Д. Ю. Федоров

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ PYTHON

УЧЕБНОЕ ПОСОБИЕ
ДЛЯ ПРИКЛАДНОГО БАКАЛАВРИАТА

Рекомендовано Учебно–методическим отделом высшего образования в качестве учебного пособия для студентов высших учебных заведений, обучающихся по инженерно–техническим направлениям

Книга доступна в электронной библиотечной системе biblio-online.ru

Москва ▪ Юрайт ▪ 2019

УДК 004.42(075.8)
ББК 32.973-018я73
Ф33

***Автор*:**

Федоров Дмитрий Юрьевич — старший преподаватель кафедры вычислительных систем и программирования факультета информатики и прикладной математики Санкт-Петербургского государственного экономического университета.

***Рецензенты*:**

Чудаков О. Е. — профессор, доктор технических наук, профессор кафедры специальных информационных технологий Санкт-Петербургского университета МВД России;

Трофимов В. В. — доктор технических наук, профессор, заслуженный деятель науки Российской Федерации, заведующий кафедрой информатики Санкт-Петербургского государственного экономического университета.

Федоров, Д. Ю.

Ф33 Программирование на языке высокого уровня Python : учеб. пособие для прикладного бакалавриата / Д. Ю. Федоров. — М. : Издательство Юрайт, 2019. — 126 с. — (Серия : Бакалавр. Прикладной курс).

ISBN 978-5-534-04479-9

В учебном пособии рассматриваются теоретические основы современных технологий и методов программирования, практические вопросы создания программ, а также основные алгоритмические конструкции и их реализация на языке высокого уровня Python.

Соответствует актуальным требованиям Федерального государственного образовательного стандарта высшего образования

Для студентов высших учебных заведений, обучающихся по инженерно-техническим направлениям.

УДК 004.42(075.8)
ББК 32.973-018я73

ISBN 978-5-534-04479-9

Оглавление

Предисловие

Современное общество является информационным, в нем все большее число людей занято получением, переработкой и использованием информации с применением компьютерных технологий. Компьютеры используются практически во всех областях человеческой деятельности: в профессиональной, учебной, досуговой сферах.

Дальнейшее развитие информационного общества требует разработки большого количества качественных программных продуктов, обеспечивающих удовлетворение растущих потребностей людей. В этих условиях весьма актуальным становится овладение современными технологиями программирования. Исследования в области управления персоналом показывают, что профессия программиста будет одной из самых востребованных в XXI в.

Целью и задачами учебного пособия «Программирование на языке высокого уровня Python» является систематизация и упорядочение сведений о технологиях разработки современных программных продуктов. В книге рассматриваются основные алгоритмические конструкции и их реализация на языке высокого уровня Python. Рассмотрение теоретических основ программирования сопровождается большим количеством примеров, иллюстрирующих приемы создания программ, а также заданиями для самостоятельного выполнения, позволяющими сформировать у обучающихся практические навыки программирования.

Пособие предназначено для студентов, обучающихся по программам бакалавриата и магистратуры, а также для всех, интересующихся современными технологиями программирования.

В результате изучения материалов пособия обучающиеся должны:

знать

- структуру программы;
- основные типы данных, их особенности;
- стандартные модули языка;

уметь

- выполнять стандартные операции над данными различного типа;

• применять стандартные алгоритмические структуры для обработки данных;

владеть

• основными принципами объектно-ориентированного подхода при создании программ;

• спецификой работы с переменными различных типов данных.

Глава 1
ЗНАКОМСТВО С ЯЗЫКОМ ПРОГРАММИРОВАНИЯ PYTHON

Python — высокоуровневый язык программирования общего назначения с динамической типизацией, автоматическим управлением памятью, поддержкой многопоточных вычислений и удобными структурами данных. На рис. 1.1 перечислены области применения языка программирования Python.

Прежде чем переходить к выполнению программ на языке Python, рассмотрим, как запускаются программы на компьютере (рис. 1.2). Выполнение программ осуществляется операционной системой (Microsoft Windows, GNU/Linux и пр.). В задачи операционной системы входит выделение ресурсов (оперативной памяти и пр.) для программы, запрет или разрешение на доступ к устройствам ввода/вывода и т.д.

Для запуска программ, написанных на языке программирования Python, необходима программа-интерпретатор[1] (*виртуальная*

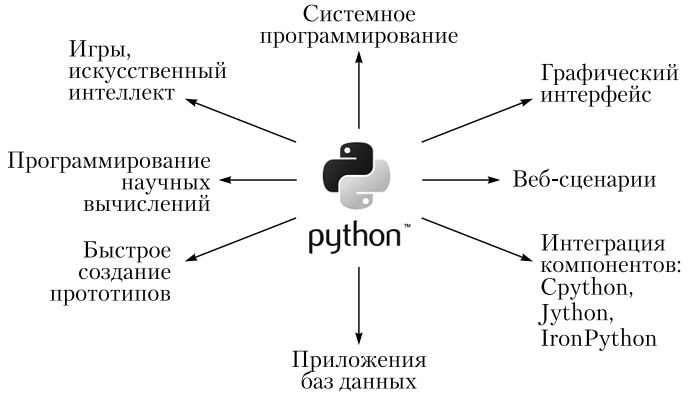

Рис. 1.1. **Области применения языка программирования Python**

[1] Python является интерпретируемым языком программирования (команды выполняются шаг за шагом), в отличие от компилируемых, где текст программы переводится в эквивалентный машинный код.

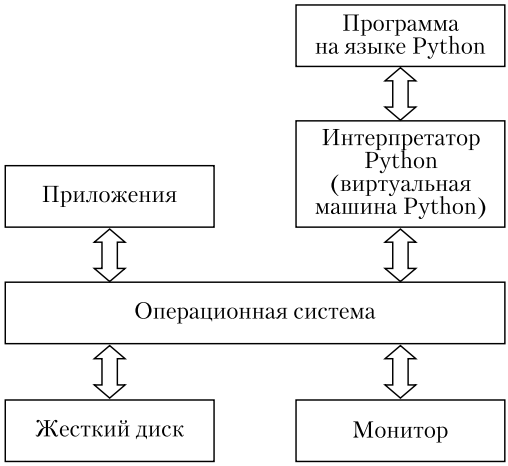

Рис. 1.2. **Схема запуска программ**

машина) Python. Данная программа скрывает от Python-программиста все особенности операционной системы, поэтому, написав программу на Python в системе Windows, ее можно запустить, например, в GNU/Linux и получить схожий результат.

Скачать и установить интерпретатор Python[1] можно совершенно бесплатно с официального сайта http://python.org. Для работы нам понадобится интерпретатор Python версии 3.6 или выше. В процессе установки рекомендуется указать путь **C:/Python36-32**.

После установки программы-интерпретатора запустите интерактивную графическую среду IDLE[2] и дождитесь появления приглашения для ввода команд:

```
Type "copyright", "credits" or "license()" for more
information.
>>>
```

---

[1] Процесс установки зависит от операционной системы.

[2] Выбор среды IDLE обусловлен тем, что она входит в стандартную поставку интерпретатора Python и является свободно распространяемой.

Глава 2
ИНТЕЛЛЕКТУАЛЬНЫЙ КАЛЬКУЛЯТОР

В самом начале обучения Python можно рассматривать как интерактивный интеллектуальный калькулятор. В интерактивном режиме IDLE найдем значения математических выражений. После завершения набора выражения нажмите клавишу <*Enter*> для завершения ввода и последующего выполнения указанных выражений:

```
>>> 3.0 + 6
9.0
>>> 4 + 9
13
>>> 1 - 5
-4
>>> _ + 6
2
```

Нижним подчеркиванием в предыдущем примере обозначается последний полученный результат.

Если совершить ошибку при вводе команды, то интерпретатор Python сообщит об этом:

```
>>> a
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    a
NameError: name 'a' is not defined
```

В математических выражениях в качестве операндов могут использоваться как *целые числа*[1] (1, 4, –5), так и *вещественные*[2] (в программировании их еще называют *числами с плавающей точкой*): 4.111, –9.3. Математические операции, доступные над числами в Python[3], представлены в табл. 2.1.

---

[1] А также комплексные числа и логические значения (**True**, **False**).

[2] Объяснение правил хранения вещественных чисел в компьютере выходит за рамки учебника, сравните в следующем примере результаты вычислений:

```
>>> 2/3 + 1
1.6666666666666665
>>> 5/3
1.6666666666666667
>>>
```

[3] Любопытно, что в Python выражение **(b * (a // b) + a % b)** эквивалентно **a**.

Таблица 2.1

Математические операторы в Python

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление (в результате вещественное число)
//	Деление с округлением вниз
**	Возведение в степень
%	Остаток от деления

```
>>> 5/3
1.6666666666666667
>>> 5 // 3
1
>>> 5 % 3
2
>>> 5 ** 67
677626357803440271254658000543713569641113281 25
```

Если один из операндов является вещественным числом, то в результате вычислений получится вещественное число.

При вычислении математических выражений Python придерживается приоритета операций (следует математическим соглашениям):

```
>>> -2 ** 4
-16
>>> -(2**4)
-16
>>> (-2) ** 4
16
```

В случае сомнений в порядке применения математических операторов и для упрощения чтения выражений будет полезным обозначить приоритет в виде круглых скобок.

Выражаясь в терминах программирования, только что мы познакомились с *числовым типом данных* (целочисленным типом **int** и вещественным типом **float**), т.е. множеством числовых значений и множеством математических операций, которые можно выполнять над данными значениям.

Язык программирования Python предоставляет большой выбор встроенных типов данных, о которых речь пойдет дальше.

Глава 3
ПЕРЕМЕННЫЕ

Рассмотрим выражение **y = x + 3 * 6**, где **y** и **x** являются *переменными*, которые могут содержать некоторые значения. На языке Python вычислить значение **y** при **x** равном 1 можно следующим образом:

```
>>> x = 1
>>> y = x + 3 * 6
>>> y
19
```

В выражении нельзя использовать переменную, если ранее ей не было присвоено значение с помощью *инструкции присваивания*. Для Python такие переменные не определены, и их использование приведет к ошибке.

Содержимое переменной **y** можно вывести на экран, если в интерактивном режиме ввести ее имя.

Имена переменным задает программист, но есть несколько ограничений, связанных с их наименованием. Имена переменных нельзя начинать с цифры и в качестве имен переменных нельзя использовать ключевые слова, которые для Python имеют определенный смысл (эти слова подсвечиваются в IDLE оранжевым цветом, табл. 3.1).

Таблица 3.1

Ключевые слова Python

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None	—	—

Далее мы часто будем обращаться к формуле перевода из шкалы в градусах по Цельсию (T_C) в шкалу в градусах по Фаренгейту (T_F):

```
TF = 9/5 * TC + 32
```

Определим значение T_F при T_C, равном 26. Создадим переменную с именем **cel**, содержащую значение целочисленного типа 26:

```
>>> cel = 26
>>> cel
26
>>> 9/5 * cel + 32
78.80000000000001
```

В момент выполнения инструкции присваивания **cel** = 26 в памяти компьютера создается *объект* (рис. 3.1), расположенный по некоторому *адресу* (условно обозначим его как *id*1), имеющий значение 26 целочисленного типа **int**. Затем создается переменная с именем **cel**, которой *присваивается адрес* объекта *id*1.

Таким образом, переменные в Python содержат адреса объектов. Иначе можно сказать, что *переменные ссылаются на объекты*. Тип переменной определяется типом объекта, на который она ссылается. В дальнейшем для упрощения будем говорить, что переменная хранит значение.

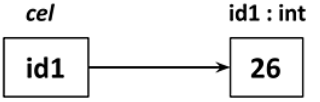

Рис. 3.1. **Модель памяти для выражения cel = 26**[1]

Вычисление следующего выражения приведет к присваиванию переменной **cel** значения 72, т.е. сначала вычисляется правая часть, затем результат присваивается левой части:

```
>>> cel = 26 + 46
>>> cel
72
```

Рассмотрим более сложный пример:

```
>>> diff = 20
>>> double = 2 * diff
>>> double
40
```

Во втором выражении в первую очередь произойдет вычисление правой части, где на место переменной **diff** подставится значение 20, и результат вычисления присвоится переменной **double**. По окончании вычислений память будет иметь вид, представленный на рис. 3.2.

---

[1] Рисунки к учебному пособию даны по изданию: *Федоров Д. Ю.* Основы программирования на примере языка Python : учеб. пособие. СПб., 2016.

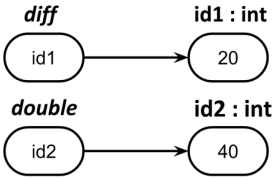

Рис. 3.2. **Схема памяти Python при работе с переменными**

Далее присвоим переменной **diff** значение 5 и выведем на экран содержимое переменных **double** и **diff**:

```
>>> diff = 5
>>> double
40
>>> diff
5
```

В момент присваивания переменной **diff** значения 5 в памяти (рис. 3.3) создастся объект по адресу *id*3, содержащий целочисленное значение 5. После этого изменится содержимое переменной **diff**, вместо адреса *id*1 туда запишется адрес *id*3. Также Python увидит, что на объект по адресу *id*1 больше никто не ссылается и поэтому удалит его из памяти (произведет автоматическую *сборку мусора*).

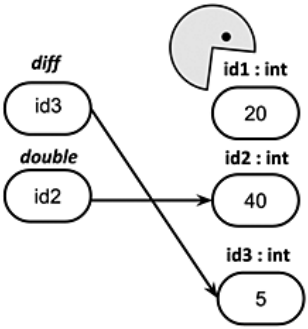

Рис. 3.3. **Схема памяти Python при работе с переменными**

Внимательный читатель заметил, что Python не изменяет существующие числовые объекты, а создает новые, т.е. объекты числового типа данных в Python являются *неизменяемыми*.

У начинающих программистов часто возникает недоумение при виде следующих выражений:

```
>>> num = 20
>>> num = num * 3
>>> num
60
```

Если вспомнить, что сначала вычисляется правая часть выражения, то все станет на свои места.

Глава 4
ФУНКЦИИ

Функцией в программировании называется последовательность инструкций, которая выполняет вычисления. С чем можно сравнить функцию? Напрашивается аналогия с «черным ящиком», когда мы знаем, что поступает на вход и что при этом получается на выходе, а внутренности «черного ящика» от нас скрыты. В качестве примера можно привести банкомат. На вход банкомата поступает пластиковая карточка (пин-код, денежная сумма), на выходе мы ожидаем получить запрашиваемую сумму. Нас не очень сильно интересует принцип работы банкомата до тех пор, пока он работает без сбоев.

Рассмотрим встроенную функцию с именем **abs**, принимающую на вход один аргумент — объект числового типа, и возвращающую абсолютное значение для этого объекта (рис. 4.1).

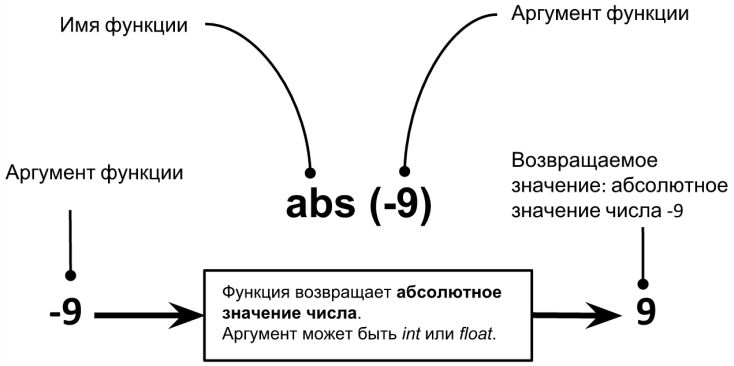

Рис. 4.1. **Функция как «черный ящик»**

Пример вызова функции **abs** с *аргументом* –9 имеет вид:

```
>>> abs(-9)
9
>>> d = 1
>>> n = 3
>>> abs(d - n)
2
```

```
>>> abs(-9) + abs(5.6)
14.6
```

Результат вызова функции можно присвоить переменной, использовать его в качестве операндов математических выражений, что позволяет формировать более сложные выражения.

Рассмотрим примеры нескольких встроенных математических функций.

Функция **pow(x, y)** возвращает значение **x** в степени **y**. Эквивалентно записи **x**y**, с которой мы уже встречались.

```
>>> pow(4, 5)
1024
```

Функция **pow** может принимать третий аргумент, тогда вызов функции с аргументами **pow(x, y, z)** эквивалентен вычислению выражения **(x ** y) % z**:

```
>>> pow(4, 5, 3)
1
```

Функция **round(number)** возвращает число с плавающей точкой, округленное до 0 цифр после запятой (по умолчанию). Функция может быть вызвана с двумя аргументами: **round(number [, ndigits])**, где **ndigits** — число знаков после запятой:

```
>>> round(4.56666)
5
>>> round(4.56666, 3)
4.567
```

Помимо составления сложных математических выражений Python позволяет передавать результат вызова функции в качестве аргументов других функций без использования дополнительных переменных.

На рис. 4.2 представлен пример вызова функций и порядок их вычисления. В этом примере на месте числовых объектов –2, 4.3 могут находиться более сложные выражения, поэтому они также нуждаются в вычислении.

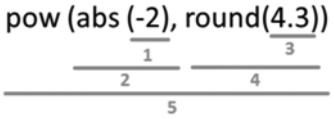

Рис. 4.2. **Порядок вычисления составного выражения**

На практике часто при написании программ требуется преобразовывать типы объектов.

Функция **int** принимает любое значение и преобразует его в целое число, если это возможно (возвращает 0, если аргументы не переданы):