

Практическое занятие №16

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

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

Задание 1.

Постановка задачи.

Создайте класс "Матрица", который имеет атрибуты количество строк и столбцов. Добавьте методы для сложения, вычитания и умножения матриц.

Тип алгоритма.

Линейный.

Текст программы.

```
import random

class Matrix:

    def __init__(self, rows, cols, data):

        self.rows = rows

        self.cols = cols

        self.data = data

    def add(self, other):

        if self.rows != other.rows or
self.cols != other.cols:

            raise ValueError("Matrix
dimensions do not match for addition")

        result = [[0 for _ in
range(self.cols)] for _ in range(self.rows)]

        for i in range(self.rows):

            for j in range(self.cols):

                result[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("Матрицы
несовместимы для вычитания.")

        result = [[0 for _ in
range(self.cols)] for _ in range(self.rows)]

        for i in range(self.rows):

            for j in range(self.cols):

                result[i][j] = self.data[i][j] -
other.data[i][j]

        return result

```

```

def mul(self, other):

    if self.cols != other.rows:

        raise ValueError("Матрицы
несовместимы для умножения.")

    result = [[0 for _ in
range(self.cols)] for _ in range(other.rows)]

    for i in range(self.cols):

        for j in range(other.rows):

            for k in range(self.rows):

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

    return result

```

```

matrix1 = [[random.randint(-5, 5) for i
in range(2)] for i in range(2)]

matrix2 = [[random.randint(-5, 5) for i
in range(2)] for i in range(2)]

```

```

print(f"Матрица 1: {matrix1}")

print(f"Матрица 2: {matrix2}")

```

```

matrix1 = Matrix(len(matrix1[0]),
len(matrix1), matrix1)

```

```

matrix2 = Matrix(len(matrix2[0]),
len(matrix2), matrix2)

res_add = matrix1.add(matrix2)

print(f"Сумма матриц: {res_add}")

res_sub = matrix1.sub(matrix2)

print(f"Разность матриц: {res_sub}")

res_mul = matrix1.mul(matrix2)

print(f"Произведение матриц:
{res_mul}")

```

Протокол работы программы.

Матрица 1: [[1, 4], [-5, 2]]
Матрица 2: [[-3, 0], [0, -1]]
Сумма матриц: [[-2, 4], [-5, 1]]
Разность матриц: [[4, 4], [-5, 3]]
Произведение матриц: [[-3, -4], [15, -2]]

Задание 2.

Постановка задачи.

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.

Тип алгоритма.

Линейный.

Текст программы.

```

class Vehicle:

    def __init__(self, max_speed, num_wheels):

        self.max_speed = max_speed

        self.num_wheels = num_wheels

class Car(Vehicle):

    def __init__(self, max_speed, num_wheels,
num_passengers):

        super().__init__(max_speed,
num_wheels)

        self.num_passengers = num_passengers

```

```
class Motorcycle(Vehicle):  
  
    def __init__(self, max_speed, num_wheels,  
engine_type):  
  
        super().__init__(max_speed,  
num_wheels)  
  
        self.engine_type = engine_type
```

```
car = Car(200, 4, 5)
```

```
motorcycle = Motorcycle(120, 2,  
"бензиновый")
```

```
print("Максимальная скорость  
автомобиля:", car.max_speed)
```

```
print("Количество колес мотоцикла:",  
motorcycle.num_wheels)
```

```
print("Тип двигателя мотоцикла:",  
motorcycle.engine_type)
```

Протокол работы программы.

Максимальная скорость автомобиля: 200
Количество колес мотоцикла: 2
Тип двигателя мотоцикла: бензиновый

Задание 3.

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.

Тип алгоритма.

Линейный.

Текст программы.

```
import pickle  
  
import random
```

```
class Matrix:  
  
    def __init__(self, rows, cols, data):  
  
        self.rows = rows
```

```
self.cols = cols
```

```
self.data = data
```

```
def add(self, other):
```

```
    if self.rows != other.rows or self.cols !=  
other.cols:
```

```
        raise ValueError("Матрицы  
несовместимы для сложения.")
```

```
    result = [[0 for _ in range(self.cols)] for _  
in range(self.rows)]
```

```
    for i in range(self.rows):
```

```
        for j in range(self.cols):
```

```
            result[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("Матрицы  
несовместимы для вычитания.")
```

```
    result = [[0 for _ in range(self.cols)] for _  
in range(self.rows)]
```

```
    for i in range(self.rows):
```

```
        for j in range(self.cols):
```

```
            result[i][j] = self.data[i][j] -  
other.data[i][j]
```

```
    return result
```

```
def mul(self, other):
```

```
    if self.cols != other.rows:
```

```
        raise ValueError("Матрицы  
несовместимы для умножения.")
```

```
    result = [[0 for _ in range(self.cols)] for _  
in range(other.rows)]
```

```
        for i in range(self.cols):
            for j in range(other.rows):
                for k in range(self.rows):
                    result[i][j] += self.data[i][k] *
other.data[k][j]
        return result
```

```
def save_def(matri, file):
    with open(file, 'wb') as f:
        pickle.dump(matri, f)
```

```
def load_def(file):
    with open(file, 'rb') as f:
        matr = pickle.load(f)

    return matr
```

```
matrix1 = [[random.randint(-5, 5) for i in
range(2)] for i in range(2)]
```

```
matrix2 = [[random.randint(-5, 5) for i in
range(2)] for i in range(2)]
```

```
matrix3 = [[random.randint(-5, 5) for i in
range(2)] for i in range(2)]
```

```
print(f"Матрица 1: {matrix1}")
```

```
print(f"Матрица 2: {matrix2}")
```

```
print(f"Матрица 3: {matrix3}")
```

```
matrix1 = Matrix(len(matrix1[0]),
len(matrix1), matrix1)
```

```
matrix2 = Matrix(len(matrix2[0]),
len(matrix2), matrix2)
```

```
matrix3 = Matrix(len(matrix3[0]),  
len(matrix3), matrix3)
```

```
matrix_info = [matrix1, matrix2, matrix3]
```

```
for mat in matrix_info:
```

```
    save_def(mat, 'matrixes.pkl')
```

```
    matrixes = load_def('matrixes.pkl')
```

```
    print(f"Сумма матриц:  
{matrixes.add(matrixes)}")
```

```
    print(f"Разность матриц:  
{matrixes.sub(matrixes)}")
```

```
    print(f"Произведение матриц:  
{matrixes.mul(matrixes)}")
```

Протокол работы программы.

Матрица 1: $\begin{bmatrix} -5 & -2 \\ -3 & 0 \end{bmatrix}$

Матрица 2: $\begin{bmatrix} -2 & 4 \\ -5 & 4 \end{bmatrix}$

Матрица 3: $\begin{bmatrix} 5 & -1 \\ -5 & -2 \end{bmatrix}$

Сумма матриц: $\begin{bmatrix} -10 & -4 \\ -6 & 0 \end{bmatrix}$

Разность матриц: $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Произведение матриц: $\begin{bmatrix} 31 & 10 \\ 15 & 6 \end{bmatrix}$

Сумма матриц: $\begin{bmatrix} -4 & 8 \\ -10 & 8 \end{bmatrix}$

Разность матриц: $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Произведение матриц: $\begin{bmatrix} -16 & 8 \\ -10 & -4 \end{bmatrix}$

Сумма матриц: $\begin{bmatrix} 10 & -2 \\ -10 & -4 \end{bmatrix}$

Разность матриц: $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Произведение матриц: $\begin{bmatrix} 30 & -3 \\ -15 & 9 \end{bmatrix}$

Вывод: Закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрел навыки составления программ с использованием ООП в IDE PyCharm Community.