

**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 7

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Криворучко Ярослав Евгеньевич

_____ (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

7. Поле `first` — дробное число, левая граница диапазона; поле `second` — дробное число, правая граница диапазона. Реализовать метод `rangecheck()` — проверку заданного числа на принадлежность диапазону.

Был создан класс `Task` с полями `first` и `second`. Для проверки принадлежности числа к введенному диапазону, был создан специальный метод `def rangecheck()` (рисунок 1)

```
Ввод [4]: class Task:

    def __init__(self, first=0, second=2, number=1):
        self.first = float(first)
        self.second = float(second)
        self.number = float(number)

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

        print(self.first, "...", self.second)
        self.number = float(input("\nВведите любое число: "))
        #self.number = float(input() if prompt is None else input(prompt))

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

    def display(self):
        print(f"\nДиапазон: {self.first}...{self.second}. Введенное число: {self.number}")

if __name__ == '__main__':

    r1 = Task()
    r1.display()
    r1.rangecheck()
    r2 = Task()
    r2.read()
    r2.display()
    r2.rangecheck()
```

Рисунок 1 – Листинг программы

Результат работы программы изображен на рисунке 2

```
Диапазон: 0.0...2.0. Введенное число: 1.0

Число 1.0 входит в диапазон 0.0...2.0

Введите диапазон:
Введите левую границу диапазона: 1
Введите правую границу диапазона: 10
1.0 ... 10.0

Введите любое число: 5

Диапазон: 1.0...10.0. Введенное число: 5.0

Число 5.0 входит в диапазон 1.0...10.0
```

Рисунок 2 – Результат программы

Задание №2

7. Создать класс Date для работы с датами в формате «год.месяц.день». Дата представляется структурой с тремя полями типа unsigned int: для года, месяца и дня. Класс должен включать не менее трех функций инициализации: числами, строкой вида «год.месяц.день» (например, «2004.08.31») и датой. Обязательными операциями являются: вычисление даты через заданное количество дней, вычитание заданного количества дней из даты, определение високосности года, присвоение и получение отдельных частей (год, месяц, день), сравнение дат (равно, до, после), вычисление количества дней между датами.

Для решение данной задачи, был создан класс Date с тремя атрибутами экземпляра класса, определенных в методе `__init__` (year, month, day), в которые по умолчанию были переданы начальные данные (рисунок 3.1)

```
Ввод [ ]: from datetime import datetime, timedelta
|
class Date:
    def __init__(self, year, month, day):
        self.year = int(year)      #Год
        self.month = int(month)    #Месяц
        self.day = int(day)        #День
```

Рисунок 3.1 – Поля класса

Также были добавленные методы для инициализации даты разными способами (рисунок 3.2)

```
#Инициализация числами
def read_num(self):
    print("Инициализация числами: ")
    self.year = input("Введите год: ")
    self.month = input("Введите месяц: ")
    self.day = input("Введите день: ")
#Инициализация строкой ввода "год.месяц.день"
def read_stru(self):
    print("Инициализация строкой ввода \"год.месяц.день\":")
    date = input("Введите дату в формате \"год.месяц.день\": ")

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

    self.year = int(d[0])
    self.month = int(d[1])
    self.day = int(d[2])
#Инициализация датой
def read_date(self):
    print("Инициализация датой")
    date = input("Введите дату в формате \"год.месяц.день\": ")

    d = datetime.strptime(date, "%Y.%m.%d")
    self.year = int(d.year)
    self.month = int(d.month)
    self.day = int(d.day)
```

Рисунок 3.2 – Методы инициализации даты

Были созданы методы для реализации заданных по условию задач (рисунок 3.3)

```
#вычисление даты через заданное количество дней
def payment(self):
    date = datetime(self.year,self.month,self.day)
    num = int(input("Введите целое число: "))
    date += timedelta(days=num)
    print(f"Через {num} дней(я) дата станет равной: {date.year}.{date.month}.{date.day} \n")
#вычитание заданного количества дней из даты
def subtraction(self):
    date = datetime(self.year,self.month,self.day)
    num = int(input("Введите целое число: "))
    date -= timedelta(days=num)
    print(f"Через {num} дней(я) дата станет равной: {date.year}.{date.month}.{date.day} \n")
#определение високосности года
def LeapYear(self):
    year = int(input("Введите год для проверки: "))
    if (year % 4 == 0) and (year % 100 != 0) or (year % 400 == 0):
        print(f"Год {year} високосный")
    else:
        print(f"Год {year} не високосный")
#сравнение дат (равно, до, после)
#вычисление количества дней между датами
def comparison(self):
    print()
    date = input("Введите первую дату в формате \"год.месяц.день\": ")
    d = date.split('.', maxsplit=2)
    date1 = datetime(int(d[0]),int(d[1]),int(d[2]))

    date = input("Введите вторую дату в формате \"год.месяц.день\": ")
    d = date.split('.', maxsplit=2)
    date2 = datetime(int(d[0]),int(d[1]),int(d[2]))

    if(date1>date2):
        print(f"Дата {date1.year}.{date1.month}.{date1.day} находится после даты {date2.year}.{date2.month}.{date2.day}")
        #self.cnumber(date1,date2)
    elif(date1<date2):
        print(f"Дата {date1.year}.{date1.month}.{date1.day} находится до даты {date2.year}.{date2.month}.{date2.day}")
        date1,date2 = date2,date1
        #self.cnumber(date2,date1)
    else:
        print(f"Дата {date1.year}.{date1.month}.{date1.day} равна дате {date2.year}.{date2.month}.{date2.day}")
        #self.cnumber(date1,date2)
    timedelta = date1 - date2
    print(f"Количество дней между введенными датами равно: {timedelta.days} дня")

#вывод на дисплей
def display(self):
    print(f"Дата: {self.year}.{self.month}.{self.day}")
```

Рисунок 3.3 – Методы класса

Для удобства реализовано меню, для выбора определенных методов класса (рисунок 3.4)

```

if __name__ == '__main__':
    print("\n-----\n1. Вызов метода read_num\n-----")
    print("2. Вызов метода display\n-----")
    print("3. Вызов метода read_stru\n-----")
    print("4. Вызов метода read_date\n-----")
    print("5. Вызов метода payment\n-----")
    print("6. Вызов метода subtraction\n-----")
    print("7. Вызов метода LeapYear\n-----")
    print("8. Вызов метода comparison\n-----")
    d = Date(2022,11,5)
    while True:
        conf = input("\nВведите номер меню: ")
        if(conf=='1'):
            print("\n-----\nВызов метода read_num\n-----\n")
            d.read_num()
            print("\n-----\nВызов метода display\n-----\n")
            d.display()
        elif(conf=='2'):
            print("\n-----\nВызов метода display\n-----\n")
            d.display()
        elif(conf=='3'):
            print("\n-----\nВызов метода read_stru\n-----\n")
            d.read_stru()
            d.display()
        elif(conf=='4'):
            print("\n-----\nВызов метода read_date\n-----\n")
            d.read_date()
            d.display()
        elif(conf=='5'):
            print("\n-----\nВызов метода payment\n-----\n")
            d.payment()
        elif(conf=='6'):
            print("\n-----\nВызов метода subtraction\n-----\n")
            d.subtraction()
        elif(conf=='7'):
            print("\n-----\nВызов метода LeapYear\n-----\n")
            d.LeapYear()
        elif(conf=='8'):
            print("\n-----\nВызов метода comparison\n-----\n")
            d.comparison()
        else:
            print("Данного пункта нет в меню!")

```

Рисунок 3.4 – Листинг программы

Результат работы программы изображен на рисунках 4.1-4.4

```
-----  
1. Вызов метода read_num  
-----  
2. Вызов метода display  
-----  
3. Вызов метода read_stru  
-----  
4. Вызов метода read_date  
-----  
5. Вызов метода payment  
-----  
6. Вызов метода subtraction  
-----  
7. Вызов метода LeapYear  
-----  
8. Вызов метода comparison  
-----
```

Введите номер меню: 1

```
-----  
Вызов метода read_num  
-----
```

Инициализация числами:
Введите год: 2015
Введите месяц: 5
Введите день: 5

```
-----  
Вызов метода display  
-----
```

Дата: 2015.5.5

Введите номер меню: 2

```
-----  
Вызов метода display  
-----
```

Дата: 2015.5.5

Рисунок 4.1 – Результат выполнения

```

Введите номер меню: 3
-----
Вызов метода read_stru
-----

Инициализация строкой ввода "год.месяц.день":
Введите дату в формате "год.месяц.день": 2016.6.5
Дата: 2016.6.5

Введите номер меню: 4
-----
Вызов метода read_date
-----

Инициализация датой
Введите дату в формате "год.месяц.день": 2003.11.25
Дата: 2003.11.25

Введите номер меню: 5
-----
Вызов метода payment
-----

Введите целое число: 30
Через 30 дней(я) дата станет равной: 2003.12.25

Введите номер меню: 6
-----
Вызов метода subtraction
-----

Введите целое число: 30
Через 30 дней(я) дата станет равной: 2003.10.26

```

Рисунок 4.1 – Результат выполнения

```

Введите номер меню: 7
-----
Вызов метода LeapYear
-----

Введите год для проверки: 2100
Год 2100 не високосный

Введите номер меню: 8
-----
Вызов метода comparison
-----

Введите первую дату в формате "год.месяц.день": 2022.4.9
Введите вторую дату в формате "год.месяц.день": 2000.4.9
Дата 2022.4.9 находится после даты 2000.4.9
Количество дней между введенными датами равно: 8035 дня

```

Рисунок 4.1 – Результат выполнения

Файл 4.1.ipynb с решением задач находится на **Github**:
https://github.com/YaroStavr/LR8_Artificial-Intelligence.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 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`.