Оглавление

Предисловие

Ни для кого не секрет, что роботы все чаще встречаются в нашей повседневной жизни. Роботы развлекают детей, летают в космос, помогают спасателям, управляют автомобилями и боевой техникой. С каждым поколением они становятся все совершеннее. О новых разработках в робототехнике мы узнаем с экранов телевизоров и на сайтах в Интернете, со страниц журналов и книг. Современные роботы способны видеть, распознавать объекты окружающего их мира и выполнять сложные операции. Более того, современные роботы способны обучаться, адаптируясь к внешней среде. Поведение таких роботов очень близко к поведению живых существ.

Значимой составляющей в подобных роботах является система компьютерного зрения. Эта технология позволяет роботам с помощью специальных алгоритмов и программ анализировать поток данных, который они получают с видеокамеры. Главными задачами компьютерного зрения является обнаружение, идентификация и классификация объектов окружающего мира, которые попали в поле зрения видеокамеры. Развитые системы компьютерного зрения могут распознавать различные активности, которые проявляют обнаруженные объекты.

До недавнего времени изучение, проектирование и создание устройств, оснащенных простейшим компьютерным зрением, потребовали бы от разработчика больших интеллектуальных и материальных ресурсов. Но прогресс не стоит на месте. С появлением недорогих компактных микрокомпьютеров, а также доступного программного обеспечения изучение и программирование несложных систем с компьютерным зрением стало возможно даже для школьников.

Книга, которую вы держите в руках, ориентирована на читателей, начинающих изучать компьютерное зрение на языке Python. Весь материал разбит на три главы и снабжен иллюстрациями, ссылками на источники и программными кодами. После каждой главы приводится список литературы и интернет-ресурсов, которые могут быть полезны для дополнительного изучения.

Первая глава представляет собой краткое введение в программирование на языке Python. В настоящее время доступно множество различных источников и интернет-ресурсов, с помощью которых можно освоить основы программирования на Python, поэтому здесь этот вопрос не рассматривается подробно. Начальные навыки программирования в Python довольно легко освоить самостоятельно. Синтаксис языка Python прост и понятен, а его особенности можно постичь в процессе практики.

Вторая глава посвящена ключевой теме книги — изучению основ компьютерного зрения. Одним из самых мощных и доступных инструментов программирования компьютерного зрения на Python является библиотека **OpenCV (Open Source Computer Vision Library)** — библиотека компьютерного зрения с открытым исходным кодом. Наиболее важные, с точки зрения автора, вопросы, первоначально необходимые для работы с OpenCV, изложены в этой главе. Весь материал главы носит прикладной характер и условно разбит на пять тем:

1) установка OpenCV;
2) основные операции в OpenCV;
3) выделение объекта по цветовой маске;
4) работа с видеопотоком;
5) контурный анализ.

В третьей главе рассмотрены вопросы, связанные с подготовкой микрокомпьютера Raspberry Pi 3 к работе, установкой на него библиотеки OpenCV и программированием движения робоплатформы, оснащенной видеокамерой, вдоль черной линии. На сегодняшний день для реализации систем с компьютерным зрением самой подходящей по многим критериям является, пожалуй, техническая платформа на основе семейства **микрокомпьютеров Raspberry Pi**. Вычислительных ресурсов Raspberry Pi хватает для большинства задач начального уровня.

Предлагаемая книга не является исчерпывающим пособием по изучению компьютерного зрения с помощью OpenCV-Python. Здесь в сжатой форме собран материал, который будет полезным для начинающих изучать язык программирования Python и основы компьютерного зрения с библиотекой OpenCV-Python.

Итак, давайте окунемся в интересный мир программирования и робототехники! Вместе разберемся с принципами, на которых реализовано компьютерное зрение роботов. Настроим свой первый микрокомпьютер и научим его управлять робоплатформой по картинке с видеокамеры.

1

Начало программирования на Python

О языке Python

Python — это интерпретируемый язык программирования. Исходный код преобразуется в машинный частями по мере его выполнения специальной программой, которая называется **интерпретатор**. Язык Python придумал голландец Гвидо ван Россум примерно в 1991 г.

После того как Россум разработал язык Python и выложил его в Интернет, появилось целое сообщество программистов, которые активно его улучшают и совершенствуют. Новые версии языка регулярно выходят на его официальном сайте[1] [1].

Python отличается своим уникальным синтаксисом. Даже начинающий программист может легко научиться кодировать на этом языке программирования. Одной из особенностей языка является его требовательность к отступам, заставляющая программистов воспитывать в себе культуру написания программных кодов. В результате программный текст получается структурированным, легко читается и понимается. Добавим еще, что Python — это полноценный универсальный объектно-ориентированный язык программирования.

ПОЯСНЕНИЕ

Технология объектно-ориентированного программирования[2] основана на представлении программы в виде взаимодействия не-

---

[1] Python Software Fundation. // Официальный сайт разработчика Python. URL: **http://python.org**

[2] Объектно-ориентированное программирование. // Викиучебник. URL: **https://ru.wikibooks.org/wiki/Объектно-ориентированное_программирование**

которого числа **объектов**. При этом каждый **объект** принадлежит к конкретному **классу** и обладает характерным для этого класса набором **свойств** и **методов**. Например, Шарик и Бобик являются объектами, которые взаимодействуют между собой (лают), при этом оба относятся к классу «Собаки». Следовательно, они, как и все собаки, имеют четыре лапы, хвост и умеют лаять. А Барсик является объектом, который принадлежит к классу «Кошки». Он тоже имеет четыре лапы и хвост, но в отличие от собак умеет мяукать. И в то же время оба класса «Собаки» и «Кошки» имеют общие свойства: четыре лапы и хвост, которые были унаследованы от более широкого класса «Домашние животные». К объектно-ориентированным языкам относятся языки C/C++, Jawa, Object Pascal.

Кроме того, Python является свободно распространяемым языком на основании лицензии, совместимой с GNU General Public License.

Кроссплатформенные возможности Python позволяют программировать на этом языке под различными операционными системами, например Windows, Linux, Mac OS и др. Это добавляет привлекательности Python среди программистов.

Установка Python и дополнительных библиотек

Установка Python не представляет особых трудностей. Предварительно необходимо скачать установщик Python с официального сайта [1]. В целях безопасности ваших данных на компьютере не рекомендуется скачивать Python из сторонних ресурсов. Итак, заходим на сайт Python и в меню *Downloads* скачиваем последнюю актуальную версию для своей операционной системы (**рис. 1.1**). В нашем случае это будет операционная система Windows.

После окончания загрузки следует открыть папку (по умолчанию установочный файл сохранится в папке *Загрузки*) с загруженным исполняемым файлом и запустить его. В диалоговом окне установщика (**рис. 1.2**) следует поставить галочку напротив пункта *Add Python 3.7 to PATH*. Затем ждем завершение процесса установки. По окончании установки Python запускается из меню *Пуск*. Более подробные инструкции по установке, в том числе и на другие операционные системы, можно посмотреть здесь[1] [2, 3].

[1] Python. Урок 1. Установка. // Devpractice. Разработка программного обеспечения, технологии и наука. URL: **https://devpractice.ru/python-lesson-1-install/**

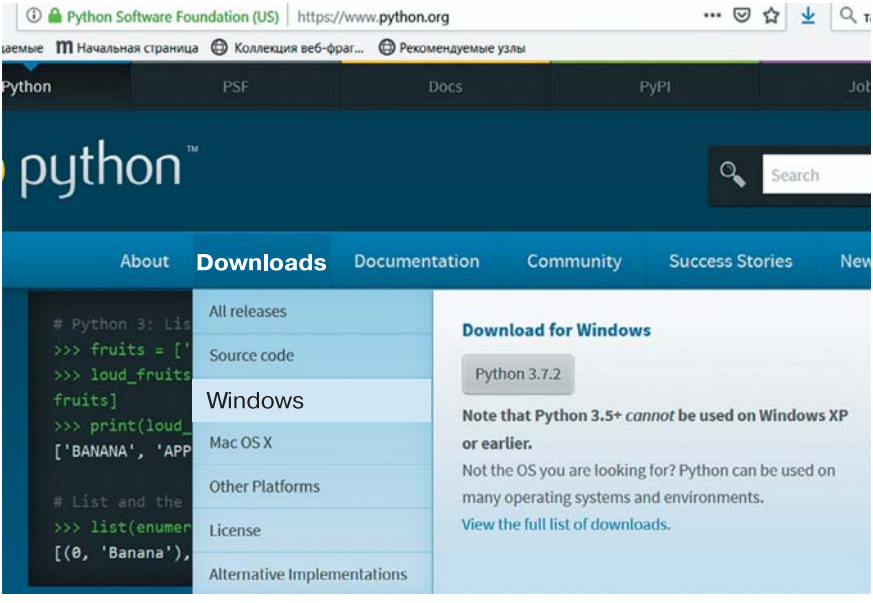

Рис. 1.1. Меню загрузки Python для операционной системы Windows

Для работы в Python, кроме стандартных библиотек и пакетов, иногда требуется установить дополнительные специализированные библиотеки. В настоящее время под различные задачи разработано огромное количество библиотек и пакетов для Python. Чтобы установить дополнительные пакеты и библиотеки, используют утилиту **pip**.

Рис. 1.2. Диалоговое окно установщика с выбором параметров установки языка Python

ПОЯСНЕНИЕ

pip — система управления пакетами, которая используется для установки и управления программными пакетами и библиотеками, написанными на Python. Начиная с версии Python 2.7.9 и Python 3.4, дистрибутив Python по умолчанию содержит утилиту управления пакетами **pip** (или **pip3** для Python 3).

С помощью **pip**, используя соединение с Интернетом, можно легко установить практически любой дополнительный пакет или библиотеку. Работа с **pip** осуществляется в командной строке Windows (**рис. 1.3**).

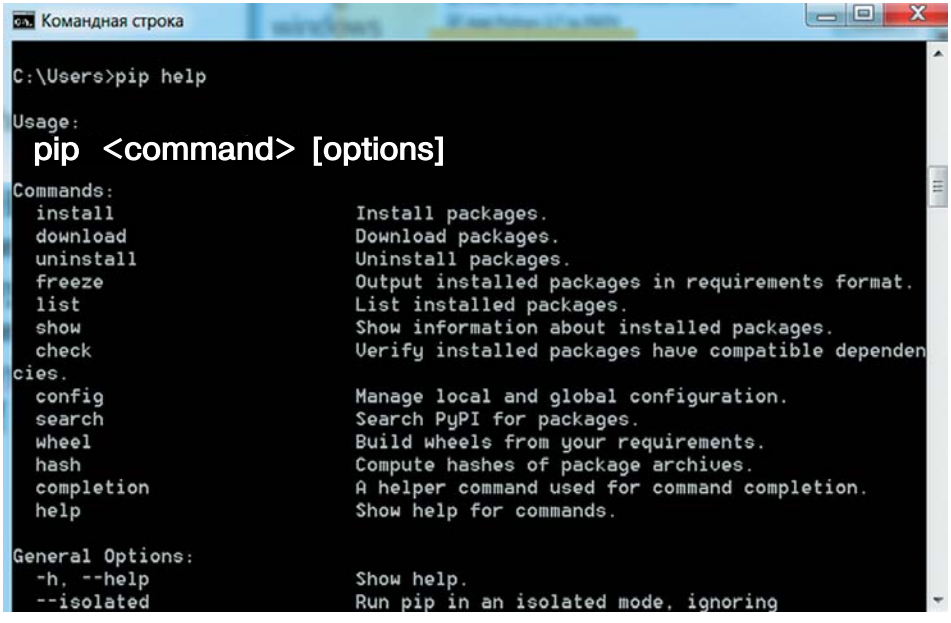

Рис. 1.3. Командная строка Windows

Установка нового пакета производится следующим образом. Например, чтобы установить пакет **numpy**, необходимо ввести в командной строке:

```
pip install numpy
```

или для Python 3:

```
pip3 install numpy
```

В Linux установка этого же пакета (подробнее об этом будет в главе 3) выполняется с помощью команды от имени администратора:

```
sudo pip3 install numpy
```

ПОЯСНЕНИЕ

Numpy — это пакет языка Python, предназначенный для работы с большими многомерными массивами и матрицами. Содержит в себе большую библиотеку высокоуровневых (и очень быстрых) математических функций для операций с этими массивами[1, 2]. В дальнейшем этот пакет нам понадобится для работы с компьютерным зрением. Данный пакет является обязательным компонентом для работы с OpenCV.

Утилита **pip** выполнит загрузку пакета и распакует его в специальную папку с установленным Python. Как видите, установка дополнительного пакета с помощью **pip** не представляет большого труда. Полная документация по работе с **pip** представлена на сайте[3] [4]. Ниже приведены основные команды **pip**, где `<package_name>` — название устанавливаемого вами пакета:

`pip help` — помощь по доступным командам;
`pip install <package_name>` — установка пакета(ов);
`pip uninstall <package_name>` — удаление пакета(ов);
`pip list` — список установленных пакетов;
`pip show <package_name>` — показывает информацию об установленном пакете;
`pip search` — поиск пакетов по имени;
`pip install-U` — обновление пакета(ов);
`pip install--force-reinstall` — при обновлении переустановить пакет, даже если он последней версии.

---

1 Документация по numpy. URL: **https://docs.scipy.org/doc/numpy/reference/**

2 Numpy для начинающих. URL: **https://pythonworld.ru/numpy**

3 pip — The Python Package Installer. **URL: https://pip.pypa.io/en/stable/**

Более подробно с установкой и удалением дополнительных пакетов, в том числе и без использования утилиты **pip**, можно ознакомиться в [5–7].

Программирование в Python предусматривает два режима: консольный и интерактивный. В первом случае в окне будет высвечено приглашение к вводу команды (>>>). В этом режиме интерпретатор исполняет введенную команду сразу после нажатия клавиши **enter**. Другой режим — это работа во встроенной среде разработки IDLE, в которой тоже есть интерактивный режим работы. Но в отличие от консольного варианта здесь можно писать полноценные программы и наблюдать подсветку синтаксиса (в зависимости от значения синтаксической единицы она выделяется определенным цветом[1]). Программы на Python часто называют **скриптами**, которые с помощью среды IDLE можно сохранять в файлы с расширением **.py**.

Чтобы создать скрипт в среде IDLE, после запуска программы в меню следует выбрать команду *File* → *New Window* (Crtl + N). В результате откроется новое окно, в котором пишется программный код. И наконец, чтобы его запустить для исполнения, необходимо выполнить команду меню *Run* → *Run Module* (F5). После этого в консольном окне Python появится результат выполнения кода (**рис. 1.4**).

Рис. 1.4. Окно программного кода (слева) и консольное окно (справа)

Синтаксис языка Python

Программирование на Python с самого начала приучает начинающего программиста писать правильный и удобочитаемый код (соблюдать принципы, называемые общим словом **readability**). В отличие от других языков в Python значение имеют не только лексемы языка.

---

[1] Если набирать код, не сохранив сначала файл, то подсветка синтаксиса будет отсутствовать.

ПОЯСНЕНИЕ

Лексема — это структурная единица языка программирования, которая состоит только из разрешенных символов алфавита языка и не содержит в своем составе другие структурные единицы языка или символы. Например, лексемами языка программирования являются: ключевые слова языка, имена переменных, имена констант, имена функций, знаки операций и т. д. Все ключевые слова (иначе — зарезервированные или служебные) являются постоянной частью языка программирования и записываются в точности так, как это установлено правилами языка. Ключевые слова нельзя использовать в качестве имен других структурных единиц языка. В Python строчные и ПРОПИСНЫЕ буквы **различаются**.

В Python особое внимание уделяется **отступам** строк в программном коде. Наличие или отсутствие отступов строк определяет правильность программного кода с точки зрения логики и синтаксиса. Другими словами, если отступы выставлены неверно, то может нарушиться логика работы программы или возникнуть синтаксическая ошибка в, казалось бы, «правильном» программном коде.

Приведем несколько **правил**, которые нам понадобятся на первом этапе и которые надо соблюдать, чтобы избежать ошибок в программном коде.

- Конец строки является концом инструкции — команды или блока, в конце строки точка с запятой не ставится.

- В Python вложенные инструкции оформляются единообразно. Сначала пишется основная инструкция, завершающаяся двоеточием. Затем на следующей строке размещается вложенный блок команд. Он может состоять также из нескольких строк, но все они будут начинаться с одинакового отступа — на «шаг» вправо относительно основной инструкции.

```
Основная инструкция:
    Вложенный блок инструкций
```

- Инструкции, относящиеся к одному блоку, всегда имеют одинаковый отступ! «Шаг» обычно равен четырем пробелам или знаку табуляции Tab[1]. Чем глубже степень вложения, тем большее количество пробелов требуется, однако всегда оно кратно четырем.

[1] Среда автоматически заменяет нажатие клавиши Tab на 4 пробела. Это сделано для удобства пользователя, однако не приветствуется создателями языка.

```
Основная инструкция:
   Вложенная инструкция 1
      Вложенная инструкция 1.1
      Вложенная инструкция 1.2
   Вложенная инструкция 2
      Вложенная инструкция 2.1
      Вложенная инструкция 2.2
```

• В некоторых случаях, например для удобочитаемости программного кода, несколько инструкций можно записать в одной строке через точку с запятой[1]:

```
p = 3; q = 5; s = input("Введите число:")
```

• Длинную инструкцию разрешается записывать в несколько строк, поместив ее внутри круглых, квадратных или фигурных скобок:

```
while (i < (a + b)/2 or
            q = False):

i +=1
```

• Вложенная инструкция может располагаться в той же строке, что и тело основной, если тело вложенной инструкции не является составным, то есть состоит только из одной команды, например:

```
for i in range(1, n): f = f * i
```

Следуя этим правилам, на первом этапе можно в большинстве случаев исключить ошибки синтаксиса при написании кода. Конечно, полное понимание, как правильно писать код на Python, невозможно без практических навыков.

ДОПОЛНЕНИЕ

Более полную информацию, как правильно писать программы на языке Python, можно получить в официальном руководстве по написанию кода. Кроме того, в соглашении будет полезно ознакомиться с тем, что такое документирование кода и для чего это необходимо.
PEP 8 — Style Guide for Python Code // Python Software Fundation. URL: **https://www.python.org/dev/peps/pep-0008**

[1] Не увлекайтесь этим чрезмерно, чтобы не получить обратный эффект.

PEP 8 — руководство по написанию кода на Python // Python 3 для начинающих. URL: **https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html**
PEP 257 — Docstring Conventions // Python Software Fundation. URL: **https://www.python.org/dev/peps/pep-0257**
Документирование кода в Python. PEP 257 // Python 3 для начинающих. URL: **https://pythonworld.ru/osnovy/dokumentirovanie-koda-v-python-pep-257.html**

Программирование на языке Python с помощью исполнителя Turtle

Библиотека **Turtle** («черепаха») представляет собой удобный исполнитель, средой которого является двумерное графическое поле. Эта библиотека очень хороша для тех, кто только начинает изучение основ алгоритмизации и программирования. Она уже встроена в Python, поэтому ее не надо отдельно устанавливать. Библиотека **Turtle** содержит набор простых и понятных команд, результат выполнения которых наглядно отображается в графическом окне. Также с помощью **Turtle** можно легко перестроиться на программирование в Python. Ниже мы разберем несколько примеров программ, экспериментируя с которыми можно самостоятельно развить в себе навыки программирования в Python. Рассмотрим работу некоторых из множества методов **Turtle**.

Turtle-методы

Для рисования и перемещения по рабочему полю в исполнитель **Turtle** заложен ряд простых для понимания методов. Главный инструмент **Turtle** — **перо**. Оно имеет два состояния: опущено и поднято (по умолчанию опущено). Кроме того, можно задать толщину и цвет пера. Рисование осуществляется путем перемещения исполнителя **Turtle** с опущенным пером. Ниже дан список команд, с помощью которых можно запрограммировать **Turtle** для рисования простейших рисунков. Символ «|» означает выбор одного из нескольких вариантов вызова данной команды.

Методы перемещения и рисования

`forward(n)` | `fd(n)` — перемещение исполнителя вперед на *n* пикселей;

`backward(n)` | `bk(n)` | `back(n)` — перемещение исполнителя назад на *n* пикселей;

`right(a)` | `rt(a)` — поворот исполнителя вправо (по часовой стрелке) на *a* градусов;

`left(a)` | `lt(a)` — поворот влево (против часовой стрелки) на *a* градусов

`goto(x,y)` | `setpos(x,y)` | `setposition(x,y)` — перейти в точку с координатами (*x, y*);

`setx(x)` — сместиться в указанную координату *x*, сохраняя *y*;

`sety(y)` — сместиться в указанную координату *y*, сохраняя *x*;

`setheading(a)` | `seth(a)` — задать направление на *a* градусов;

`home()` — вернуться в точку с координатами (0,0);

`circle(r)` — нарисовать окружность радиусом *r*;

`dot(s,c)` — нарисовать точку в текущей позиции, где *s* — размер знака точки в пикселях; *c* — название цвета в кавычках, например «red», «yellow», «dimgray» и т. д.[1];

`stamp()` — создать клон «черепашки»;

`undo()` — отменить последнее действие;

`speed(s)` — задать скорость рисования s по шкале от 0 до 10 условных единиц.

Методы состояния

`pos()` — возвращает текущую позицию курсора (*x, y*);

`xcor()` — возвращает текущую позицию курсора по координате x;

`ycor()` — возвращает текущую позицию курсора по координате y;

`distance(x,y)` — возвращает значение расстояния от текущей позиции курсора до координаты точки (*x, y*);

Методы управления пером

`pd()` — опустить перо;

`pu()` — поднять перо;

`pensize()` | `width()` — установить толщину пера по шкале от 0 до 10 условных единиц;

`pencolor()` — устанавливает/сообщает цвет пера;

---

[1] Цветовая схема — названия цветов. URL: **https://undoshutdown.blogspot.com/2018/06/matplotlib-python.html**

`fillcolor()` — устанавливает/сообщает цвет заливки;
`begin_fill()` — начало заливки установленным цветом;
`end_fill()` — конец заливки установленным цветом.

Рассмотрим примеры двух простых программ для рисования:

1) прямоугольника с текущей позиции курсора;
2) окружности с центром в заданной точке.

1. Чтобы нарисовать прямоугольник, сделаем запрос двух целых чисел. Для хранения их значений создадим две **переменные**.

ПОЯСНЕНИЕ

Переменная — это именованная область памяти компьютера, выделенная для хранения данных. Доступ к данным и изменение их значения в ходе работы программы осуществляются с помощью *идентификатора* — уникального имени переменной. В Python для переменной не требуется специального объявления, она может быть объявлена в любом месте программы путем присваивания ей любого значения. По мере выполнения программы тип переменной может меняться. Более подробно о переменных и их типах в Python можно прочитать здесь[1,2]. В основном мы будем использовать функции, одноименные с типами переменных, — функции преобразования типов данных. О том, что такое процедура или функция, мы подробно остановимся позднее. Приведем часть основных типов данных и процедур, к ним приводящих:

Тип данных	Описание	Приведение к типу (функция)	Пример
`int`	Целое число	`int()`	5 –3
`float`	Число с плавающей точкой	`float()`	5.0 –3.4
`str`	Строка как последовательный набор символов	`str()`	«Привет!»
`bool`	Логическая переменная	`bool()`	True False

[1] Переменные в Python. URL: **https://codelessons.ru/python/uroki-dlya-novichkov-python/peremennye-v-python.html**

[2] Типы данных в Python. URL: **http://pythonicway.com/python-data-types**

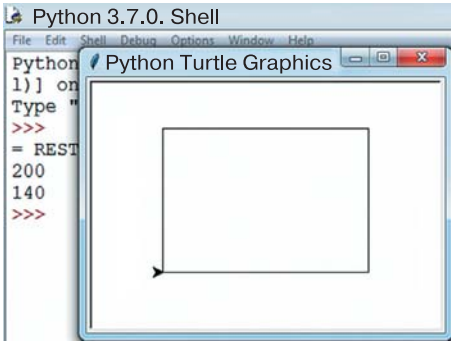

Рис. 1.5. Рисование прямоугольника с помощью исполнителя **Turtle**

Перейдем к практическому заданию. Пусть значения, записанные в переменные, будут определять длину и ширину некоторого прямоугольника. Дважды в цикле нарисуем одну за другой стороны прямоугольника: длину и ширину. Ниже представлен небольшой программный код. Результат работы программы изображен на **рис. 1.5**. После символа # даны комментарии, которые игнорируются исполнителем.

```
# подключение библиотеки Turtle
from turtle import * # загрузить всё из корневого
# каталога библиотеки turtle
# ввести с клавиатуры размеры прямоугольника (a * b)
a = int(input())  # ввод целого числа в переменную a
b = int(input())  # ввод целого числа в переменную b
# рисовать стороны прямоугольника
for i in range(0,2): # выполняем два раза
    fd(a)  # вперед на a
    lt(90) # поворот налево на 90
    fd(b)  # вперед на b
    lt(90) # поворот налево на 90
```

2. Стандартная функция **Turtle** для рисования окружности по умолчанию рисует ее из нижней точки, что порой не очень удобно. Составим небольшую программу, которая рисует окружность заданного радиуса с центром в заданной точке. Для этого надо поднять перо, сместиться в заранее рассчитанную точку, опустить перо и нарисовать окружность. При этом исполнитель **Turtle** должен быть ориентирован по умолчанию — слева направо. Результат работы программы представлен на **рис. 1.6**.

КОМПЬЮТЕРНОЕ ЗРЕНИЕ НА PYTHON®

ПЕРВЫЕ ШАГИ

Шакирьянов Эдуард Данисович. Кандидат физико-математических наук, доцент кафедры цифровых технологий и моделирования Уфимского государственного нефтяного технического университета (УГНТУ), преподаватель Молодежного технопарка УГНТУ.

Еще совсем недавно создание устройств, оснащенных простейшим компьютерным зрением, было делом нешуточной сложности и требовало от разработчика больших интеллектуальных и материальных ресурсов. И вдруг все изменилось – на сцену вышли недорогие компактные микрокомпьютеры и общедоступное программное обеспечение. Теперь даже школьникам среднего и старшего возраста по плечу написать программный код для работы несложной системы с компьютерным зрением.

Книга, которую вы держите в руках, является учебным курсом для школьников, начинающих изучать компьютерное зрение с языком программирования Python® и библиотекой OpenCV. На ее страницах рассмотрены вопросы, раскрывающие особенности установки Python®, различных библиотек, в том числе OpenCV, и операционной системы Raspbian. Учебный материал изложен по отдельным темам:

- Программирование на языке Python®.
- Поиск и выделение цветных объектов на графическом изображении и в видеопотоке средствами OpenCV.
- Программирование колесной робоплатформы под управлением Raspberry Pi® 3, оснащенной CSI-камерой.

Большую помощь читателю окажут многочисленные иллюстрации и листинги программных кодов, а также ссылки на источники и интернет-ресурсы.

Книга будет полезна школьникам среднего и старшего возраста, педагогам дополнительного образования и всем начинающим изучать компьютерное зрение с помощью языка программирования Python® и открытой библиотеки компьютерного зрения OpenCV-Python.