Студент группы ИС-27 Прудников А.Г.

Практическое занятие № 16

Тема: Работа с Классами в IDE PyCharm Community.

Цель: закрепить усвоенные знания работы с классами,
основными принципы составления программ, приобрести навыки составление программ с классами и наследованием в IDE PyCharm Community.

Постановка задачи 1.

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

Текст программы:

```
class Matrix:
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols
        self.data = [[0 for _ in range(cols)] for _ in range(rows)]

    def add(self, other):
        if self.rows == other.rows and self.cols == other.cols:
            result = Matrix(self.rows, 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
        else:
            return "Матрицы имеют разные размеры и не могут быть
сложены"

    def subtract(self, other):
        if self.rows == other.rows and self.cols == other.cols:
            result = Matrix(self.rows, 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
        else:
            return "Матрицы имеют разные размеры и не могут быть
вычтены"
```

```
    def multiply(self, other):
        if self.cols == other.rows:
            result = Matrix(self.rows, 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
        else:
            return "Умножение невозможно из-за неправильных
размеров матриц"


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

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

print(matrix1.add(matrix2).data)
print(matrix1.subtract(matrix2).data)
print(matrix1.multiply(matrix2).data)
```

Протокол работы программы:

[[6, 8], [10, 12]]
[[-4, -4], [-4, -4]]
[[5, 12], [21, 32]]

Постановка задачи 2.

```
'''
7 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.
'''
```

Текст программы:

```
class Transport:
    def __init__(self, max_speed, num_wheels, color):
        self.max_speed = max_speed
        self.num_wheels = num_wheels
        self.color = color

class Car(Transport):
```

```
    def __init__(self, max_speed, num_wheels, color, brand):
        super().__init__(max_speed, num_wheels, color)
        self.brand = brand

    def honk(self):
        return "Beep beep!"

class Motorcycle(Transport):
    def __init__(self, max_speed, num_wheels, color, type):
        super().__init__(max_speed, num_wheels, color)
        self.type = type

    def wheelie(self):
        return "Doing a wheelie!"

# Пример использования
car = Car(200, 4, "Red", "Toyota")
print(car.max_speed)  # Выводит: 200
print(car.honk())  # Выводит: Beep beep!

motorcycle = Motorcycle(180, 2, "Blue", "Sport")
print(motorcycle.color)  # Выводит: Blue
print(motorcycle.wheelie())  # Выводит: Doing a wheelie!
```

Протокол работы программы:

200
Beep beep!
Blue
Doing a wheelie!

Постановка задачи 3.

```
'''
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

class Matrix:
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols
        self.data = [[0 for _ in range(cols)] for _ in range(rows)]

    def add(self, other):
        if self.rows == other.rows and self.cols == other.cols:
```

```
            result = Matrix(self.rows, 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
        else:
            return "Матрицы имеют разные размеры и не могут быть
сложены"

    def subtract(self, other):
        if self.rows == other.rows and self.cols == other.cols:
            result = Matrix(self.rows, 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
        else:
            return "Матрицы имеют разные размеры и не могут быть
вычтены"

    def multiply(self, other):
        if self.cols == other.rows:
            result = Matrix(self.rows, 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
        else:
            return "Умножение невозможно из-за неправильных
размеров матриц"

    def save(self, filename):
        with open(filename, 'wb') as file:
            pickle.dump(self, file)

    @staticmethod
    def load(filename):
        with open(filename, 'rb') as file:
            obj = pickle.load(file)
            return obj

# Пример использования
matrix1 = Matrix(2, 2)
matrix1.data = [[1, 2], [3, 4]]

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

```
# Сохранение матрицы в файл
matrix1.save('matrix_data.pkl')

# Загрузка матрицы из файла
loaded_matrix = Matrix.load('matrix_data.pkl')

print(loaded_matrix.add(matrix2).data) # Сложение матриц
print(loaded_matrix.subtract(matrix2).data)  # Вычитание матриц
print(loaded_matrix.multiply(matrix2).data)  # Умножение матриц
```

Протокол работы программы:

[[6, 8], [10, 12]]
[[-4, -4], [-4, -4]]
[[5, 12], [21, 32]]

Вывод: в процессе выполнения практического занятия выработал навыки работы с классами, наследованием в IDE PyCharm Community. Выполнены разработка кода, отладка, тестирование, оптимизация программного кода. Готовые программные коды выложены на GitHub.