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 4

Выполнил: студент группы ИВТ-б-о-18-1
Дрищёв Данила Николаевич
_____(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

Ввод [5]:

```
class bank():
    def __init__(self, first = 0, second = 0):
        self.first = first
        self.second = second
        self.list = [1, 2 , 5, 10, 50, 100, 500, 1000, 5000]

    def summ(self):
        return self.first * self.second
    def read(self):
        print(self.list)
        self.first = (int(input("Введите номинал купюры: ")))
        self.second = (int(input("Введите кол-во купюр: ")))
        if self.first in self.list:
            self.display()
        else:
            return "Выберите другое значение номинала купюры"

    #вывод на экран
    def display(self):
        print(f'Сумма равна: {self.summ()}')
if __name__ == '__main__':
    bank = bank()
    key = int(input("Нажмите 1 чтобы вывести возможные номиналы"))
    if key == 1:
        bank.read()
    else:
        print("Ошибка")

```

```
Нажмите 1 чтобы вывести возможные номиналы1
[1, 2, 5, 10, 50, 100, 500, 1000, 5000]
Введите номинал купюры: 100
Введите кол-во купюр: 50
Сумма равна: 5000
```

Ввод [ ]:

```
Поле first – целое положительное число, номинал купюры; номинал может принимать
значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000. Поле second – целое положительное число,
количество купюр данного достоинства. Реализовать метод summa() – вычисление
денежной суммы.
```


Рисунок 1 - Готовая программа

Задание №2

Ввод [ ]:
```
Создать класс Triangle для представления треугольника. Поля данных должны включать
углы и стороны. Требуется реализовать операции: получения и изменения полей данных,
вычисления площади, вычисления периметра, вычисления высот, а также определения
вида треугольника (равносторонний, равнобедренный или прямоугольный).
```

Ввод [3]:
```
import math
class Triangle():
    def __init__(self, a, b, c, ab ,bc ,ac):
        self.a = a
        self.b = b
        self.c = c
        self.ab = ab
        self.bc = bc
        self.ac = ac

    def perimeter(self):
        return self.ab + self.bc + self.ac

    def visota(self):
        return (self.ab + self.bc + self.ac)/2

    def ploshchad(self):
        return math.sqrt(self.visota()*(self.visota()-self.ab)*(self.visota()-self.bc)*(self.visota()-self.ac))

    def dype(self):
        if (self.ab*self.ab + self.bc*self.bc == self.ac*self.ac) or (self.ab*self.ab + self.ac*self.ac == self.bc*self.bc)
            return("Прямоугольный")
        elif (self.ab*self.ab + self.bc*self.bc < self.ac*self.ac) or (self.ab*self.ab + self.ac*self.ac < self.bc*self.bc)
            return("Равнобедренный")
        else:
            return("Равносторонний")

if __name__ == '__main__':
    Triangle = Triangle(60, 60, 60, 3, 4, 5)
    print("Периметр = ", Triangle.perimeter())
    print("Высота =", Triangle.visota())
    print("Площадь = ", Triangle.ploshchad())
    print("Тип треугольника = ", Triangle.dype())
```

```
Периметр =  12
Высота = 6.0
Площадь =  6.0
Тип треугольника =  Прямоугольный
```


Рисунок 2 – Готовая программа

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.