Практическое занятие № 16

Задача №1

Тема: составление программ для работы с ООП и библиотекой pickle в IDE PyCharm Community.

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, работая с библиотекой pickle 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:

```
import pickle

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

    def __str__(self):
        return "\n".join(" ".join(str(x) for x in row) for row in
self.matrix)

    def __add__(self, other):
        if self.rows != other.rows or self.columns != other.columns:
            raise ValueError("Матрицы должны иметь одинаковые размеры.")
        result = Matrix(self.rows, self.columns)
        for i in range(self.rows):
            for j in range(self.columns):
                result.matrix[i][j] = self.matrix[i][j] + other.matrix[i][j]
        return result

    def __sub__(self, other):
        if self.rows != other.rows or self.columns != other.columns:
            raise ValueError("Матрицы должны иметь одинаковые размеры.")
        result = Matrix(self.rows, self.columns)
        for i in range(self.rows):
            for j in range(self.columns):
                result.matrix[i][j] = self.matrix[i][j] - other.matrix[i][j]
        return result

    def __mul__(self, other):
        if self.columns != other.rows:
            raise ValueError(
```

```
                "Количество столбцов первой матрицы должно быть равно "
                "количеству строк второй матрицы."
            )
        result = Matrix(self.rows, other.columns)
        for i in range(self.rows):
            for j in range(other.columns):
                for k in range(self.columns):
                    result.matrix[i][j] += self.matrix[i][k] *
other.matrix[k][j]
        return result


def save_def(matrix_list):
    with open('pz-16/matrix_data.pkl', 'wb') as f:
        pickle.dump(matrix_list, f)

def load_def():
    try:
        with open('pz-16/matrix_data.pkl', 'rb') as f:
            matrix_list = pickle.load(f)
        return matrix_list
    except FileNotFoundError:
        return None



# Пример использования
matrix1 = Matrix(2, 3)
matrix1.matrix = [[1, 2, 3], [4, 5, 6]]
matrix2 = Matrix(3, 2)
matrix2.matrix = [[7, 8], [9, 10], [11, 12]]
matrix3 = Matrix(4, 5)
matrix3.matrix = [[4, 5, 6], [7, 8, 9]]

matrix_list = [matrix1, matrix2, matrix3]

save_def(matrix_list)

loaded = load_def()



print("Матрица 1:")
print(matrix1)

print("\nМатрица 2:")
print(matrix2)

# print("\nСложение:")
# print(matrix1 + matrix2)

# print("\nВычитание:")
# print(matrix1 - matrix2)  # Вызовет ValueError, т.к. размеры не совпадают

print("\nУмножение:")
print(matrix1 * matrix2)

print("\nУмножение:")
print(matrix2 * matrix1)

print('\n')

for i in loaded:
    print(i)
```

Протокол работы программы:

Матрица 1:
1 2 3
4 5 6

Матрица 2:
7 8
9 10
11 12

Умножение:
58 64
139 154

Умножение:
39 54 69
49 68 87
59 82 105

1 2 3
4 5 6
7 8
9 10
11 12
4 5 6
7 8 9

Process finished with exit code 0

Практическая №2

Постановка задачи:

Создайте базовый класс "Человек" со свойствами "имя", "возраст" и "пол". От этого
класса унаследуйте классы "Мужчина" и "Женщина" и добавьте в них свойства,
связанные с социальным положением (например, "семейное положение",
"количество детей" и т.д.).

Текст программы:

```
class Person:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def __str__(self):
        return f"Имя: {self.name}, Возраст: {self.age}, Пол: {self.gender}"

class Man(Person):
    def __init__(self, name, age, marital_status, number_of_children):
        super().__init__(name, age, "Мужчина")
        self.marital_status = marital_status
        self.number_of_children = number_of_children

    def __str__(self):
        return super().__str__() + f", Семейное положение:
{self.marital_status}, Количество детей: {self.number_of_children}"

class Woman(Person):
    def __init__(self, name, age, marital_status, number_of_children):
        super().__init__(name, age, "Женщина")
        self.marital_status = marital_status
        self.number_of_children = number_of_children

    def __str__(self):
        return super().__str__() + f", Семейное положение:
{self.marital_status}, Количество: {self.number_of_children}"

# Example usage
man1 = Man("John", 30, "Женат", 2)
woman1 = Woman("Jane", 25, "Замужем", 1)

print(man1)
print(woman1)
```

Протокол работы программы:
Имя: John, Возраст: 30, Пол: Мужчина, Семейное положение: Женат, Количество детей: 2
Имя: Jane, Возраст: 25, Пол: Женщина, Семейное положение: Замужем, Количество: 1

Process finished with exit code 0

Process finished with exit code 0

Вывод:

Оценив итоги выполнения этой задачи по работе с сохранением и загрузкой объектов с использованием библиотеки pickle, я улучшил свои навыки в создании и обработке структурированных данных. Полученные знания

позволят применять методы сериализации и десериализации в будущих проектах. Готовые программные коды выложены на GitHub.