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 9

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Полещук Константин Сергеевич
_____(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

Поле first — целое положительное число, часы; поле second — целое положительное число, минуты. Реализовать метод minutes() — приведение времени в минуты.

Был создан класс Time с полями firs(часы) и second(минуты), для доступа к переменным были реализованы get и set методы. Для преобразования часов в минуты, был создан специальный метод def minutes() (рисунок 1)

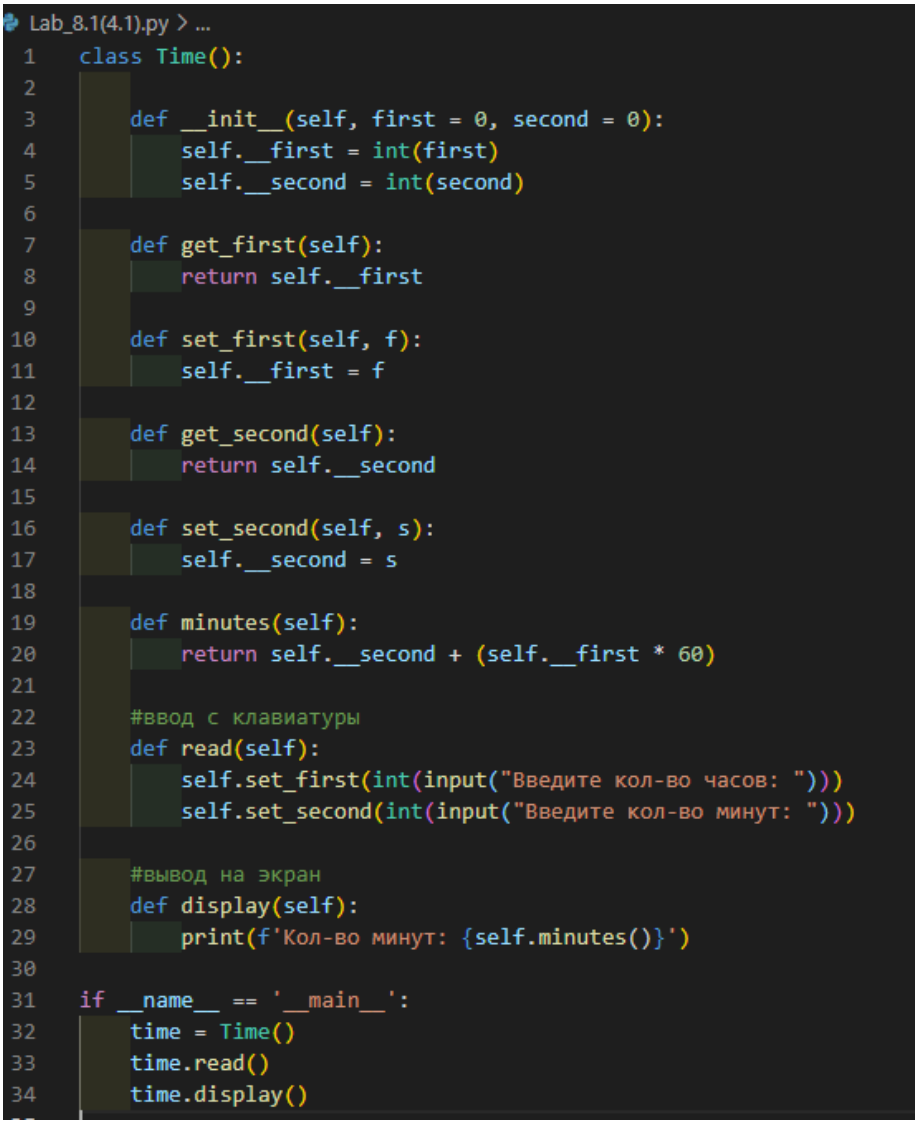

Рисунок 1 – Листинг программы

Результат работы программы изображен на рисунке 2

```
Введите кол-во часов: 4
Введите кол-во минут: 52
Кол-во минут: 292
PS C:\Users\termo\YandexDisk\шарага\4 курс\8 семестр\Python>
```

Рисунок 2 – Результат программы

Задание №2

Реализовать класс Account, представляющий собой банковский счет. В классе должны быть четыре поля: фамилия владельца, номер счета, процент начисления и сумма в рублях.

Открытие нового счета выполняется операцией инициализации. Необходимо выполнять следующие операции: сменить владельца счета, снять некоторую сумму денег со счета, положить деньги на счет, начислить проценты, перевести сумму в доллары, перевести сумму в евро, получить сумму прописью (преобразовать в числительное).

Для решение данной задачи, был создан класс Account с двумя атрибутами класса (usd, eur), в которые были записаны курсы валют и тремя атрибутами экземпляра класса, определенных в методе __init__ (SecondName, AccountNumber, PrecAcc и AmountRub), в которые по умолчанию были переданы начальные данные (рисунок 3.1)

```
class Account():

    usd = 112 # Курс доллара
    eur = 122 # Курс евро

    def __init__(self, SecondName = "Иванов", AccountNumber = 4578, PrecAcc = 10.2, AmountRub
        self.__SecondName = SecondName          # Фамилия владельца
        self.__AccountNumber = AccountNumber    # Номер счета
        self.__PrecAccrual = PrecAcc            # Процент начисления
        self.__AmountRub = float(AmountRub)     # Сумма в рублях
```

Рисунок 3.1 – Поля класса

Также были добавленные методы доступа к полям – get и set (рисунок 3.2)

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

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

def get_accnumber(self):
    return self.__AccountNumber

def set_accnumber(self, number):
    self.__AccountNumber = number

def get_prec(self):
    return self.__PrecAccrual

def set_prec(self, prec):
    self.__PrecAccrual = prec

def get_rub(self):
    return self.__AmountRub

def set_rub(self, rub):
    self.__AmountRub = rub
```

Рисунок 3.2 – Get И Set методы

Были созданы методы с работой данных пользователя, ввод, вывод информации, снятия денег и перевода в другую валюту (рисунок 3.3)

```python
# Смена владельца счета
def change_name(self):
    self.set_name(input("Введите фамилию нового владельца: "))

# Снятие денег со счета
def take_money(self):
    print("Сумма на счету:", self.get_rub())
    amount_entered = float(input("Введите сумму денег, которые хотите снять:"))
    if amount_entered > self.get_rub():
        print(f"Вы не можете ввести сумму больше вашего счета: {self.get_rub()}")
        self.take_money()
    else:
        self.set_rub(self.get_rub() - amount_entered)
        print("Отстаток счета:", self.get_rub())

# Зачисление денег на счет
def give_money(self):
    print("Сумма на счету:", self.get_rub())
    amount_entered = float(input("Введите сумму денег, которые хотите зачислить:"))
    self.set_rub(self.get_rub() + amount_entered)
    print("Сумма на счету:", self.get_rub())

# Начисление процентов
def add_perecent(self):
    print("Процент начисления:", self.get_prec())
    add_prec = float(input("Введите кол-во процентов, которые хотите начислить:"))
    self.set_prec(self.get_prec() + add_prec)
    print("Начисленные проценты:", self.get_prec())

# Перевод суммы в доллары
def conver_usd(self):
    print(f"Ваш счет в рублях: {self.get_rub()}")
    print(f"Ваш счет в долларах: {self.get_rub() / self.usd}")

# Перевод суммы в евро
def conver_eur(self):
    print(f"Ваш счет в рублях: {self.get_rub()}")
    print(f"Ваш счет в евро: {self.get_rub() / self.eur}")
```

Рисунок 3.3 – Методы класса

Для удобства, метод получения суммы прописью был реализован с помощью созданного модуля Num2Text, в котором были написаны 3 функции с перевод числа в прописной вариант (рисунок 3.4)

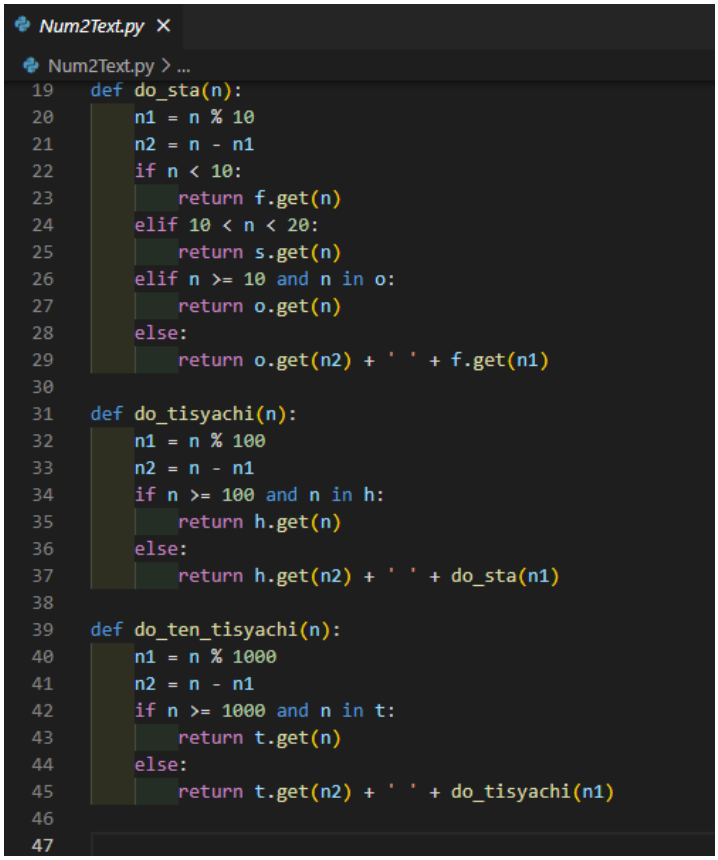

Рисунок 3.4 – Листинг модуля Num2text

Для работы с модулем, метод класса – sum_to_text(), вызывает функции модуля и передает им числа, после чего получает перевод прописью и выводит их на экран пользователя (рисунок 3.5)

Рисунок 3.5 – Листинг метода sum_to_text

Результат работы программы изображен на рисунках 4.1-4.4

```
Владелец счета: Иванов
Номер счета: 4578
Сумма счета: 5848.0 руб.
Процент начисления: 10.2 %

Сменить владельца счета >> [1]
Снять сумму денег со счета >> [2]
Пополнить счет>> [3]
Начислить проценты >> [4]
Перевести сумму в доллары >> [5]
Перевести сумму в евро >> [6]
Получить сумму прописью (до 9999) >> [7]
Выход >> [8]
>>
```

Рисунок 4.1 – Результат выполнения

```
Владелец счета: Полещук
Номер счета: 4578
Сумма счета: 5848.0 руб.
Процент начисления: 10.2 %

Сменить владельца счета >> [1]
Снять сумму денег со счета >> [2]
Пополнить счет>> [3]
Начислить проценты >> [4]
Перевести сумму в доллары >> [5]
Перевести сумму в евро >> [6]
Получить сумму прописью (до 9999) >> [7]
Выход >> [8]
>>1
Введите фамилию нового владельца: Полещук
```

Рисунок 4.1 – Результат выполнения

```
Владелец счета: Полещук
Номер счета: 4578
Сумма счета: 5848.0 руб.
Процент начисления: 10.2 %

Сменить владельца счета >> [1]
Снять сумму денег со счета >> [2]
Пополнить счет>> [3]
Начислить проценты >> [4]
Перевести сумму в доллары >> [5]
Перевести сумму в евро >> [6]
Получить сумму прописью (до 9999) >> [7]
Выход >> [8]
>>6
Ваш счет в рублях: 5848.0
Ваш счет в евро: 47.9344262295082
```

Рисунок 4.1 – Результат выполнения

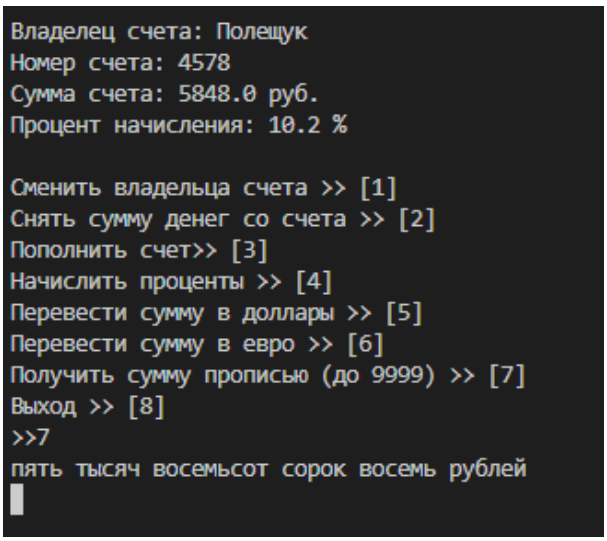

Рисунок 4.1 – Результат выполнения

Файл Lab_7(2.13).py и Data.py с решением задачи, находятся на **Github**: https://github.com/Scratchykaktus/Python.git

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.