САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР

ЛАНЬ

Т. П. НИКИТИНА,
Л. В. КОРОЛЕВ

ПРОГРАММИРОВАНИЕ. ОСНОВЫ PYTHON ДЛЯ ИНЖЕНЕРОВ

Учебное пособие

ЛАНЬ
· САНКТ-ПЕТЕРБУРГ · МОСКВА · КРАСНОДАР ·
2023

УДК 004.43
ББК 22.19я73

Н 62 **Никитина Т. П.** Программирование. Основы Python для инженеров : учебное пособие для вузов / Т. П. Никитина, Л. В. Королев. — Санкт-Петербург : Лань, 2023. — 156 с. : ил. — Текст : непосредственный.

ISBN 978-5-507-45284-2

Пособие посвящено рассмотрению базовых конструкций языка Python, в частности, сначала приведены примеры простейших программ в императивном стиле программирования и примеры решения несложных задач линейной, разветвляющейся и циклической структуры, задач с последовательностями и файлами. Далее дана реализация в виде программ на Python алгоритмов методов вычислительной математики. Большое внимание уделено практике использования библиотек numpy, matplotlib, pandas и turtle, для анализа данных и их графической интерпретации.

Учебное пособие предназначено для использования в учебном процессе студентами, обучающимися по направлениям подготовки «Технологические машины и оборудование», «Химия», «Энергетическое машиностроение», «Эксплуатация транспортно-технологических машин и комплексов» и других инженерных специальностей всех форм обучения при изучении дисциплин математического и естественнонаучного цикла. Учебное пособие разработано в соответствиями с требованиями Федерального государственного образовательного стандарта.

УДК 004.43
ББК 22.19я73

Рецензент

В. А. СОКОЛОВ — доктор физико-математических наук, профессор, зав. кафедрой теоретической информатики Ярославского государственного университета им. П. Г. Демидова.

Обложка
П. И. ПОЛЯКОВА

Введение

Язык программирования Python разработал голландец Гвидо ван Россум. Python — интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой. Встроенные высокоуровневые структуры данных в сочетании с динамической типизацией и связыванием делают язык привлекательным для быстрой разработки приложений (RAD, Rapid Application Development). Кроме того, его можно использовать в качестве сценарного языка для связи программных компонентов.

Интерпретатор выполняет инструкции построчно: после приглашения к работе записывается строка с необходимыми действиями, после нажатия клавиши Enter, интерпретатор выдает результат. Python можно использовать как калькулятор.

Другой вариант работы — работа в среде разработки IDLE (Integrated Development and Learning Environment).

Основные понятия и инструкции Python

Структура программы

Программы на языке Python состоят из инструкций и являются обычными текстовыми файлами, которые обычно имеют расширение *.py*. Эти файлы для просмотра и редактирования можно открывать с помощью любого текстового редактора, например программы Блокнот.

Правила записи инструкций.

- Конец строки является концом инструкции.
- Вложенные инструкции объединяются в блоки по величине отступов.
- Отступ может быть любым, главное, чтобы в пределах одного вложенного блока был одинаковый отступ. Не следует использовать отступ в один пробел, так как существенно снижается наглядность и восприятие человеком блочной структуры программы. Используйте четыре пробела (знак табуляции).
- Вложенные инструкции в Python записываются по следующему шаблону: основная инструкция завершается двоеточием, вслед за которым, чаще всего после нажатия клавиши Enter, располагается вложенный блок кода с необходимым отступом:

```
for i in range (0, n):
    if a[0][i]>0: k+=1
    if a[n-1] [i]>0: k+=1
```

- Тело составной инструкции может располагаться в той же строке, что и тело основной, если тело составной инструкции не содержит составных инструкций:

```
if x > y: print(x)
```

- Можно записать несколько инструкций в одной строке, разделяя их точкой с запятой:

```
a = 1; b = 2; print (a, b)
```

- Допустимо записывать одну инструкцию на нескольких строках. Достаточно ее заключить в пару круглых, квадратных или фигурных скобок:

```
if (a == 1 and b == 2 and
c == 3 and d == 4): # Не забываем про двоеточие
print ('if занимает две строки')
```

Имена переменных

Каждый объект программы должен иметь идентификатор, задаваемый пользователем. На основе идентификатора строится имя объекта, которое позволяет обращаться как ко всему объекту, так и к отдельным его составляющим.

Синтаксические конструкции Python записывают с использованием уникальных ключевых слов, которые нельзя использовать в качестве идентификаторов.

Список ключевых слов можно получить с помощью следующих инструкций:

```
import keyword
print(keyword.kwlist)
```

Результат

```
['and', 'assert', 'break', 'class', 'continue', 'def', 'del',
'elif', 'else', 'except', 'exec', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or',
'pass', 'print', 'raise', 'return', 'try', 'while', 'yield']
```

Переменную можно связать с объектом в любом месте блока, важно, чтобы это произошло до ее использования, иначе появится синтаксическая ошибка *NameError*. В частности, связывание имен со значениями происходит в инструкциях присваивания.

Правила записи.

- Всегда следует связывать переменную со значением до ее использования.
- Необходимо избегать глобальных переменных и передавать все необходимые данные через параметры.
- Убрать связь имени с объектом можно с помощью оператора *del*. В этом случае, если объект не имеет других ссылок на него, он будет удален. Для управления памятью в Python используется подсчет ссылок, для удаления наборов объектов с зацикленными ссылками — сборка мусора (*garbage collection*).

В каждой точке программы интерпретатор «видит» три пространства имен: локальное, глобальное и встроенное. Пространство имен связано с понятием блока кода. В Python блоком кода является то, что исполняется как единое целое, например тело цикла, функции, условной инструкции.

Локальные имена — имена, которым присвоено значение в блоке, доступны только в нем. Глобальные имена — имена, определяемые на уровне блока модуля или те, которые явно заданы как *global*. Встроенные имена — имена из специального словаря *builtins*.

Области видимости имен могут быть вложенными друг в друга, например, внутри вызванной функции видны имена, определенные в вызывающем коде. Переменные, которые используются в блоке кода, но связаны со значением вне кода, называются свободными переменными.

Константы и переменные

Данные представлены константами и переменными. Все данные в Python представляют собой объекты.

Каждый объект содержит как минимум три вида данных:

- счетчик ссылок — используется для управления памятью;
- тип;
- значение.

Python — это язык программирования с динамической типизацией, то есть в ходе выполнения программы одна и та же переменная может хранить значения различных типов. Типы данных можно разделить на встроенные в интерпретатор и не встроенные, которые можно использовать при импортировании соответствующих модулей.

В языке Python, как и в других языках программирования, например C++, различают неизменяемые (константные) и изменяемые типы данных. Основное различие при работе с ними заключается в том, что для данных с неизменяемым типом запрещены инструкции, меняющие значение объекта.

К основным встроенным типам относятся следующие.

- *None* (неопределенное значение переменной).
- Логические значения:
  - *True*;
  - *False*.
- Числа (неизменяемые типы):
  - *int* — целое число;
  - *float* — вещественное число;
  - *complex* — комплексное число.
- Списки:
  - *list* — список;
  - *tuple* — кортеж (неизменяемый тип);
  - *range* — диапазон (неизменяемый тип).
- Строки *str* (неизменяемый тип).

Арифметические константы могут быть представлены в программе своими значениями (явно):

- целые числа 4 687 –45 0;
- вещественные значения:
  - с фиксированной точкой 1.45 –3.789654 0.00453;