Студент группы ИС-27 Якименко А.С

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

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

2) 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.)

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.

Тип алгоритма: разветвлённый

Текст программы

```
# Задача 1:
# Создайте класс «Матрица», который имеет атрибуты количества строк и
столбцов.
# Добавьте методы для сложения, вычитания и умножения матриц.
#
# Задача 2:
# 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.)
#
# Задача 3:
# Для задачи 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.
```

```python
import pickle

# Задача 1
class Matrix:
    def __init__(self, rows, cols, data=None):  # инициализация матриц
        self.rows = rows
        self.cols = cols
        self.data = data or [[0] * cols for _ in range(rows)]

    def __add__(self, other):  # добавление в матрицу
        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):  # вычитание с матрицы
        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):  # умножение матриц
        if self.cols != other.rows:
            raise ValueError("Количество столбцов в первой матрице должно
равняться количеству строк во второй")
        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][j] * other.data[k][j]
        return result

    def __str__(self):  # вывод в строковом формате
        return '\n'.join([' '.join(map(str, row)) for row in self.data])

# Задача 2
class Human:  # класс человека
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

class Man(Human):  # подкласс мужского пола
    def __init__(self, name, age, gender, marital_status=None,
number_of_children=0):
        super().__init__(name, age, gender)
        self.marital_status = marital_status
        self.number_of_children = number_of_children

class Woman(Human):  # подкласс женского пола
    def __init__(self, name, age, gender, marital_status=None,
number_of_children=0):
        super().__init__(name, age, gender)
        self.marital_status = marital_status
        self.number_of_children = number_of_children
```

```
# Задача 3
def save_def(filename, *matrices):
    with open(filename, 'wb') as file:
        pickle.dump(matrices, file)

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

# Задача 1 вывод
m1 = Matrix(2, 2, [[1, 2], [3, 4]])
m2 = Matrix(2, 2, [[5, 6], [7, 8]])
m3 = m1 + m2
m4 = m1 - m2
m5 = m1 * m2
print('Матрица 1')
print(m1)
print('')
print('Матрица 2')
print(m2)
print('')
print('сложение')
print(m3)
print('')
print('вычитание')
print(m4)
print('')
print('умножение')
print(m5)
print('--------------------')
# Задача 2 вывод
man = Man(name="Альберт", age=30, gender="Мужчина",
marital_status="Разведён", number_of_children=0)
woman = Woman(name="Яна", age=28, gender="Женщина", marital_status="В браке",
number_of_children=2)
print(f"Мужчина: {man.name}, Возраст: {man.age}, Семейное положение:
{man.marital_status}, Дети: {man.number_of_children}")
print(f"Женщина: {woman.name}, Возраст: {woman.age}, Семейное положение:
{woman.marital_status}, Дети: {woman.number_of_children}")
print('--------------------')

# Задача 3 вывод
save_def('matrices.pkl', m1, m2, m3, m4, m5)
loaded_matrices = load_def('matrices.pkl')
for m in loaded_matrices:
    print(m)
```

Протокол работы программы

Матрица 1

1 2

3 4

Матрица 2

5 6

7 8

сложение

6 8

10 12

вычитание

-4 -4

-4 -4

умножение

12 28

36 56

Мужчина: Альберт, Возраст: 30, Семейное положение: Разведён, Дети: 0

Женщина: Яна, Возраст: 28, Семейное положение: В браке, Дети: 2

---

1 2

3 4

5 6

7 8

6 8

10 12

-4 -4

-4 -4

12 28

36 56

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