Тема 13. Графика. Модуль Turtle

13.1. Инициализация графики

На языке программирования Python реализована возможность работы с различными графическими библиотеками. Данная тема посвящена работе с библиотекой **Turtle** (Черепаха). Исполнитель «**Черепаха**» представляет собой некое перо (или хвост, оставляющий след), которое можно отпускать, поднимать, перемещать. Также перу можно устанавливать цвет и толщину.

Для работы с исполнителем Черепаха сначала нужно подключить модуль **turtle**

```
import turtle
```

Подключение таким способом требует при обращении перед каждым методом писать имя модуля. Если все-таки вам не хочется перед методами писать имя модуля, то напишите:

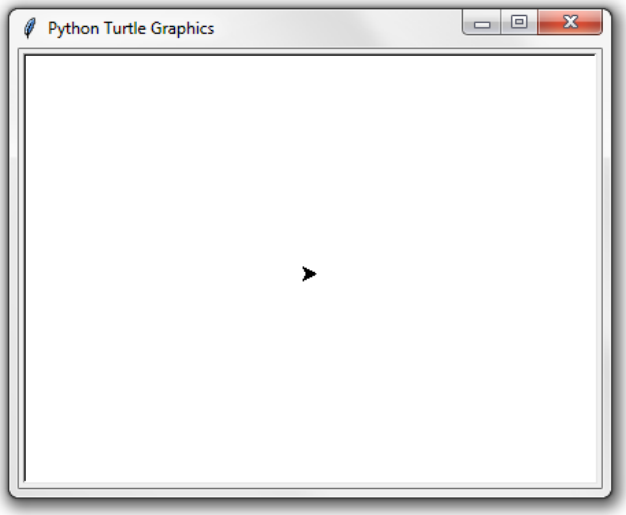
```
from turtle import *
```

Но если написать только команду подключения модуля, мы ничего не увидим после запуска программы. Для отображения графического окна необходимо добавить команду:

```
turtle.reset()
```

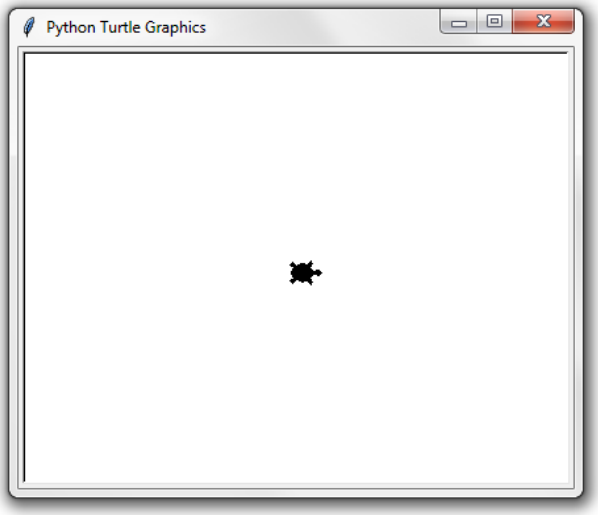
На самом деле у этой команды более широкий функционал. Эта команда выставляет Черепаху в начальное положение, сбрасывает все параметры на значения по умолчанию и стирает все, что было нарисовано до этого.

Теперь, после запуска программы мы увидим окно «**Python Turtle Graphics**». В центре окна будет отображаться перо в виде стрелки, повернутой вправо. Это начальное положение исполнителя «Черепаха». Координаты точки (0,0). При рисовании используется декартова система координат.

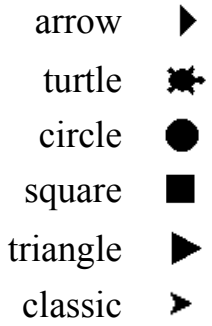


В многих средах программирования исполнителя «Черепаха» (например в таких, как «ЛогоМиры» и «КуМир») перо отображается в виде черепахи. Аналогичный вид перу можно задать и в Python. Для этого достаточно добавить команду:

```
turtle.shape('turtle')
```



Перо может принимать и другой вид:



Текущее направление Черепахи определяется либо направлением стрелки, либо направлением туловища Черепахи. Полный список команд исполнителя можно посмотреть при помощи инструкции:

```
help('turtle')
```

При ее вызове получим очень большой список, в который входят не только команды для рисования, но также математические и другие команды. В рамках этой главы мы будем рассматривать только графические команды.

13.2. Список команд исполнителя для рисования

1. **up()** – поднять перо. Используется в случае, если необходимо переместить Черепаху без оставления следа рисования.

```
turtle.up()
```

2. **down()** – опустить перо. Используется для того, чтобы при перемещении Черепаха оставляла за собой след (в виде линии).

```
turtle.down()
```

3. **width**(*число*) – толщина пера. Задает толщину пера в пикселях.

```
turtle.width(2) # установить толщину пера 2px
```

4. **goto**(*x, y*) – перемещение пера в точку с координатами (x,y).

```
turtle.goto(100,75)
```

5. **color**(*цвет1,[цвет2]*) – цвет пера и заливки.

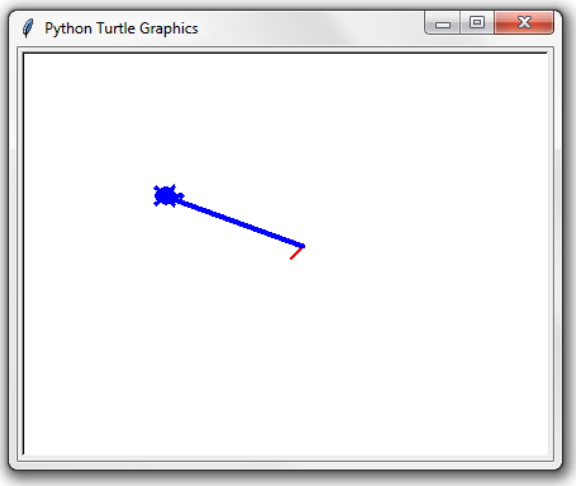
   Если указан только один цвет, то цвет пера и цвет заливки совпадают, а если 2 цвета через запятую, то первый – это цвет пера, а второй – цвет заливки.
   Допустимы два способа задания цвета: словесное название цвета и шестнадцатиричное обозначение цвета. При шестнадцатиричном обозначении в начало добавляется символ «#».

```
turtle.color('red') # задали красный цвет пера
turtle.color('#ff0000') # задали красный цвет пера
```

Пример:

```
import turtle

turtle.reset()
turtle.shape('turtle')
turtle.down()
turtle.color('red')
turtle.width(2)
turtle.goto(10,10)
turtle.color('#0000ff')
turtle.width(4)
turtle.goto(-100,50)
```

В результате получим рисунок:



Обратите внимание на направление Черепахи. Оно осталось изначальным, так как простые перемещения не меняют направление.

6. **forward(***шаг***)** – перемещение вперед. Перемещает Черепаху вперед по направлению стрелки или туловища Черепахи на количество точек, указанное в параметре *шаг*. Шаг может быть как целым числом, так и дробным.

```
turtle.forward(50) # перемещение на 50 точек вперед
```

Если в качестве параметра указать отрицательное число, то Черепаха сместится назад.

7. **backward(***шаг***)** – перемещение назад. Перемещает Черепаху в направлении, противоположном направлению стрелки или туловища Черепахи на количество точек, указанное в параметре *шаг*.

```
turtle.backward(50) # перемещаем на 50 точек назад
```

Если в качестве параметра указать отрицательное число, то Черепаха сместится вперед.

8. **right(***угол***)** – поворот вправо. Поворачивает исполнителя вправо (по часовой стрелке) относительно направления его движения на значение, указанное в параметре *угол*. Угол по умолчанию измеряется в градусах.

```
turtle.right(90) # поворот на 90 градусов по часовой стрелке
```

9. **left(***угол***)** – поворот влево. Поворачивает исполнителя влево (против часовой стрелки) относительно направления его движения на значение, указанное в параметре *угол*. Угол по умолчанию измеряется в градусах.

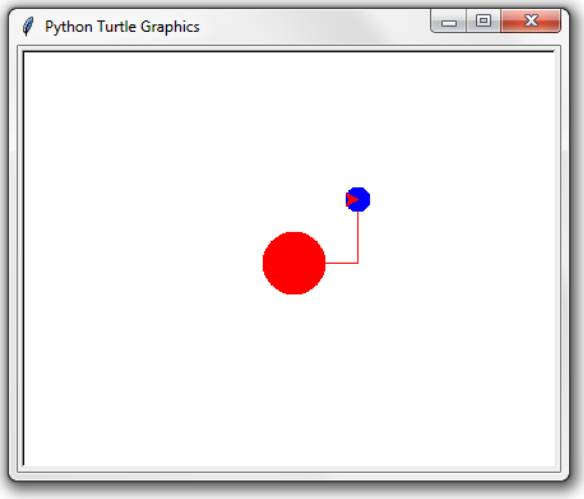
```
turtle.left(90) # поворот на 90 градусов против часовой стрелки
```

10. **dot(***[size],[color]***)** – рисует точку в текущей позиции Черепахи. Если параметры не указаны, то размер точки равен 1, цвет черный. Иначе размер и цвет берутся из параметров.

Пример:

```
import turtle

turtle.reset()
turtle.down()
turtle.color('red')
turtle.dot(50)
turtle.goto(50, 0)
turtle.goto(50, 50)
turtle.dot(20, 'blue')
```

В результате получим рисунок:

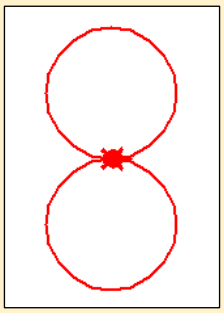


11. **circle**(*r*) – рисование окружности. Рисование окружности радиусом *r* из текущей позиции исполнителя. Радиус может быть как положительным, так и отрицательным числом. Если радиус положительный, то окружность рисуется против часовой стрелки, а если отрицательный, то по часовой стрелке.

```
turtle.circle(-50) # рисуем окружность радиуса 50 по часовой стрелке
```

Пример:

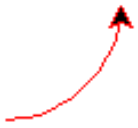
```
import turtle

turtle.reset()
turtle.shape('turtle')
turtle.down()
turtle.color('red')
turtle.width(2)
turtle.circle(50)
turtle.circle(-50)
```



Обратите внимание на направления движения Черепахи.

12. **circle**(*r*, *a*) – рисование дуги. Рисование дуги радиусом *r* и углом, равным *a*. Отсчет угла идет из текущей позиции исполнителя. Аналогично радиус может быть как положительным, так и отрицательным. При положительном радиусе дуга рисуется против часовой стрелки, а при отрицательном по часовой стрелке.

```
turtle.circle(50,90) # рисуем дугу радиусом 50 и углом 90 градусов
                     # против часовой стрелки
```



13. **begin_fill**() – включение закрашивания области.
    end_fill() – выключение закрашивания области.

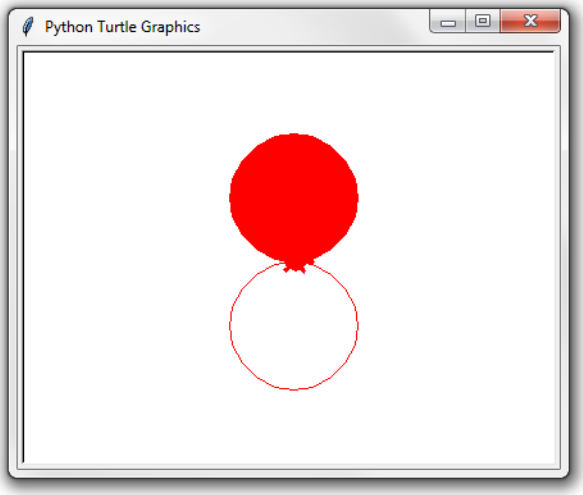
При включении режима закрашивания, все фигуры, нарисованные после него, будут закрашиваться. Поэтому, не забывайте отключать режим закрашивания после заливки нужной фигуры.

```
turtle.begin_fill() # включили режим закрашивания
turtle.end_fill()   # выключили режим закрашивания
```

Пример:

```
import turtle

turtle.reset()
turtle.shape('turtle')
turtle.down()
turtle.color('red')
turtle.width(1)
turtle.begin_fill()
turtle.circle(50)
turtle.end_fill()
turtle.circle(-50)
```

В результате получим:



Незамкнутые области также закрашиваются. При этом первая и последняя точка соединяются по прямой линии.

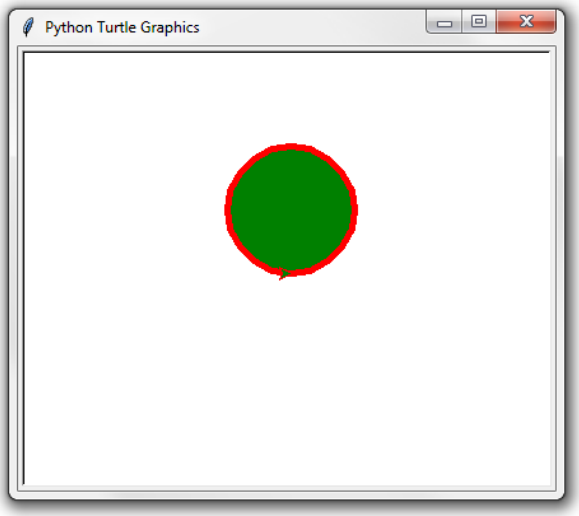
В примере выше и цвет пера, и цвет заливки совпадают. Можно сделать цвета различными. Есть 2 способа.

Способ 1

Использование в свойстве *color* двух цветов. Первый – цвет пера, второй – цвет заливки.

Пример:

```
import turtle

turtle.reset()
turtle.down()
turtle.width(5)
turtle.color('red', 'green')
turtle.begin_fill()
turtle.circle(50)
turtle.end_fill()
```



Способ 2

Отдельно задаем цвет пера и цвет заливки. Способы задания те же, что и у *color*.

pencolor(*цвет*) – цвет пера.

```
turtle.pencolor('red')
```

fillcolor(*цвет*) – цвет заливки.

```
turtle.fillcolor('green')
```

Пример:

```
import turtle

turtle.reset()
turtle.down()
turtle.pencolor('red')
turtle.fillcolor('green')
turtle.begin_fill()
turtle.circle(50)
turtle.end_fill()
```

При использовании этого способа, если не указать цвет заливки, то он по умолчанию будет черным.

14. **write**(*строка*) – вывод текста. Выводит текст, указанный в параметре «строка» в текущую позицию Черепахи.

```
turtle.write('Python') # выводим на рисунок слово 'Python'
```

Есть и другие параметры, например, изменение размера и стиля шрифта. Подробнее об этой команде можно узнать с помощью инструкции

```
help('turtle.write')
```

15. **radians()** – перевод в радианы. Во время рисования переводит углы при поворотах из градусов в радианы. По умолчанию углы измеряются в градусах.

```
turtle.radians()
```

16. **degrees()** – перевод в градусы. Во время рисования переводит углы из радиан в градусы. По умолчанию углы измеряются в градусах.

```
turtle.degrees()
```

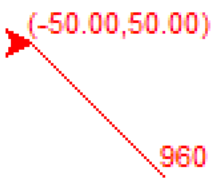
17. **clear()** – очистка. Очищает область рисования в любой момент вызова. Никакие другие параметры не сбрасываются.

```
turtle.clear()
```

18. **window_width()** – получает текущую ширину окна.
19. **window_height()** – получает текущую высоту окна.
20. **position()** – получает текущую позицию Черепахи.

Пример:

```
import turtle

turtle.reset()
turtle.down()
turtle.color('red')
turtle.write(turtle.window_width())
turtle.goto(50,50)
turtle.write(turtle.position())
```



21. **turtle.xcor()** – получает координату *x* текущей позиции Черепахи.

```
turtle.xcor()
```

22. **turtle.ycor()** – получает координату *y* текущей позиции Черепахи.

```
turtle.ycor()
```

23. **turtle.setx**(*число*) – изменение координаты *x* Черепахи.

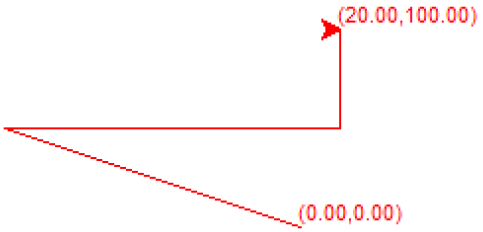
```
turtle.setx(10)
```

24. **turtle.sety**(*число*) – изменение координаты *y* Черепахи.

```
turtle.sety(10)
```

Пример:

```
import turtle

turtle.reset()
turtle.down()
turtle.color('red')
turtle.goto(50,50)
turtle.setx(20)
turtle.sety(20)
turtle.write(turtle.position())
```



25. **tracer**(*f*, [*delay*]) – отображение процесса рисования.

Первый параметр включает или выключает визуализацию процесса рисования (трассировку). По умолчанию визуализация включена.
Если ***f*** = 1, то процесс рисования отображается (значение по умолчанию).
Если ***f*** = 0, то процесс рисования не отображается. Эта команда позволяет ускорить процесс рисования. Но стоит помнить, что если вы отключили режим трассировки, то после окончания рисования нужно обязательно его включить, иначе результат рисования может отображаться некорректно.

```
turtle.tracer(0) # отключили режим трассировки
turtle.circle(50) # нарисовали изображение
turtle.tracer(1) # включили режим трассировки
```

Второй параметр позволяет настроить задержку между шагами рисования. Измеряется в миллисекундах. Удобно использовать, когда необходимо видеть процесс рисования пошагово.

Пример:

```
import turtle

turtle.reset()
turtle.down()
turtle.tracer(1,100)
turtle.color('red')
turtle.goto(50,50)
turtle.setx(20)
turtle.sety(20)
```

26. **bgcolor(**'*цвет*'**)** – установка цвета фона окна для рисования. По умолчанию цвет фона белый.

```
turtle.bgcolor('blue')
```

27. **home()** – возвращает Черепаху в начальное положение, в точку (0,0).

```
turtle.home()
```

28. **mainloop()** – задержка окна. Запрещает автоматическое закрытие окна при закрытии интерпретатора. При попытке закрытия запрашивает подтверждение. Ставится в самом конце программы.

```
turtle.mainloop()
```