Студент группы ИС-23 Григорян М.Г.

Практическое занятие №16

Тема: Составление программ с использованием ООП в IDE PyCharm Community.

Цели: Закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составлениепрограмм с ООП в IDE PyCharm Community.

Постановка задачи:

Создайте класс «Матрица», который имеет атрибуты количества строк и столбцов. Добавьте методы для сложения, вычитания и умножения матриц.

Текст программы:

PZ_16 › pz_16_1.py › Matrix › _add_

```python
# Создайте класс «Матрица», который имеет атрибуты количества строк и столбцов.
# Добавьте методы для сложения, вычитания и умножения матриц.

class Matrix:
    def __init__(self, data=None, rows=3, cols=3):
        self.data = [[0 for _ in range(cols)] for _ in range(rows)] if not data else data
        self.rows = len(self.data)
        self.cols = len(max(self.data))

        if len(max(self.data)) != len(min(self.data)):
            raise ValueError("Matrix must have same count of columns and rows")

    def __add__(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Matrices must have the same dimensions")
        result = Matrix(rows=self.rows, cols=self.cols)
        for i in range(self.rows):
            for j in range(self.cols):
                result.data[i][j] = self.data[i][j] + other.data[i][j]
        return result

    def __sub__(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Matrices must have the same dimensions")
        result = Matrix(rows=self.rows, cols=self.cols)
        for i in range(self.rows):
            for j in range(self.cols):
                result.data[i][j] = self.data[i][j] - other.data[i][j]
        return result

    def __mul__(self, other):
        if self.cols != other.rows:
            raise ValueError("Number of columns in the first matrix must be equal to the number of rows in the second matrix")
        result = Matrix(rows=self.rows, cols=other.cols)
        for i in range(self.rows):
            for j in range(other.cols):
                for k in range(self.cols):
```

```
                    result.data[i][j] += self.data[i][k] * other.data[k][j]
        return result

    def __str__(self):
        return '\n'.join([' '.join(map(str, row)) for row in self.data])


if __name__ == '__main__':
    matrix1 = Matrix(
        [[1, 2], [3, 4]],
    )

    matrix2 = Matrix(
        [[5, 6], [7, 8]],
    )

    sum_matrix = matrix1 + matrix2
    print("Сумма матриц:")
    print(sum_matrix)

    diff_matrix = matrix1 - matrix2
    print("Разность матриц:")
    print(diff_matrix)

    product_matrix = matrix1 * matrix2
    print("Произведение матриц:")
    print(product_matrix)

```

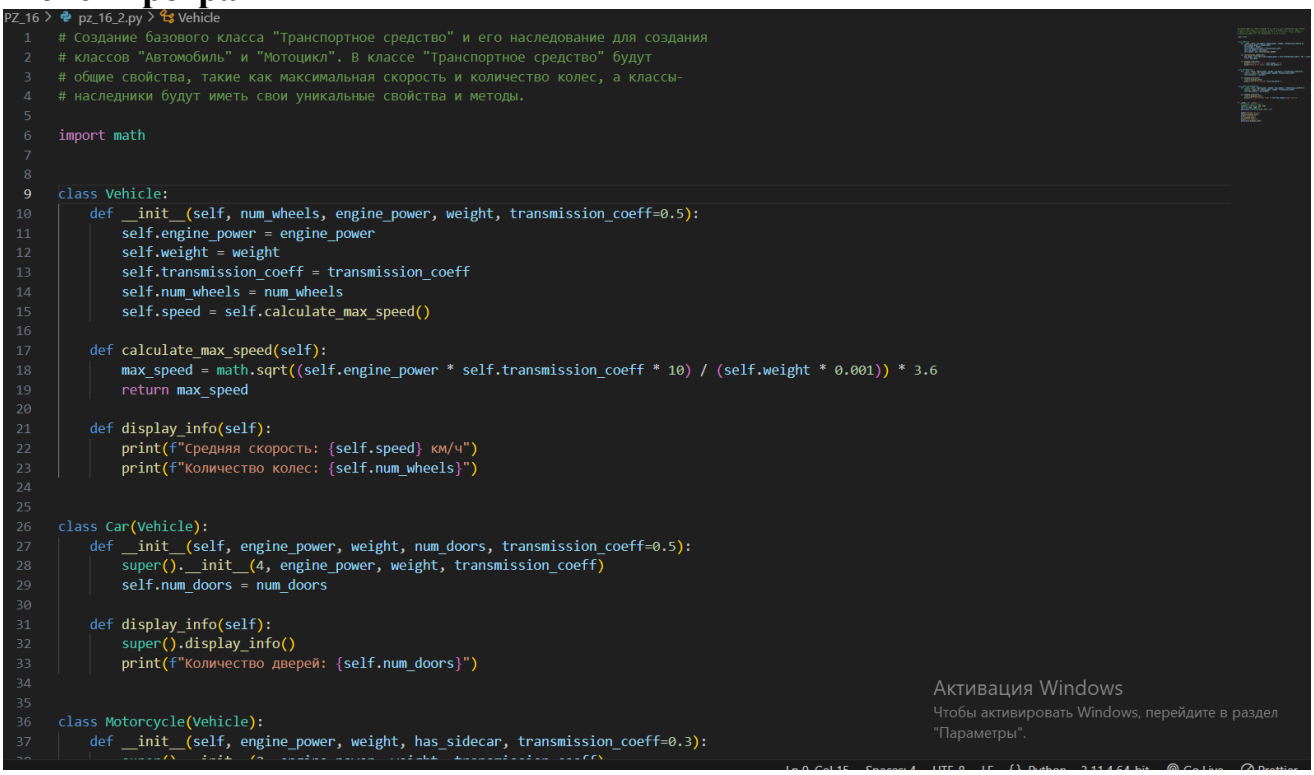
Протокол программы:

```
PS C:\Users\User\Desktop\PZ> & C:/Users/User/AppData/Local/Programs/Python/Python311/python.exe c:/Users/User/Desktop/PZ/PZ_16/pz_16_1.py
Сумма матриц:
6 8
10 12
Разность матриц:
-4 -4
-4 -4
Произведение матриц:
19 22
43 50
PS C:\Users\User\Desktop\PZ>
```

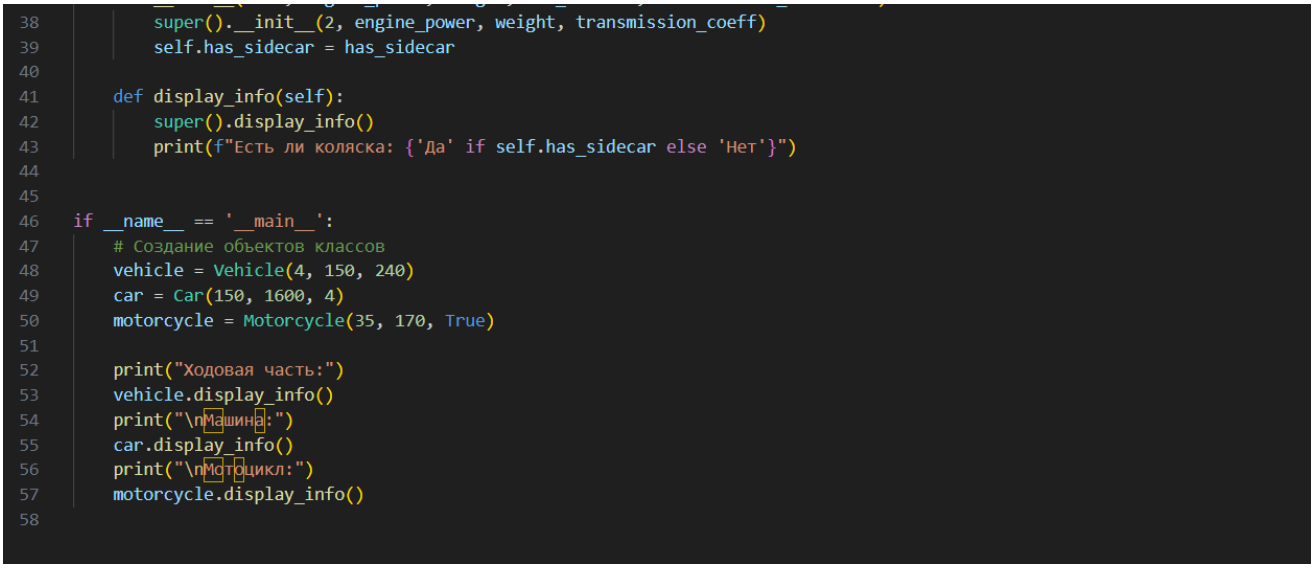
Постановка задачи:

Создайте базовый класс “Транспортное средство” и его наследование для создания классов “Автомобиль” и “Мотоцикл”. В классе “Транспортное средство” будут общие свойства, такие как максимальная скорость и количество колес, а классы-наследники будут иметь свои уникальные свойства и методы.

Текст программы:



```
PZ_16 > pz_16_2.py > Vehicle
# Создание базового класса "Транспортное средство" и его наследование для создания
# классов "Автомобиль" и "Мотоцикл". В классе "Транспортное средство" будут
# общие свойства, такие как максимальная скорость и количество колес, а классы-
# наследники будут иметь свои уникальные свойства и методы.

import math


class Vehicle:
    def __init__(self, num_wheels, engine_power, weight, transmission_coeff=0.5):
        self.engine_power = engine_power
        self.weight = weight
        self.transmission_coeff = transmission_coeff
        self.num_wheels = num_wheels
        self.speed = self.calculate_max_speed()

    def calculate_max_speed(self):
        max_speed = math.sqrt((self.engine_power * self.transmission_coeff * 10) / (self.weight * 0.001)) * 3.6
        return max_speed

    def display_info(self):
        print(f"Средняя скорость: {self.speed} км/ч")
        print(f"Количество колес: {self.num_wheels}")


class Car(Vehicle):
    def __init__(self, engine_power, weight, num_doors, transmission_coeff=0.5):
        super().__init__(4, engine_power, weight, transmission_coeff)
        self.num_doors = num_doors

    def display_info(self):
        super().display_info()
        print(f"Количество дверей: {self.num_doors}")


class Motorcycle(Vehicle):
    def __init__(self, engine_power, weight, has_sidecar, transmission_coeff=0.3):
```

```
        super().__init__(2, engine_power, weight, transmission_coeff)
        self.has_sidecar = has_sidecar

    def display_info(self):
        super().display_info()
        print(f"Есть ли коляска: {'Да' if self.has_sidecar else 'Нет'}")


if __name__ == '__main__':
    # Создание объектов классов
    vehicle = Vehicle(4, 150, 240)
    car = Car(150, 1600, 4)
    motorcycle = Motorcycle(35, 170, True)

    print("Ходовая часть:")
    vehicle.display_info()
    print("\nМашина:")
    car.display_info()
    print("\nМотоцикл:")
    motorcycle.display_info()
```

Протокол программы:

```
PS C:\Users\User\Desktop\PZ>
                          > & C:/Users/User/AppData/Local/Programs/Python/Python311/python.exe c:/Users/User/Desktop/PZ/PZ_16/pz_16_2.py
Ходовая часть:
Средняя скорость: 201.24611797498108 км/ч
Количество колес: 4

Машина:
Средняя скорость: 77.94228634059948 км/ч
Количество колес: 4
Количество дверей: 4

Мотоцикл:
Средняя скорость: 89.46902191458751 км/ч
Количество колес: 2
Есть ли коляска: Да
PS C:\Users\User\Desktop\PZ>
```

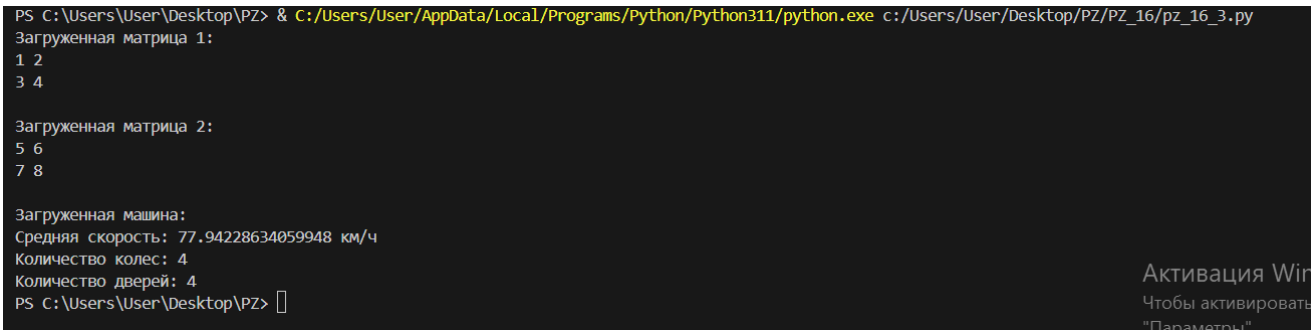
Постановка задачи:

Для задачи из блока 1 создать две функции, save_def и load_def, которые позволяют сохранять информацию из экземпляров класса (3 шт.) в файл и загружать ее обратно. Использовать модуль pickle для сериализации и десериализации объектов Python в бинарном формате.

Текст программы:

```
import pickle

from pz_16_1 import Matrix
from pz_16_2 import Car


def save_def(filename, objects):
    with open(filename, 'wb') as file:
        pickle.dump(objects, file)


def load_def(filename):
    with open(filename, 'rb') as file:
        objects = pickle.load(file)
    return objects


matrix1 = Matrix([[1, 2], [3, 4]], 2, 2)

matrix2 = Matrix([[5, 6], [7, 8]], 2, 2)

car = Car(150, 1600, 4)

objects = [matrix1, matrix2, car]
save_def("objects.pkl", objects)

loaded_objects = load_def("objects.pkl")

print("Загруженная матрица 1:")
print(loaded_objects[0])
print("\nЗагруженная матрица 2:")
print(loaded_objects[1])
print("\nЗагруженная машина:")
loaded_objects[2].display_info()
```

Протокол программы:



Вывод:
Я закрепила усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрела навык и составлениепрограмм с ООП в IDE PyCharm Community.