Практическое занятие № 16

Тема: : составление программ с использованием ООП.

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

Постановка 1 задачи:
Тип алгоритма: линейный

Текст программы:

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

import random


class Matrix:
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols
        self.data = [[random.randint(1, 10) for _ in range(cols)] for _ in range(rows)]

    def __add__(self, other):
        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

    def __sub__(self, other):
        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

    def __mul__(self, other):
        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

    def __str__(self):
        return str(self.data)


a = Matrix(2, 3)
print(a)

b = Matrix(2, 3)
print(b)
```

```
print('сложение матириц : ', a + b)
print('вычитание матириц : ', a - b)
print('умножение матириц : ', a * b)
```

Протокол работы программы:

[[10, 2, 10], [2, 3, 7]]
[[7, 1, 4], [5, 9, 5]]
сложение матириц : [[17, 3, 14], [7, 12, 12]]
вычитание матириц : [[3, 1, 6], [-3, -6, 2]]
умножение матириц : [[70, 2, 40], [10, 27, 35]]

Process finished with exit code 0

Постановка 2 задачи:

Тип алгоритма: линейный

Текст программы:

```
class Transport:
    def __init__(self, max_speed, wheels, weight, max_people_on_ts,engine_size):
        self.max_speed = max_speed
        self.wheels = wheels
        self.weight = weight
        self.max_people_on_ts = max_people_on_ts
        self.engine_size = engine_size

    def info(self):
        return f"У этого ТС максимальная скорость = {self.max_speed} км/ч, транспорт имеет
{self.wheels} колеса, его вес составляет  { self.weight} кг " \
               f"он может перевезти максимум {self.max_people_on_ts} человек, обьем его
двигателя равен {self.engine_size} куб"


class Car(Transport):
    def __init__(self, max_speed , weight, engine_size):
        super().__init__(max_speed, 4, weight, 5, engine_size)



class Motorcycle(Transport):
    def __init__(self, max_speed, weight, engine_size):
        super().__init__(max_speed, 2, weight, 4, engine_size)



# Создание экземпляров классов
car = Car(250, 750, 650)
moto = Motorcycle(320, 150, 300)

# Вывод описания транспортных средств
```

```
print(car.info())
print(moto.info())
```

Протокол работы программы:
У этого ТС максимальная скорость = 250 км/ч, транспорт имеет 4 колеса, его вес составляет 750 кг он может перевезти максимум 5 человек, обьем его двигателя равен 650 куб
У этого ТС максимальная скорость = 320 км/ч, транспорт имеет 2 колеса, его вес составляет 150 кг он может перевезти максимум 4 человек, обьем его двигателя равен 300 куб

Process finished with exit code 0

Постановка 3 задачи:
Тип алгоритма: линейный

Текст программы:

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

import random
import pickle


class Matrix:
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols
        self.data = [[random.randint(1, 10) for _ in range(cols)] for _ in range(rows)]

    def __add__(self, other):
        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

    def __sub__(self, other):
        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

    def __mul__(self, other):
        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

    def save_def(self, nameFile):
        with open(nameFile, "wb") as f:
```

```
            pickle.dump(self, f)

    def load_def(self, nameFile):
        with open(nameFile, "rb") as f:
            inform = pickle.load(f)
        print(inform)

    def __str__(self):
        return str(self.data)


a = Matrix(2, 3)
print(a)

b = Matrix(2, 3)
print(b)

print('сложение матириц : ', a + b)
print('вычитание матириц : ', a - b)
print('умножение матириц : ', a * b)

a.save_def('text')
a.load_def('text')
```

Протокол работы программы:

[[1, 2, 5], [4, 1, 5]]
[[3, 9, 3], [9, 1, 3]]
сложение матириц : [[4, 11, 8], [13, 2, 8]]
вычитание матириц : [[-2, -7, 2], [-5, 0, 2]]
умножение матириц : [[3, 18, 15], [36, 1, 15]]
[[1, 2, 5], [4, 1, 5]]

Process finished with exit code 0

Вывод: мы закрепили усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрели навыки составление программ с ООП в IDE PyCharm Community.