УДК 371.31

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ МАТЕМАТИЧЕСКИХ ЗАДАЧ НА PYTHON

Л. К. СТОФОРАНДОВА, Т. В. КОРМИЛИЦЫНА
Мордовский государственный педагогический университет
имени М. Е. Евсевьева
Саранск, Россия

Объектно-ориентированное программирование (ООП) – это парадигма программирования, которая основывается на концепции классов и объектов. Оно используется для структурирования программного обеспечения в простые, многократно используемые фрагменты схем кода (обычно называемые классами), которые используются для создания отдельных экземпляров объектов.

При ООП программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов). Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создать модель сколь угодно сложной предметной области с заданными свойствами [1, 2].

Основная задача ООП – сделать сложный код проще. Для этого программу разбивают на независимые блоки, которые называются объектами. Процедурное программирование идеально работает в простых программах, где все задачи можно решить множеством функций.

Одной из наиболее мощных черт объектно-ориентированного языка является его способность предоставить программисту возможность создавать новые классы, моделирующие данные, необходимые для решения задачи. Особое место среди объектно-ориентированных сред занимает Python, однако его освоение представляет некоторые затруднения ввиду особенностей среды, однако именно эти особенности дадут ощутимые преимущества при решении задач. Поэтому следует рекомендовать среду для работы с одаренными студентами для подготовки к олимпиадам.

Проиллюстрируем преимущества Python на примерах реализации стандартных математических алгоритмов.

Для того чтобы сложить две дроби, их нужно привести к общему знаменателю. Простейший способ увериться, что у них одинаковый знаменатель, – это использовать в его качестве произведение знаменателей дробей. Можно использовать этот метод при написании стандартных арифметических выражений с дробями, присваивая результату суммарную дробь и выводя её на экран (рис. 1). Необходимо заметить, что результат может быть получен в виде рациональной

дроби, что невозможно без специальных ухищрений в других классических средах программирования.

```
def __add__(self,otherfraction):

    newnum = self.num*otherfraction.den + self.den*otherfraction.num
    newden = self.den * otherfraction.den

    return Fraction(newnum,newden)
```

```
>>> f1=Fraction(1,4)
>>> f2=Fraction(1,2)
>>> f3=f1+f2
>>> print(f3)
6/8
>>>
```

Рис. 1. Листинг кода для выражений с дробями

Для того чтобы быть уверенными, что результат всегда имеет сокращённый вид, понадобится вспомогательная функция, умеющая сокращать дроби. В ней нужно будет находить наибольший общий делитель (НОД). Затем можно разделить числитель и знаменатель на НОД, а результат и будет сокращением до наименьших членов.

Наиболее известный алгоритм нахождения наибольшего общего делителя – это алгоритм Евклида. Он устанавливает, что наибольшим общим делителем двух чисел m и n будет n, если m делится на n нацело. Однако если этого не происходит, то ответом будет НОД n и остатка деления m на n. Алгоритм на Python приведен на рис. 2.

Теперь можно использовать эту функцию для сокращения любой дроби. Чтобы представить дробь в сокращённом виде, следует делить числитель и знаменатель на их наибольший общий делитель.

Итак, для дроби $\frac{6}{8}$ НОД равен 2. Разделив верх и низ на 2, получим новую дробь $\frac{3}{4}$ (рис. 3). Следует провести вычислительные эксперименты [3].

Таким образом, преимущества ООП в программировании математических задач следующие:

– ООП моделирует сложные объекты как воспроизводимые, простые структуры;

– многократно используемые объекты ООП могут применяться в разных программах;

– легче отлаживать, классы часто содержат всю применимую к ним информацию.

```
def gcd(m,n):
    while m%n != 0:
        oldm = m
        oldn = n

        m = oldn
        n = oldm%oldn
    return n

print gcd(20,10)
```

```
10
```

Рис. 2. Алгоритм Евклида

```
def __add__(self,otherfraction):
    newnum = self.num*otherfraction.den + self.den*otherfraction.num
    newden = self.den * otherfraction.den
    common = gcd(newnum,newden)
    return Fraction(newnum//common,newden//common)
```

```
>>> f1=Fraction(1,4)
>>> f2=Fraction(1,2)
>>> f3=f1+f2
>>> print(f3)
3/4
>>>
```

Рис. 3. Экземпляр класса с двумя методами

Программирование математических задач на Python позволяет получать результаты в виде, совпадающем с математической нотацией.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. **Кормилицына, Т. В.** Построение компьютерных моделей для учебных экспериментов / Т. В. Кормилицына // Учебный эксперимент в образовании. – 2011. – № 2. – С. 44–49.

2. **Кормилицына, Т. В.** Составление алгоритмов как основа обучения решению задач по информатике / Т. В. Кормилицына // Учебный эксперимент в образовании. – 2018. – № 2. – С. 70–75.

3. **Стофорандова, Л. К.** Вычислительный эксперимент как средство формирования навыков анализа информации / Л. К. Стофорандова // Цифровая экономика: тенденции и перспективы развития в России и мире: материалы конф. – 2021. – С. 393–397.

УДК 371+22 19 (476)

РАЗНОУРОВНЕВЫЕ ЗАДАНИЯ В ПРОЦЕССЕ ПРЕПОДАВАНИЯ МАТЕМАТИЧЕСКИХ ДИСЦИПЛИН БУДУЩИМ ЭКОНОМИСТАМ

Л. П. ФАЛЬКО
Международный университет «МИТСО»
Минск, Беларусь

Основой осуществления полноценного процесса преподавания в высшем учебном заведении является система знаний и способов деятельности студентов, что определяет качество знаний (полнота, глубина, систематичность, гибкость, осознанность, действенность).

Согласно деятельностному подходу, диагностика уровня обученности студентов начинается с определения целей обучения [1, с. 89]. Классическая классификация учебных целей, или таксономия учебных целей, была разработана в 1956 г. учеными Чикагского университета под руководством психолога Бенджамина Блума. Классификация учебных целей [2, с. 4] содержит аспекты деятельности и указывает на уровни сформированности компетенций студентов в виде способов деятельности, в виде глаголов-действий.

По Блуму [3, с. 2–3] образовательные цели-действия состоят из трех областей:

1) когнитивной («знаю») – это знания, понимание и критическое мышление;

2) аффективной («чувствую») – это эмоциональное отношение человека к различным ситуациям, его ценности, интересы и склонности;

3) психомоторной («творю») – это практические навыки и умения.