Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины
«Объектно-ориентированное программирование»
Вариант 20

Выполнила:
Михеева Елена Александровна
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Воронкин Р.А.-доцент департамента цифровых, роботехнических систем и электроники института перспективной инженерии

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Элементы объектно-ориентированного программирования в языке Python.

Цель работы: приобретение навыков по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Был изучен теоретический материал работы.
2. Приступили к выполнению задания №1: Парой называется класс с двумя полями, которые обычно имеют имена *first* и *second*. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать:

– метод инициализации __init__ ; метод должен контролировать значения аргументов на корректность;

– ввод с клавиатуры read ;

– вывод на экран display .

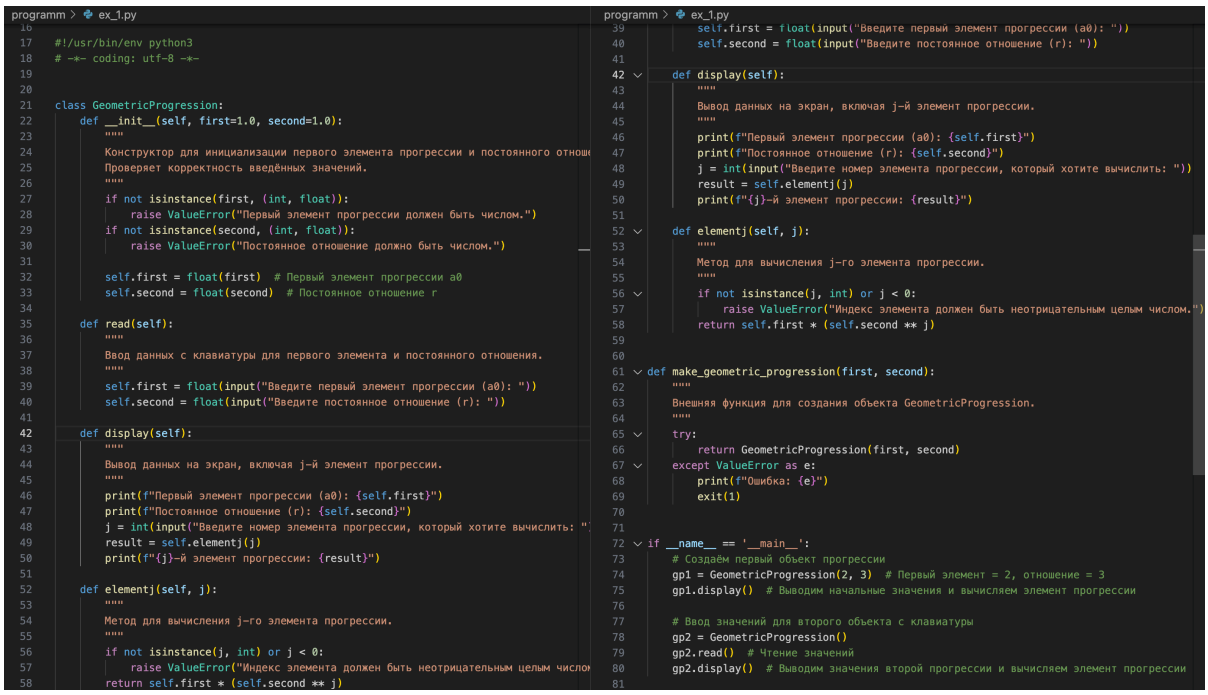
Реализовать внешнюю функцию с именем make_тип() , где тип — тип реализуемой структуры. Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

В раздел программы, начинающийся после инструкции if __name__ = '__main__': добавить код, демонстрирующий возможности разработанного класса.

Элемент геометрической прогрессии вычисляется по формуле:

$$a_j = a0 \cdot r^j, j = 0, 1, 2, \ldots$$

Поле first — дробное число, первый элемент прогрессии a; поле second — постоянное отношение. Определить метод для вычисления заданного элемента прогрессии.



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class GeometricProgression:
    def __init__(self, first=1.0, second=1.0):
        """
        Конструктор для инициализации первого элемента прогрессии и постоянного отноше
        Проверяет корректность введённых значений.
        """
        if not isinstance(first, (int, float)):
            raise ValueError("Первый элемент прогрессии должен быть числом.")
        if not isinstance(second, (int, float)):
            raise ValueError("Постоянное отношение должно быть числом.")

        self.first = float(first)  # Первый элемент прогрессии a0
        self.second = float(second)  # Постоянное отношение r

    def read(self):
        """
        Ввод данных с клавиатуры для первого элемента и постоянного отношения.
        """
        self.first = float(input("Введите первый элемент прогрессии (a0): "))
        self.second = float(input("Введите постоянное отношение (r): "))

    def display(self):
        """
        Вывод данных на экран, включая j-й элемент прогрессии.
        """
        print(f"Первый элемент прогрессии (a0): {self.first}")
        print(f"Постоянное отношение (r): {self.second}")
        j = int(input("Введите номер элемента прогрессии, который хотите вычислить: "
        result = self.elementj(j)
        print(f"{j}-й элемент прогрессии: {result}")

    def elementj(self, j):
        """
        Метод для вычисления j-го элемента прогрессии.
        """
        if not isinstance(j, int) or j < 0:
            raise ValueError("Индекс элемента должен быть неотрицательным целым число
        return self.first * (self.second ** j)
```

```python
        self.first = float(input("Введите первый элемент прогрессии (a0): "))
        self.second = float(input("Введите постоянное отношение (r): "))

    def display(self):
        """
        Вывод данных на экран, включая j-й элемент прогрессии.
        """
        print(f"Первый элемент прогрессии (a0): {self.first}")
        print(f"Постоянное отношение (r): {self.second}")
        j = int(input("Введите номер элемента прогрессии, который хотите вычислить: "))
        result = self.elementj(j)
        print(f"{j}-й элемент прогрессии: {result}")

    def elementj(self, j):
        """
        Метод для вычисления j-го элемента прогрессии.
        """
        if not isinstance(j, int) or j < 0:
            raise ValueError("Индекс элемента должен быть неотрицательным целым числом.")
        return self.first * (self.second ** j)


def make_geometric_progression(first, second):
    """
    Внешняя функция для создания объекта GeometricProgression.
    """
    try:
        return GeometricProgression(first, second)
    except ValueError as e:
        print(f"Ошибка: {e}")
        exit(1)


if __name__ == '__main__':
    # Создаём первый объект прогрессии
    gp1 = GeometricProgression(2, 3)  # Первый элемент = 2, отношение = 3
    gp1.display()  # Выводим начальные значения и вычисляем элемент прогрессии

    # Ввод значений для второго объекта с клавиатуры
    gp2 = GeometricProgression()
    gp2.read()  # Чтение значений
    gp2.display()  # Выводим значения второй прогрессии и вычисляем элемент прогрессии
```

Рисунок 1. Код программы задания №1

```
(base) elenamiheeva@MacBook-Pro-Elena Oop_1 % python3 programm/ex_1.py
Первый элемент прогрессии (a0): 2.0
Постоянное отношение (r): 3.0
Введите номер элемента прогрессии, который хотите вычислить: 6
6-й элемент прогрессии: 1458.0
Введите первый элемент прогрессии (a0): 2.7
Введите постоянное отношение (r): 3.8
Первый элемент прогрессии (a0): 2.7
Постоянное отношение (r): 3.8
Введите номер элемента прогрессии, который хотите вычислить: 6
6-й элемент прогрессии: 8129.528236799998
(base) elenamiheeva@MacBook-Pro-Elena Oop_1 %
```

Рисунок 2. Результат работы программы задания №1

3. Приступили к выполнению задания №2: составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

– метод инициализации __init__ ;

– ввод с клавиатуры read ;

– вывод на экран display .

В раздел программы, начинающийся после инструкции if __name__ = '__main__': добавить код, демонстрирующий возможности разработанного класса.

Создать класс Angle для работы с углами на плоскости, задаваемыми величиной в градусах и минутах. Обязательно должны быть реализованы:

– перевод в радианы,

– приведение к диапазону 0-360,

– увеличение и уменьшение угла на заданную величину,

– получение синуса,

– сравнение углов.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import math


class Angle:
    def __init__(self, degrees=0.0, minutes=0.0):
        """
        Конструктор для инициализации угла в градусах и минутах.
        """
        if not isinstance(degrees, (int, float)):
            raise ValueError("Значение градусов должно быть числом.")
        if not isinstance(minutes, (int, float)):
            raise ValueError("Значение минут должно быть числом.")

        self.degrees = degrees
        self.minutes = minutes
        self.normalize360()

    def read(self):
        """
        Ввод данных с клавиатуры для работы с углами на плоскости.
        """
        self.degrees = float(input("Введите градусы: "))
        self.minutes = float(input("Введите минуты: "))
        self.normalize360()

    def display(self):
        """
        Вывод данных на экран.
        """
        print(f"Угол: {self.degrees} градусов, {self.minutes} минут")

    def to_decimal_degrees(self):
        """
        Преобразование угла в десятичные градусы.
        """
        return self.degrees + self.minutes / 60.0

    def radians(self):
        """
        Перевод в радианы.
        """
        return math.radians(self.to_decimal_degrees())

    def normalize360(self):
        """
        Приведение угла к диапазону от 0 до 360 градусов.
        """
        total_degrees = self.to_decimal_degrees()
        normalized_degrees = total_degrees % 360
        self.degrees = int(normalized_degrees)
        self.minutes = (normalized_degrees - self.degrees) * 60

    def add_angle(self, degrees=0.0, minutes=0.0):
        """
        Увеличение угла на заданную величину в градусах и минутах.
        """
        total_minutes = (self.degrees + degrees) * 60 + self.minutes + minutes
        self.degrees = total_minutes // 60
        self.minutes = total_minutes % 60
        self.normalize360()

    def subtract_angle(self, degrees=0.0, minutes=0.0):
        """
        Уменьшение угла на заданную величину в градусах и минутах.
        """
        total_minutes = (self.degrees * 60 + self.minutes) - (degrees * 60 + minutes)
        self.degrees = total_minutes // 60
        self.minutes = total_minutes % 60
        self.normalize360()

    def sinangle(self):
        """
        Получение синуса угла.
        """
        return math.sin(self.radians())

    # Сравнение углов
    def equals(self, other):
        """
        Проверка на равенство двух углов.
        """
        if isinstance(other, Angle):
            return math.isclose(self.to_decimal_degrees(), other.to_decimal_degrees())
        return False

    def greater(self, other):
        """
        Проверка, больше ли текущий угол, чем другой угол.
        """
        if isinstance(other, Angle):
            return self.to_decimal_degrees() > other.to_decimal_degrees()
        return False

    def less(self, other):
        """
        Проверка, меньше ли текущий угол, чем другой угол.
        """
        if isinstance(other, Angle):
            return self.to_decimal_degrees() < other.to_decimal_degrees()
        else:
            return False


if __name__ == '__main__':
    angle1 = Angle(30, 20)  # Угол 30 градусов 20 минут
    angle2 = Angle(45, 50)  # Угол 45 градусов 50 минут
    angle1.display()
    angle2.display()

    print("Угол 1 в радианах:", angle1.radians())
    print("Синус угла 1:", angle1.sinangle())

    print("Угол 1 == Угол 2:", angle1.equals(angle2))
    print("Угол 1 > Угол 2:", angle1.greater(angle2))
    print("Угол 1 < Угол 2:", angle1.less(angle2))

    # Увеличение угол 1 на 10 градусов и 30 минут
    angle1.add_angle(10, 30)
    angle1.display()

    # Уменьшение угол 2 на 5 градусов и 40 минут
    angle2.subtract_angle(5, 40)
    angle2.display()

```

Рисунок 3. Код программы задния№2

```
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_1 % python3 programm/ex_2.py
Угол: 30 градусов, 19.99999999999993 минут
Угол: 45 градусов, 50.00000000000014 минут
Угол 1 в радианах: 0.5294165397716133
Синус угла 1: 0.50502984157469
Угол 1 == Угол 2: False
Угол 1 > Угол 2: False
Угол 1 < Угол 2: True
Угол: 40 градусов, 50.00000000000014 минут
Угол: 40 градусов, 9.999999999999858 минут
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_1 %
```

Рисунок 4. Результаты работы программы задания №2

Ответы на контрольные вопросы

1. Как осуществляется объявление класса в языке Python?

Объявление класса в Python осуществляется с использованием ключевого слова «class», за которым следует имя класса и двоеточие. Внутри класса могут быть определены методы и атрибуты.

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса являются общими для всех экземпляров класса, они определяются на уровне класса и доступны всем экземплярам. Атрибуты экземпляра, наоборот, создаются для конкретного объекта и могут иметь уникальные значения для каждого экземпляра.

3. Каково назначение методов класса?

Методы класса — это функции, определенные внутри класса, которые описывают поведение объектов данного класса. Они позволяют работать с атрибутами экземпляров и класса, а также реализовать функциональность, связанную с объектами данного класса.

4. Для чего предназначен метод «__init__()» класса?

Метод «__init__()» является конструктором класса и автоматически вызывается при создании нового экземпляра класса. Он используется для инициализации атрибутов объекта, то есть для задания их начальных значений.

5. Каково назначение «self»?

«self» — это ссылка на текущий экземпляр класса. Через «self» можно получить доступ к атрибутам и методам объекта, для которого был вызван метод. Она должна быть первым параметром всех методов экземпляра класса.

4. Как добавить атрибуты в класс?

Атрибуты можно добавить непосредственно в класс, задав их внутри класса, либо динамически, добавив их в экземпляр или сам класс после его создания.

```
class MyClass:
    class_attr = 0  # Атрибут класса
    def __init__(self, instance_attr):
        self.instance_attr = instance_attr  # Атрибут экземпляра
```

Динамическое добавление атрибута экземпляру

obj = MyClass(5)

obj.new_attr = 10

6. Как осуществляется управление доступом к методам и атрибутам в языке Python?

В Python управление доступом осуществляется через соглашение об именовании. Атрибуты и методы, начинающиеся с одного подчеркивания «_», считаются защищенными (protected) и не предназначены для прямого использования вне класса. Атрибуты и методы, начинающиеся с двойного подчеркивания «__», считаются приватными (private) и монтируются (переписываются) для предотвращения случайного доступа извне.

7. Каково назначение функции «isinstance»?

Функция «isinstance» используется для проверки, принадлежит ли объект к определенному классу или группе классов (кортежу классов). Возвращает «True», если объект является экземпляром указанного класса или его подклассов, и «False» в противном случае.

Вывод: в ходе работы были приобретены навыки по работе с классами и объектами при написании программ.