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 11

Выполнил: студент группы ИВТ-б-о-18-1
Солдатенко Евгений Михайлович
_____(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

Линейное уравнение y=Ax+B. Поле first — дробное число, коэффициент A; поле second — дробное число, коэффициент B. Реализовать метод root() — вычисление корня линейного уравнения. Метод должен проверять неравенство коэффициента B нулю.

Был создан класс Line с полями firs(A) и second(B), для доступа к переменным были реализованы get и set методы. (рисунок 1)

Ввод [1]:
```
#Линейное уравнение y=Ax+B. Поле first – дробное число, коэффициент A;
#поле second – дробное число, коэффициент B.
#Реализовать метод root() – вычисление корня линейного уравнения.
#Метод должен проверять неравенство коэффициента B нулю.

class Line():

    def __init__(self, first = 0, second = 0):
        self.__first = int(first)
        self.__second = int(second)

    def get_first(self):
        return self.__first

    def set_first(self, A):
        self.__first = A

    def get_second(self):
        return self.__second

    def set_second(self, B):
        self.__second = B

    def root(self):
        return -(self.__second) / self.__first

    #ввод с клавиатуры
    def read(self):
        self.set_first(int(input("Введите A: ")))
        self.set_second(int(input("Введите B: ")))

    #вывод на экран
    def display(self):
        print(f'Результат: {self.root()}')

if __name__ == '__main__':
    time = Line()
    time.read()
    time.display()
```

Рисунок 1 – Листинг программы

Результат работы программы изображен на рисунке 2

```
Введите A: 10
Введите B: -10
Результат: 1.0
```

Рисунок 2 – Результат программы

Задание №2

Реализовать класс Bankomat, моделирующий работу банкомата. В классе должны содержаться поля для хранения идентификационного номера банкомата, информации о текущей сумме денег, оставшейся в банкомате, минимальной и максимальной суммах, которые позволяется снять клиенту в один день. Сумма денег представляется полями- номиналами 10-1000 (см. задание 13). Реализовать метод инициализации банкомата, метод загрузки купюр в банкомат и метод снятия определенной суммы денег. Метод снятия денег должен выполнять проверку на корректность снимаемой суммы: она не должна быть меньше минимального значения и не должна превышать максимальное значение. Метод toString()должен преобразовать в строку сумму денег, оставшуюся в банкомате.

```python
#Реализовать класс Bankomat, моделирующий работу банкомата. В классе должны содержаться поля для хранения идентификационного номера банкомата
#информации о текущей сумме денег, оставшейся в банкомате, минимальной и максимальной суммах, которые позволяется снять клиенту в один день.
#Сумма денег представляется полями- номиналами 10-1000 (см. задание 13). Реализовать метод инициализации банкомата,
#метод загрузки купюр в банкомат и метод снятия определенной суммы денег. Метод снятия денег должен выполнять проверку на корректность
#снимаемой суммы: она не должна быть меньше минимального значения и не должна превышать максимальное значение.
# Метод toString()должен преобразовать в строку сумму денег, оставшуюся в банкомате.


import os
import Num2Text

class Bankomat():

    max = 10000
    min = 10

    def __init__(self, Idx = 1, Ostatok = 4578, Min_sum = 10, Max_sum = 10000):
        self.__Idx = Idx         # Id
        self.__Ostatok = Ostatok    #текущая сумма остатка в банкомате
        self.__Min_sum = Min_sum          # минимальная сумма которую можно снять в 1 день
        self.__Max_sum = Max_sum   # максимальная сумма которую можно снять в 1 день
```

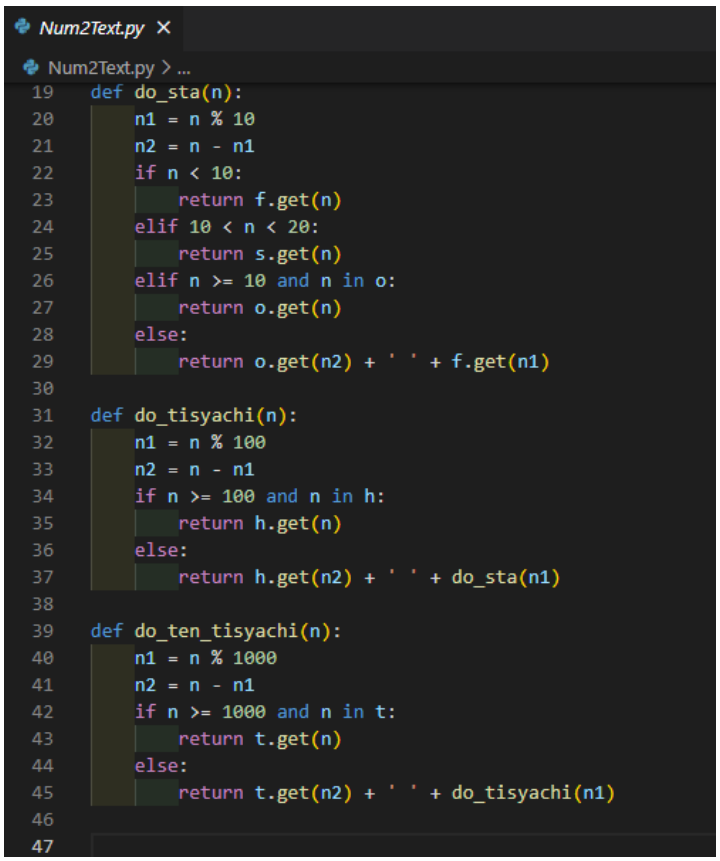
Рисунок 3.1 – Поля класса

```python
def get_name(self):
    return self.__Idx

def set_name(self, n):
    self.__Idx = n

def get_sumostat(self):
    return self.__Ostatok

def set_sumostat(self, ostat):
    self.__Ostatok = ostat

def get_minisum(self):
    return self.__Min_sum

def set_minisum(self, mini):
    self.__Min_sum = mini

def get_maxisum(self):
    return self.__Max_sum

def set_maxisum(self, maxi):
    self.__Max_sum = maxi
```

Рисунок 3.2 – Get и Set методы

```python
# Снятие денег со счета
def take_money(self):
    print("Сумма на счету:", self.get_sumostat())
    amount_entered = int(input("Введите сумму денег, которые хотите снять:"))
    if amount_entered >= self.get_minisum() and amount_entered <= self.get_maxisum():
        if amount_entered > self.get_sumostat():
            print(f"Вы не можете вывести сумму больше вашего счета: {self.get_sumostat()}")
            self.take_money()
        else:
            self.set_sumostat(self.get_sumostat() - amount_entered)
            print("Отстаток счета:", self.get_sumostat())
    else:
        print(f"Вы не можете вывести сумму больше 10000 и меньше 10")

# Зачисление денег на счет
def give_money(self):
    print("Сумма на счету:", self.get_sumostat())
    amount_entered = int(input("Введите сумму денег, которые хотите зачислить:"))
    self.set_sumostat(self.get_sumostat() + amount_entered)
    print("Сумма на счету:", self.get_sumostat())
```

Рисунок 3.3 – Методы класса



```
Num2Text.py ×
Num2Text.py > ...
19  def do_sta(n):
20      n1 = n % 10
21      n2 = n - n1
22      if n < 10:
23          return f.get(n)
24      elif 10 < n < 20:
25          return s.get(n)
26      elif n >= 10 and n in o:
27          return o.get(n)
28      else:
29          return o.get(n2) + ' ' + f.get(n1)
30
31  def do_tisyachi(n):
32      n1 = n % 100
33      n2 = n - n1
34      if n >= 100 and n in h:
35          return h.get(n)
36      else:
37          return h.get(n2) + ' ' + do_sta(n1)
38
39  def do_ten_tisyachi(n):
40      n1 = n % 1000
41      n2 = n - n1
42      if n >= 1000 and n in t:
43          return t.get(n)
44      else:
45          return t.get(n2) + ' ' + do_tisyachi(n1)
46
47
```

Рисунок 3.4 – Листинг модуля Num2text

```
# Получение суммы прописью
def sum_to_text(self):

    # 1 - 99
    if 0 < self.get_sumostat() < 100:
        print(Num2Text.do_sta(self.get_sumostat()), "рублей")
    # 100 - 999
    elif 99 < self.get_sumostat() < 1000:
        print(Num2Text.do_tisyachi(self.get_sumostat()), "рублей")
    # 1000 - 9999
    elif 999 < self.get_sumostat() < 10000:
        print(Num2Text.do_ten_tisyachi(self.get_sumostat()), "рублей")
    # 10000 - 99999
    #elif self.get_maxisum() < 100000:
```

Рисунок 3.5 – Листинг метода sum_to_text

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 языка 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 помешает 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.