Студент группы ИС-22 Ткаченко Р.А.

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 задачи.

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

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

    def __add__(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Матрицы должны иметь одинаковые размеры для
сложения.")
        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):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Матрицы должны иметь одинаковые размеры для
вычитания.")
        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):
        if self.rows != other.rows or self.cols != other.cols:
            raise ValueError("Матрицы должны иметь одинаковые размеры для
умножения.")
        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
```

```
if __name__ == '__main__':
    matrix_1 = Matrix(3, 3)
    matrix_2 = Matrix(3, 3)
    print(matrix_1)
    print()
    print(matrix_2)
    matrix_add = matrix_1 + matrix_2
    print()
    print(matrix_add)
    matrix_subtract = matrix_1 - matrix_2
    print()
    print(matrix_subtract)
    matrix_multiply = matrix_1 * matrix_2
    print()
    print(matrix_multiply)
```

Протокол работы программы:

6 9 8

5 4 6

1 1 5

7 2 10

2 5 5

5 10 6

13 11 18

7 9 11

6 11 11

-1 7 -2

3 -1 1

-4 -9 -1

42 18 80

10 20 30

5 10 30

Process finished with exit code 0

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.).

Тип алгоритма: линейный

Текст программы:

```
"""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.)."""


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

    def __str__(self):
        return f'Name: {self.name}, Age: {self.age}, Gender: {self.gender}'


class Man(Human):
    def __init__(self, name, age, gender, marital_status, children_count):
        super().__init__(name, age, gender)
        self.marital_status = marital_status
        self.children_count = children_count

    def __str__(self):
        return f'{super().__str__()}\nMarital Status: {self.marital_status},
Children Count: {self.children_count}'


class Woman(Human):
    def __init__(self, name, age, gender, marital_status, children_count):
        super().__init__(name, age, gender)
        self.marital_status = marital_status
        self.children_count = children_count

    def __str__(self):
        return f'{super().__str__()}\nMarital Status: {self.marital_status},
Children Count: {self.children_count}'


if __name__ == '__main__':
    man = Man('Maikl', 42, 'm', 'worker', 3)
    woman = Woman('Larisa', 50, 'f', 'worker', 2)
    print(man)
    print(woman)
```

Протокол работы программы:

Name: Maikl, Age: 42, Gender: m

Marital Status: worker, Children Count: 3

Name: Larisa, Age: 50, Gender: f

Marital Status: worker, Children Count: 2

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:

```
"""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."""

import pickle

from PZ_16_1 import Matrix


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


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


if __name__ == '__main__':
    file_name = 'data_matrix.bin'
    matrix_1 = Matrix(3, 3)
    matrix_2 = Matrix(3, 3)
    matrix_add = matrix_1 + matrix_2
    matrixs = [matrix_1, matrix_2, matrix_add]
    save_def(matrixs, file_name)
    matrixs = load_def(file_name)
    for matrix in matrixs:
        print(matrix)
        print()
    print(woman)
```

Протокол работы программы:

7 8 1

10 10 3

9 1 5

9 4 9

6 5 1

8 9 5

16 12 10

16 15 4

17 10 10

Process finished with exit code 0

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