грамма вычислит неправильное значение. Однако если при наборе имени INTEREST_RATE вы сделаете опечатку, то интерпретатор Python выведет сообщение с указанием, что такое имя не определено.

Контрольная точка

2.40. Каковы три преимущества от использования именованных констант?

2.41. Напишите инструкцию Python, которая задает именованную константу для 10-процентной скидки.

2.12 Введение в черепашью графику

Ключевые положения

Черепашья графика, или графика с относительными командами, — это интересный и очень простой способ изучения элементарных принципов программирования. Система черепашьей графики языка Python имитирует "черепаху", которая повинуется командам рисования простых графических изображений.

В конце 1960-х годов преподаватель Массачусетского технологического института (MIT) Сеймур Пейперт начал использовать роботизированную черепаху для обучения программированию. Черепаха была связана с компьютером, на котором обучаемый мог вводить команды, побуждающие черепаху перемещаться. У черепахи также имелось перо, которое можно было поднимать и опускать, и поэтому ее можно было класть на лист бумаги и программировать рисование изображений. Python имеет систему *черепашьей графики*, которая имитирует эту роботизированную черепаху. Данная система выводит на экран небольшой курсор (черепаху). Для того чтобы перемещать черепаху по экрану, рисуя линии и геометрические фигуры, можно использовать инструкции Python.

Видеозапись "Введение в черепашью графику" (Introduction to Turtle Graphics)

Первый шаг в использовании системы черепашьей графики Python состоит в написании приведенной ниже инструкции:

```
import turtle
```

Система черепашьей графики не встроена в интерпретатор Python и хранится в файле как *модуль* `turtle`. Поэтому инструкция `import turtle` загружает данный модуль в память, чтобы интерпретатор Python мог его использовать.

При написании программы Python, в которой используется черепашья графика, в начале программы следует написать инструкцию импорта `import`. Если есть желание поэкспериментировать с черепашьей графикой в интерактивном режиме, то эту инструкцию можно набрать прямо в оболочке Python:

```
>>> import turtle
>>>
```

Рисование отрезков прямой при помощи черепахи

Черепаха языка Python первоначально расположена в центре графического окна, которое служит ее холстом. Для того чтобы отобразить черепаху в ее окне, в интерактивном режиме

можно ввести команду `turtle.showturtle()`. Ниже приводится демонстрационный сеанс, который импортирует модуль черепахи и затем показывает ее:

```
>>> import turtle
>>> turtle.showturtle()
```

В результате появляется графическое окно (рис. 2.11). Черепаха совсем не похожа на черепаху в собственном смысле этого слова. Напротив, она скорее выглядит, как указатель стрелки (➤). Это очень важно, потому что он указывает в ту сторону, куда черепаха обращена в настоящее время. Если дать черепахе команду переместиться вперед, то она переместится в том направлении, куда указывает наконечник стрелки. Давайте попробуем. Для перемещения черепахи вперед на *n* пикселов применяется команда `turtle.forward(n)`. (Просто наберите желаемое число пикселов вместо *n*.) Например, команда `turtle.forward(200)` переместит черепаху вперед на 200 пикселов. Ниже приводится пример полного сеанса в оболочке Python:

```
>>> import turtle
>>> turtle.forward(200)
>>>
```

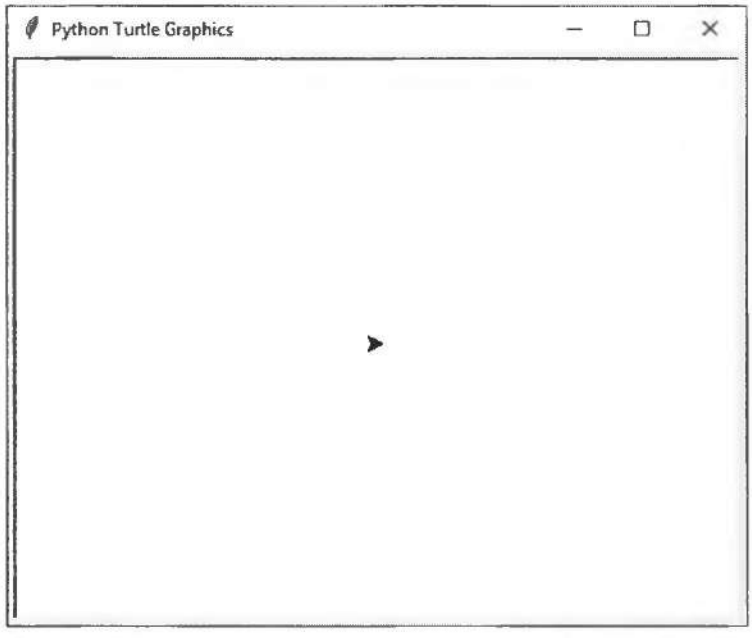

РИС. 2.11. Графическое окно черепахи

На рис. 2.12 представлен результат этого интерактивного сеанса. Обратите внимание, что по ходу перемещения черепахи была начерчена линия.

Поворот черепахи

Когда черепаха появляется в начале сеанса, она по умолчанию направлена на восток (т. е. вправо, или под углом 0°) — рис. 2.13.

Вы можете повернуть черепаху так, чтобы она смотрела в другом направлении, используя команду `turtle.right(угол)` либо команду `turtle.left(угол)`. Команда `turtle.right(угол)`

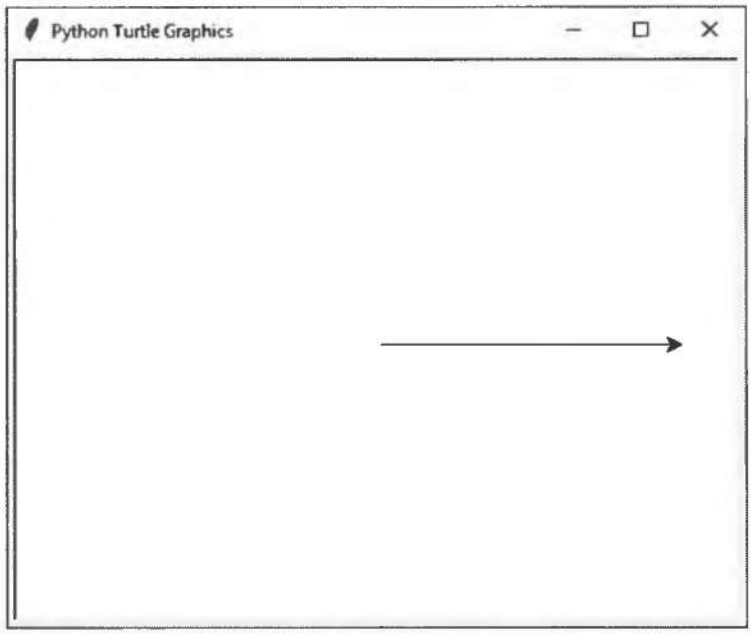

РИС. 2.12. Черепаха перемещена вперед на 200 пикселов

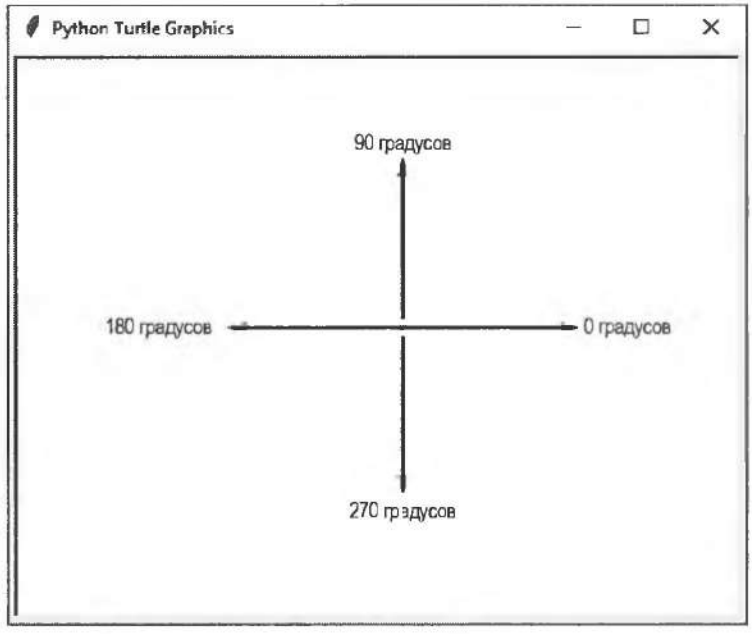

РИС. 2.13. Угловые направления черепахи

поворачивает черепаху вправо на градусы *угла*, а команда `turtle.left(`*угол*`)` поворачивает черепаху влево на градусы *угла*. Вот пример сеанса, в котором используется команда `turtle.right(`*угол*`)`:

```
>>> import turtle
>>> turtle.forward(200)
```

```
>>> turtle.right(90)
>>> turtle.forward(200)
>>>
```

Эта сессия сначала перемещает черепаху вперед на 200 пикселов. Затем она поворачивает черепаху вправо на 90° (черепаха будет направлена вниз). Затем она перемещает черепаху вперед на 200 пикселов. Результаты сеанса показаны на рис. 2.14.

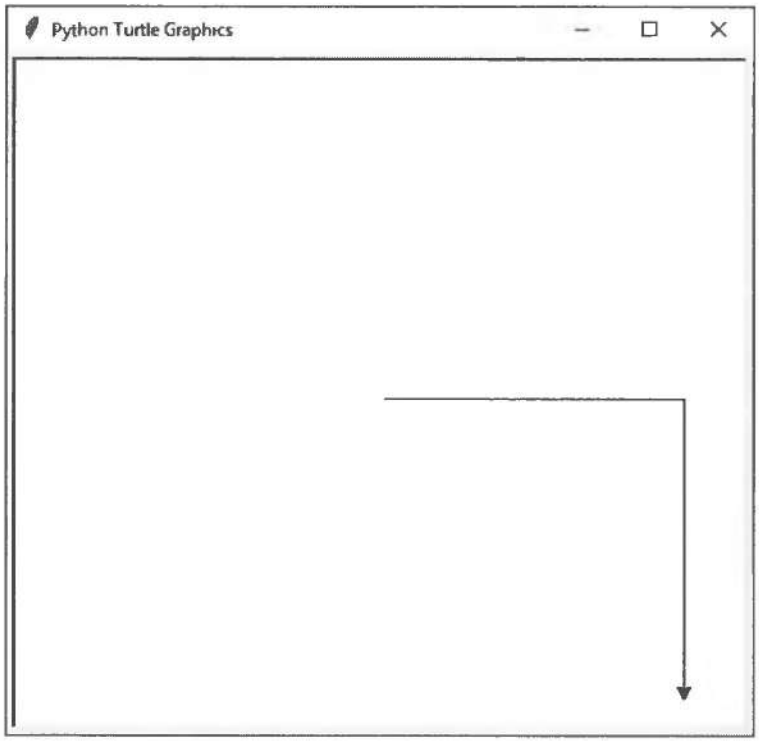

РИС. 2.14. Черепаха поворачивается вправо

Вот пример сеанса, в котором используется команда `turtle.left(`*угол*`)`:

```
>>> import turtle
>>> turtle.forward(100)
>>> turtle.left(120)
>>> turtle.forward(150)
>>>
```

Этот сеанс сначала перемещает черепаху на 100 пикселов вперед. Затем он поворачивает черепаху на 120° влево (черепаха будет смотреть на северо-запад). И далее он перемещает черепаху на 150 пикселов вперед. Результат сеанса показан на рис. 2.15.

Следует иметь в виду, что команды `turtle.right` и `turtle.left` поворачивают черепаху под указанным углом. Например, текущее угловое направление черепахи составляет 90° (строго на север). Если ввести команду `turtle.left(20)`, то черепаха будет повернута влево на 20°. Это означает, что курс черепахи будет 110°, т. е. угловое направление черепахи составит 110°.

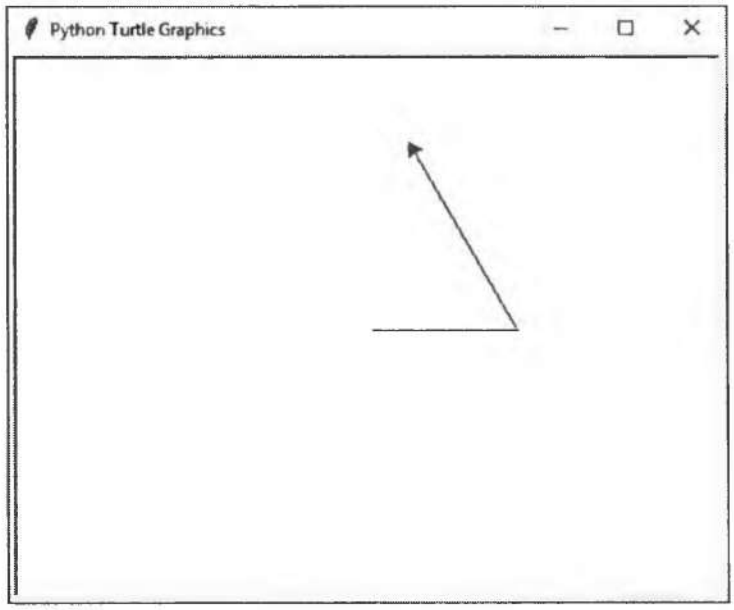

РИС. 2.15. Черепаха поворачивается влево

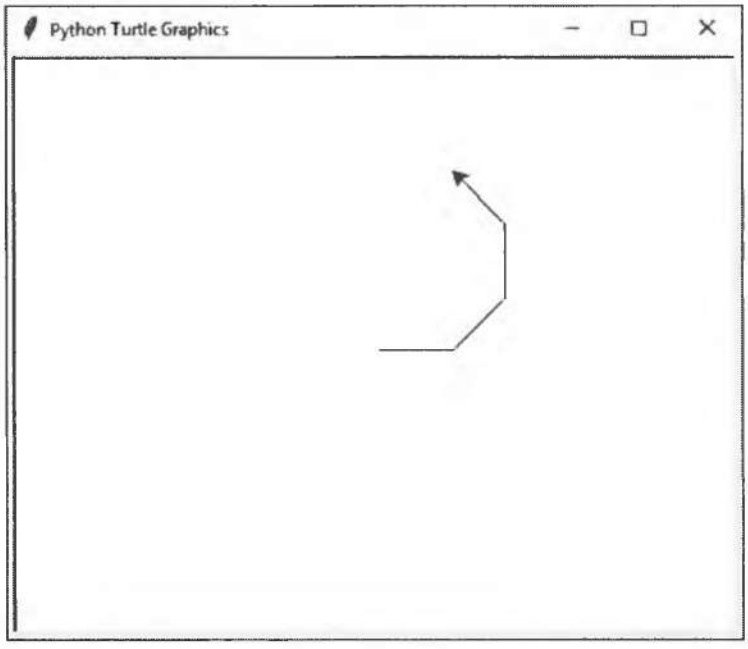

РИС. 2.16. Многократное поворачивание черепахи на 45°

В качестве еще одного примера взгляните на приведенный ниже интерактивный сеанс:

```
>>> import turtle
>>> turtle.forward(50)
>>> turtle.left(45)
>>> turtle.forward(50)
>>> turtle.left(45)
```

```
>>> turtle.forward(50)
>>> turtle.left(45)
>>> turtle.forward(50)
>>>
```

На рис. 2.16 показан результат сеанса. В начале этого сеанса направление черепахи составляет 0°. В строке 3 черепаха поворачивается на 45° влево. В пятой строке черепаха снова поворачивается влево на дополнительные 45°. В седьмой строке черепаха еще раз поворачивается на 45° влево. После всех этих поворотов на 45° угловое направление черепахи, наконец, составит 135°.

Установка углового направления черепахи в заданный угол

Команда `turtle.setheading(`*угол*`)` применяется для установки углового направления черепахи в заданный угол. В качестве аргумента *угол* надо просто указать желаемый угол. Вот пример:

```
>>> import turtle
>>> turtle.forward(50)
>>> turtle.setheading(90)
>>> turtle.forward(100)
>>> turtle.setheading(180)
>>> turtle.forward(50)
>>> turtle.setheading(270)
>>> turtle.forward(100)
>>>
```

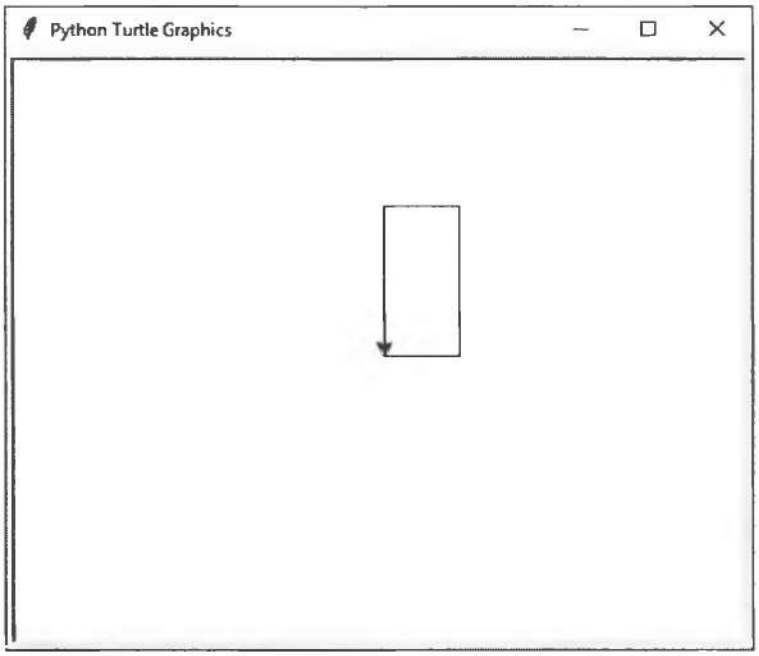

РИС. 2.17. Установка углового направления черепахи

Как обычно, первоначальное угловое направление черепахи составляет 0°. В третьей строке направление черепахи установлено в 90°. Затем в пятой строке направление черепахи установлено в 180°. Потом в седьмой строке направление черепахи установлено в 270°. Результат сеанса показан на рис. 2.17.

Получение текущего углового направления черепахи

В интерактивном сеансе можно применить команду `turtle.heading()`, чтобы вывести на экран текущее угловое направление черепахи. Пример:

```
>>> import turtle
>>> turtle.heading()
0.0
>>> turtle.setheading(180)
>>> turtle.heading()
180.0
>>>
```

Поднятие и опускание пера

Исходная роботизированная черепаха лежала на большом листе бумаги и имела перо, которое можно было поднимать и опускать. Когда перо опускалось, оно вступало в контакт с бумагой и чертило линию во время перемещения черепахи. Когда перо поднималось, оно не касалось бумаги, и поэтому черепаха могла перемещаться без начертания линии.

В Python команда `turtle.penup()` применяется для поднятия пера, а команда `turtle.pendown()` — для опускания пера. Когда перо поднято, черепаха будет перемещаться без начертания линии. Когда перо опущено, черепаха оставляет линию во время ее перемещения. (По умолчанию перо опущено.) Приведенный ниже сеанс демонстрирует пример. Результат сеанса представлен на рис. 2.18.

```
>>> import turtle
>>> turtle.forward(50)
>>> turtle.penup()
>>> turtle.forward(25)
>>> turtle.pendown()
>>> turtle.forward(50)
>>> turtle.penup()
>>> turtle.forward(25)
>>> turtle.pendown()
>>> turtle.forward(50)
>>>
```

Рисование кругов и точек

Для того чтобы черепаха нарисовала круг, применяют команду `turtle.circle(`*радиус*`)`, и она начертит круг с радиусом в пикселах, заданным аргументом *радиус*. Например, команда `turtle.circle(100)` побуждает черепаху нарисовать круг с радиусом 100 пикселов. Приведенный ниже интерактивный сеанс выводит результат, показанный на рис. 2.19:

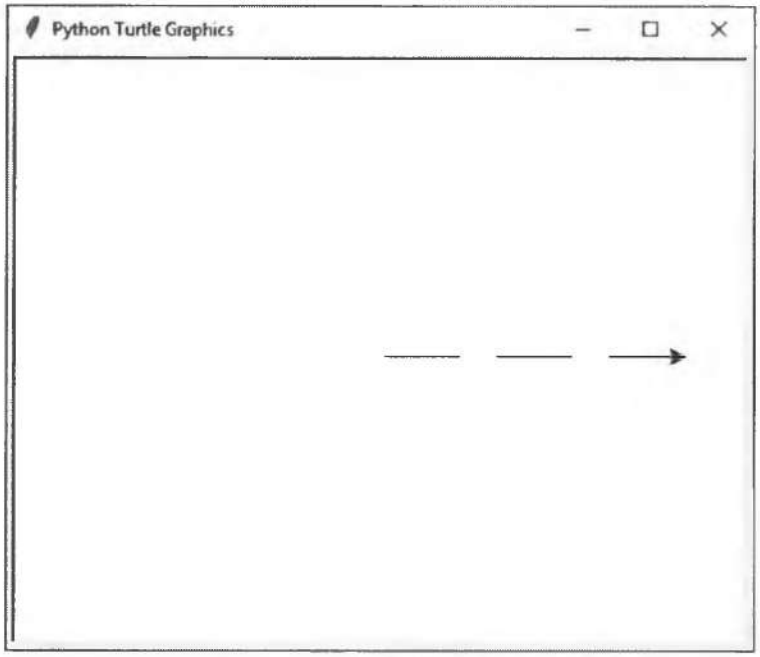

РИС. 2.18. Поднятие и опускание пера

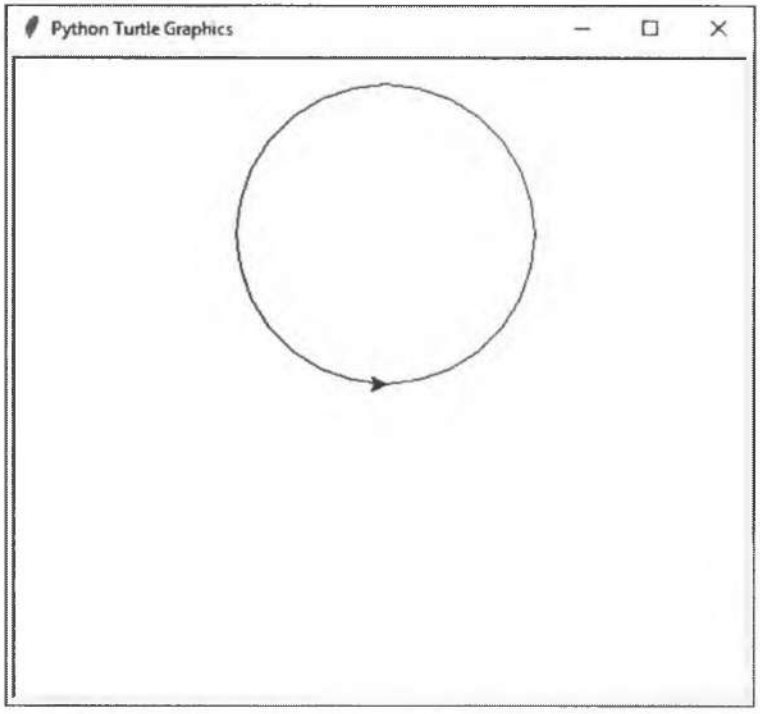

РИС. 2.19. Круг

```
>>> import turtle
>>> turtle.circle(100)
>>>
```

Команда `turtle.dot()` применяется, чтобы заставить черепаху начертить простую точку. Например, приведенный ниже интерактивный сеанс производит результат, который показан на рис. 2.20:

```
>>> import turtle
>>> turtle.dot()
>>> turtle.forward(50)
>>> turtle.dot()
>>> turtle.forward(50)
>>> turtle.dot()
>>> turtle.forward(50)
>>>
```

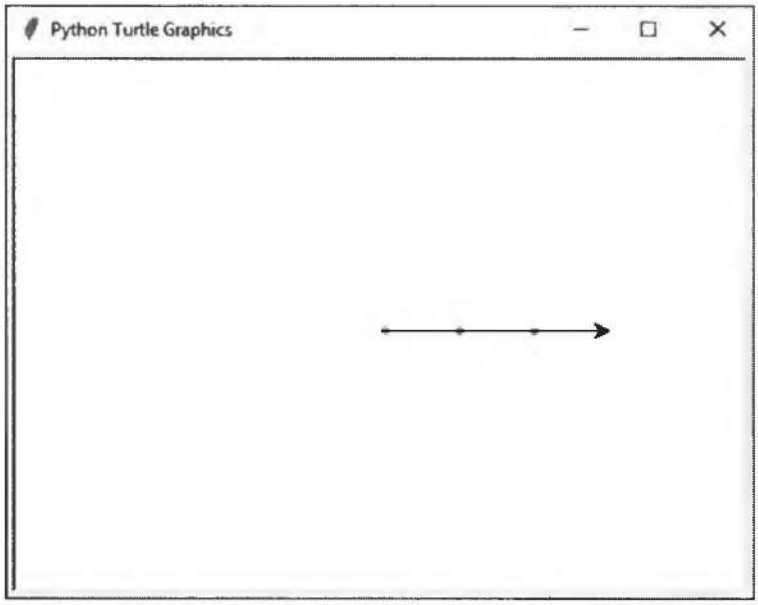

РИС. 2.20. Рисование точек

Изменение размера пера

Для изменения ширины пера черепахи в пикселах можно применить команду `turtle.pensize(`*ширина*`)`. Аргумент *ширина* — это целое число, которое задает ширину пера. Например, следующий ниже интерактивный сеанс назначает ширине пера 5 пикселов и затем рисует круг:

```
>>> import turtle
>>> turtle.pencsize(5)
>>> turtle.circle(100)
>>>
```

Изменение цвета пера

Для изменения цвета пера, которым рисует черепаха, можно применить команду `turtle.pencolor(цвет)`. Аргумент *цвет* — это название цвета в виде строкового литерала. Например, приведенный ниже интерактивный сеанс меняет цвет пера на красный и затем рисует круг:

```
>>> import turtle
>>> turtle.pencolor('red')
>>> turtle.circle(100)
>>>
```

С командой `turtle.pencolor` можно использовать многочисленные предопределенные названия цветов, и в *приложении 4* приведен их полный перечень. Вот некоторые из наиболее широко используемых названий цветов: `'red'`, `'green'`, `'blue'`, `'yellow'` и `'cyan'`[1].

Изменение цвета фона

Для изменения фонового цвета графического окна черепахи можно применить команду `turtle.bgcolor(цвет)`. Аргумент *цвет* — это название цвета в виде строкового литерала. Например, приведенный ниже интерактивный сеанс меняет цвет фона на серый, цвет рисунка на красный и затем рисует круг:

```
>>> import turtle
>>> turtle.bgcolor('gray')
>>> turtle.pencolor('red')
>>> turtle.circle(100)
>>>
```

Как было упомянуто ранее, существуют многочисленные предопределенные названия цветов, и в *приложении 4* приведен их полный перечень.

Возвращение экрана в исходное состояние

Для возвращения графического окна черепахи в исходное состояние существуют три команды: `turtle.reset()`, `turtle.clear()` и `turtle.clearscreen()`.

- Команда `turtle.reset()` стирает все рисунки, которые в настоящее время видны в графическом окне, задает черный цвет рисунка и возвращает черепаху в ее исходное положение в центре экрана. Эта команда не переустанавливает цвет фона графического окна.
- Команда `turtle.clear()` просто стирает все рисунки, которые в настоящее время видны в графическом окне. Она не меняет положение черепахи, цвет пера или цвет фона графического окна.
- Команда `turtle.clearscreen()` стирает все рисунки, которые в настоящее время видны в графическом окне, переустанавливает цвет пера в черный, переустанавливает цвет фона графического окна в белый и возвращает черепаху в ее исходное положение в центре графического окна.

[1] Красный, зеленый, синий, желтый и голубой. — *Прим. пер.*

Установление размера графического окна

Для установления размера графического окна можно применить команду `turtle.setup(`*ширина*`, `*высота*`)`. Аргументы *ширина* и *высота* — это ширина и высота в пикселах. Например, приведенный ниже интерактивный сеанс создает графическое окно шириной 640 и высотой 480 пикселов:

```
>>> import turtle
>>> turtle.setup(640, 480)
>>>
```

Перемещение черепахи в заданную позицию

Декартова система координат используется для идентификации позиции каждого пиксела в графическом окне черепахи (рис. 2.21). Каждый пиксел имеет координаты *X* и *Y*. Координата *X* идентифицирует горизонтальную позицию пиксела, координата *Y* — его вертикальную позицию. Важно понимать следующее:

- пиксел в центре графического окна находится в позиции (0, 0), т. е. его координата *X* равняется 0, координата *Y* равняется 0;
- значения координат *X* увеличиваются при перемещении в правую сторону и уменьшаются при перемещении в левую сторону окна;
- значения координат *Y* увеличиваются при перемещении вверх и уменьшаются при перемещении вниз окна;
- пикселы, расположенные справа от центральной точки, имеют положительные координаты *X*, а расположенные слева от центральной точки — отрицательные координаты *X*;

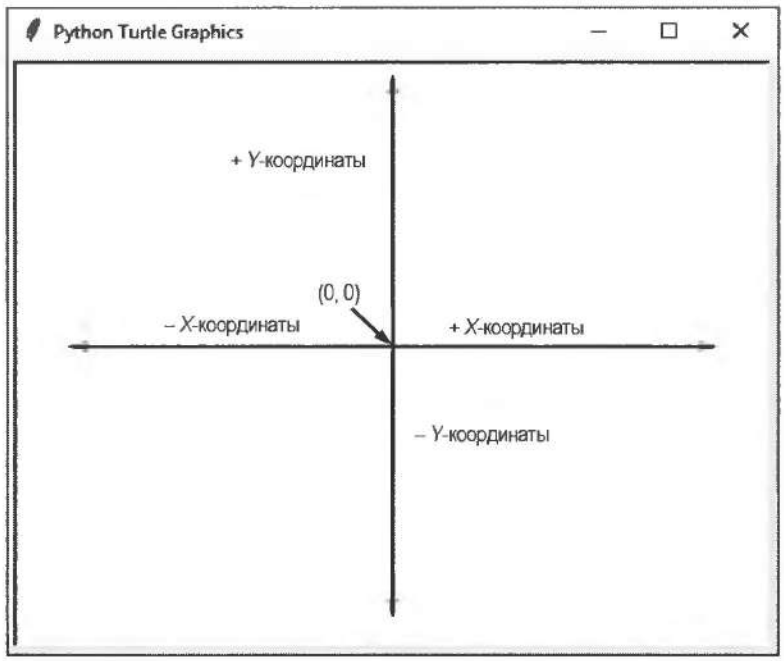

РИС. 2.21. Декартова система координат

- пикселы, расположенные выше центральной точки, имеют положительные координаты *Y*, а расположенные ниже центральной точки — отрицательные координаты *Y*.

Для перемещения черепахи из ее текущего положения в конкретную позицию в графическом окне применяется команда `turtle.goto(x, y)`. Аргументы `x` и `y` — это координаты позиции, в которую черепаха будет перемещена. Если перо черепахи опущено вниз, то по ходу перемещения черепахи будет начерчена линия. Например, приведенный ниже интерактивный сеанс чертит линии, показанные на рис. 2.22:

```
>>> import turtle
>>> turtle.goto(0, 100)
>>> turtle.goto(-100, 0)
>>> turtle.goto(0, 0)
>>>
```

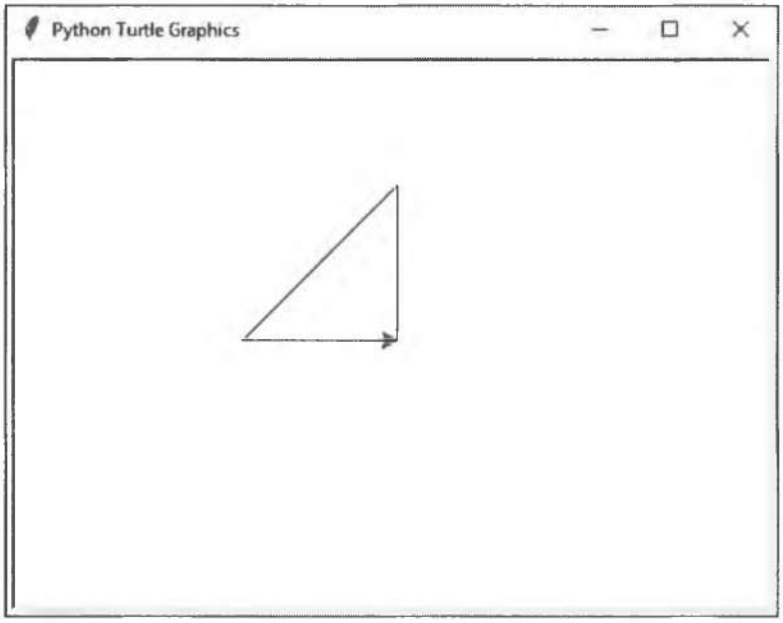

РИС. 2.22. Перемещение черепахи

Получение текущей позиции черепахи

Для отображения текущей позиции черепахи в интерактивном сеансе применяется команда `turtle.pos()`:

```
>>> import turtle
>>> turtle.goto(100, 150)
>>> turtle.pos()
(100.00, 150.00)
>>>
```

Для отображения координаты *X* черепахи служит команда `turtle.xcor()`, а для отображения координаты *Y* черепахи — команда `turtle.ycor()`:

```
>>> import turtle
>>> turtle.goto(100, 150)
>>> turtle.xcor()
100
>>> turtle.ycor()
150
>>>
```

Управление скоростью анимации черепахи

Для изменения скорости, с которой черепаха перемещается, можно применить команду `turtle.speed(`*скорость*`)`. Аргумент *скорость* — это число в диапазоне от 0 до 10. Если указать 0, то черепаха будет делать все свои перемещения мгновенно (анимация отключена). Например, приведенный ниже интерактивный сеанс отключает анимацию черепахи и затем рисует круг. В результате круг будет нарисован немедленно:

```
>>> import turtle
>>> turtle.speed(0)
>>> turtle.circle(100)
>>>
```

Если указать значение скорости в диапазоне 1–10, то 1 будет самой медленной скоростью, 10 — самой большой скоростью. Приведенный ниже интерактивный сеанс устанавливает скорость анимации в 1 (в самую медленную скорость) и затем рисует круг:

```
>>> import turtle
>>> turtle.speed(1)
>>> turtle.circle(100)
>>>
```

При помощи команды `turtle.speed()` можно получить текущую скорость анимации (аргумент *скорость* не указывается):

```
>>> import turtle
>>> turtle.speed()
3
>>>
```

Сокрытие черепахи

Если не нужно, чтобы черепаха отображалась на экране, то для этого применяется команда `turtle.hideturtle()`, которая ее прячет. Эта команда не изменяет то, как рисуется графическое изображение, она просто скрывает значок черепахи. Когда потребуется снова вывести черепаху на экран, применяется команда `turtle.showturtle()`.

Вывод текста в графическое окно

Для вывода текста в графическое окно применяется команда `turtle.write(`*текст*`)`. Аргумент *текст* — это строковый литерал, который требуется вывести на экран. После вывода левый нижний угол первого символа будет расположен в точке с координатами *X* и *Y* чере-