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 8

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Михайличенко Руслан 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

8. Поле first — целое число, левая граница диапазона, включается в диапазон; поле second — целое число, правая граница диапазона, не включается в диапазон. Пара чисел представляет полуоткрытый интервал [first, second). Реализовать метод `rangecheck()` — проверку заданного целого числа на принадлежность диапазону.

```python
class Task:

    def __init__(self, first=0, second=0, number=1):
        self.first = int(first)
        self.second = int(second)
        self.number = int(number)

    def read(self, prompt=None):
            print("\nВведите диапазон:\n")
            self.first = int(input("Введите левую границу диапазона: "))
            self.second = int(input("Введите правую границу диапазона: "))

            print(self.first, "...",self.second)

            line = int(input() if prompt is None else input(prompt))
            self.number = line


            if line == 0:
             raise ValueError()


    def rangecheck(self):
        if(self.first<self.number and self.number<self.second):
            print(f"Число {self.number} входит в диапазон {self.first}...{self.second}" )
        else:
            print(f"Число {self.number} не входит в диапазон {self.first}...{self.second}" )

    def display(self):
        print(f"{self.first}...{self.second}")

if __name__ == '__main__':

    r1 = Task()
    r1.display()
    r2 = Task()
    r2.read("Введите любое число: ")
    r2.display()
    r1.rangecheck()
    r2.rangecheck()
```

Рисунок 1 – Листинг программы

Результат работы программы изображен на рисунке 2

```
Введите левую границу диапазона: 1
Введите правую границу диапазона: 6
1 ... 6
Введите любое число: 2
1...6
Число 1 не входит в диапазон 0...0
Число 2 входит в диапазон 1...6
```

Рисунок 2 – Результат программы

Задание №2

8. Создать класс Time для работы со временем в формате «час:минута:секунда». Класс должен включать в себя не менее четырех функций инициализации: числами, строкой (например, «23:59:59»), секундами и временем. Обязательными операциями являются: вычисление разницы между двумя моментами времени в секундах, сложение времени и заданного количества секунд, вычитание из времени заданного количества секунд, сравнение моментов времени, перевод в секунды, перевод в минуты (с округлением до целой минуты).

```python
class time:

    def __init__(self, hour=0, minute=0, second=0):
        hour = int(hour)
        minute = int(minute)
        second = int(second)
```

Рисунок 3.1 – Листинг

```
def TimeToSec(self):
    self.hour= int(input('Введите часы: '))
    self.minute= int(input('Введите минуты: '))
    self.second= int(input('Введите секунды: '))

    res = (self.second + (self.minute * 60) + (self.hour * 3600))

    print(f'Перевод в секунды: {res}')

def RazSec(self):
    print('Первый момент вермени')
    self.hour= int(input('Введите часы: '))
    self.minute= int(input('Введите минуты: '))
    self.second= int(input('Введите секунды: '))

    print('Второй момент вермени')
    self.hour1= int(input('Введите часы: '))
    self.minute1= int(input('Введите минуты: '))
    self.second1= int(input('Введите секунды: '))

    sec1 = (self.second + (self.minute * 60) + (self.hour * 3600))
    sec2 = (self.second1 + (self.minute1 * 60) + (self.hour1 * 3600))

    if (sec1 >= sec2):
        sec1 -= sec2
    else:
      sec1 = sec2 - sec1
    print(f'Разница времени в секундах {sec1}')

def PerMin(self):
    self.second= int(input('Введите секунды: '))

    min = round(self.second/60)

    print(f'Перевод в минуты: {min}')
```

Рисунок 3.2 – Листинг

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

```
def Srav(self):
    print()
    time = input("Введите первый момент времени \"час.минута.секунда\": ")
    d = time.split('.', maxsplit=2)
    time1 = datetime.timedelta(hours=int(d[0]),minutes=int(d[1]),seconds=int(d[2]))

    time = input("Введите вторую дату в формате \"час.минута.секунда\": ")
    d = time.split('.', maxsplit=2)
    time2 = datetime.timedelta(hours=int(d[0]),minutes=int(d[1]),seconds=int(d[2]))

    if(time1>time2):
        print(f"Время {time1} находится после времени {time2}")
        #self.cnumber(date1,date2)
    elif(time1<time2):
        print(f"Вермя {time1} находится до времени {time2}")
        time1,time2 = time2,time1
        #self.cnumber(date2,date1)
    else:
        print(f"Время {time1} равна времени {time2}")
        #self.cnumber(date1,date2)

def Vich(self):
    print("Инициализация строкой ввода \"час.минута.секунда\":")
    time = input("Введите дату в формате \"час.минута.секунда\": ")

    d = time.split('.', maxsplit=2)

    self.hour = int(d[0])
    self.minute = int(d[1])
    self.second = int(d[2])

    time = datetime.timedelta(hours=self.hour,minutes=self.minute,seconds=self.second)
    num = int(input("Введите целое число: "))
    time -= datetime.timedelta(seconds=num)

    print(f"Полсе вычитания {num} секунд время станет равной: {time} \n")
```

Рисунок 3.3 – Листинг

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)

```python
if __name__ == '__main__':
    r1 = time()
    while True:
        os.system('cls')


        print("\nПеревести время в секунды >> [1]")
        print("Вычисление разницы >> [2]")
        print("Перевод секунд в минуты >> [3]")
        print("Сравнение двух времен >> [4]")
        print("Вычитание >> [5]")
        print("Сложение >> [6]")
        print("Выход >> [7]")

        command = int(input(">>"))

        if command == 1:
            r1.TimeToSec()
            input()
        elif command == 2:
            r1.RazSec()
            input()
        elif command == 3:
            r1.PerMin()
            input()
        elif command == 4:
            r1.Srav()
            input()
        elif command == 5:
            r1.Vich()
            input()
        elif command == 6:
            r1.Sloj()
            input()
        elif command == 7:
            break
        else:
            print(f"Неизветсная команда: {command}\n")
            input("Нажмите 'Enter' для продолжения")
```

Рисунок 3.4 – Листинг

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)

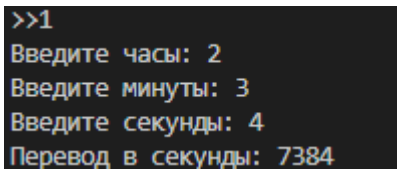

Рисунок 4.1 – Результат выполнения

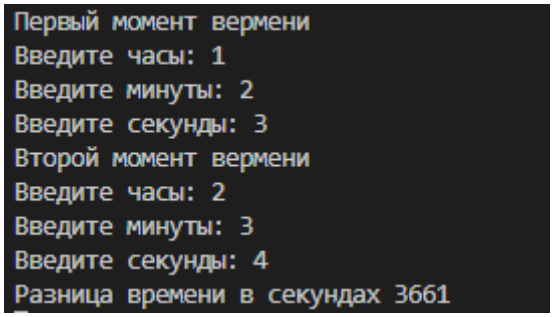

Рисунок 4.1 – Результат выполнения

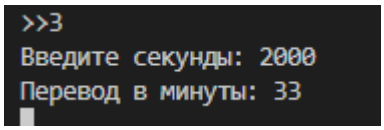

Рисунок 4.1 – Результат выполнения

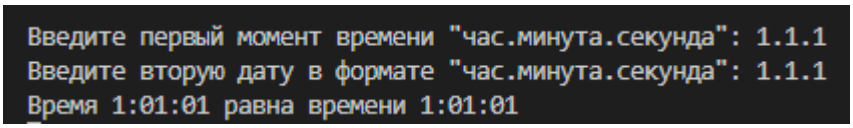

Рисунок 4.1 – Результат выполнения

Файл: https://github.com/Mihayilichenko/lab

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.