инструкцию, в которой присвоить значение атрибуту, то можно изменить значение, содержащееся внутри существующего атрибута, либо создать новый с указанным именем и содержащий присвоенное значение:

```
имя-экземпляра.имя-атрибута = значение  
del имя-экземпляра.имя-атрибута
```

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

Если вы знакомы с языками программирования Java, C#, C++ то, наверное, уже задались вопросом: “а как управлять уровнем доступа?”. В перечисленных языках вы можете явно указать для переменной, что доступ к ней снаружи класса запрещен, это делается с помощью ключевых слов (private, protected и т.д.). В Python таких возможностей нет, и любой может обратиться к атрибутам и методам вашего класса, если возникнет такая необходимость. Это существенный недостаток этого языка, т.к. нарушается один из ключевых принципов ООП – инкапсуляция. Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются getter/setter, их можно реализовать, но ничего не мешает изменить атрибут напрямую. При этом есть соглашение, что метод или атрибут, который начинается с нижнего подчеркивания, является скрытым, и снаружи класса трогать его не нужно (хотя сделать это можно).

## **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.