Практическая работа «Знакомство с библиотекой turtle»

Часть 1. Базовые команды рисования

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:**

```
from turtle import *
```

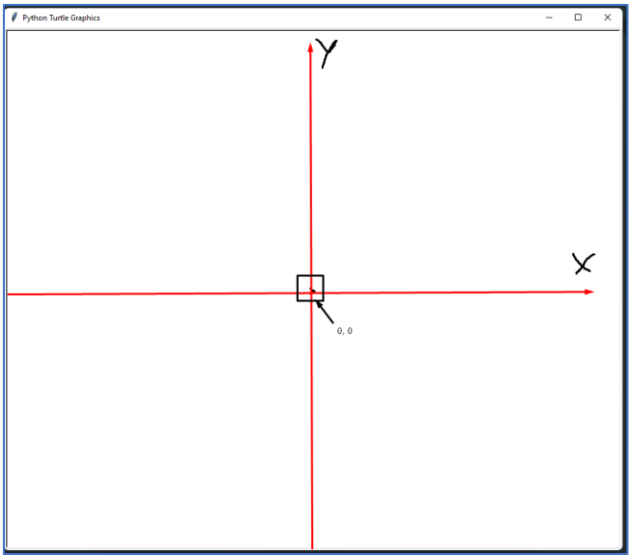
После импорта библиотеки нам необходимо создать черепашку

```
from turtle import *
leo = Turtle()
```

Теперь наша черепашка создана, но при запуске программы окно тут же закрывается. Чтобы отключить автоматическое закрытие окна необходимо в конце программы написать вызов метода **mainloop**()

```
from turtle import *
leo = Turtle()
mainloop()
```

При запуске программы мы увидим нашу черепашку, отображающуюся в виде стрелочки. **При старте программы черепашка всегда находится в центре экрана** (точка с координатами **0, 0**) и смотрит в направлении оси **X**. Пока что черепашка ничего не делает.



Для того, чтобы заставить её нарисовать какую-либо фигуру давайте познакомимся с некоторыми её командами:

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

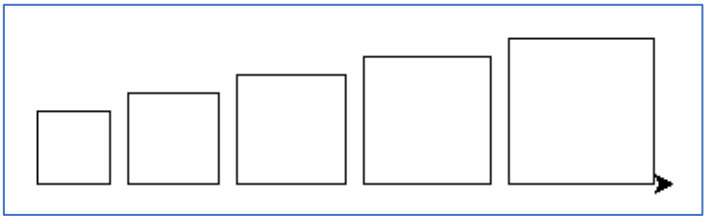
Напишем программу, которая будет рисовать нам квадрат:

```
from turtle import *
leo = Turtle()
leo.forward(40)
leo.left(90)
leo.forward(40)
leo.left(90)
leo.forward(40)
leo.left(90)
leo.forward(40)
leo.left(90)
mainloop()
```

Теперь при запуске программы вы увидите как черепашка очень быстро рисует квадрат со стороной 40 пикселей. Улучшим нашу программу, обернув повторяющиеся команды в цикл и убедимся, что результат будет тот же:

```
from turtle import *
leo = Turtle()
for i in range(4):
    leo.forward(40)
    leo.left(90)
mainloop()
```

Доработаем программу таким образом, чтобы черепашка рисовала нам следующую картинку:

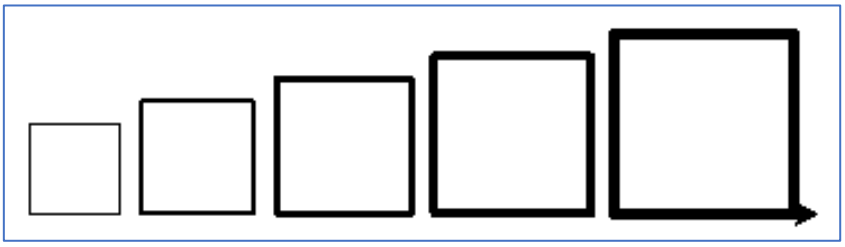


```
from turtle import *
leo = Turtle()
dlina = 40  # длина стороны
for i in range(5):  # рисуем 5 квадратов
    for j in range(4):  # рисуем очередной квадрат
        leo.forward(dlina)
        leo.left(90)
    leo.up()  # поднимаем перо, чтобы след перемещения не рисовался
    dlina += 10  # увеличиваем длину стороны следующего квадрата
    leo.forward(dlina)  # перемещаемся на стартовую позицию нового квадрата
    leo.down()  # опускаем перо
mainloop()
```

Здесь мы использовали две новые команды:

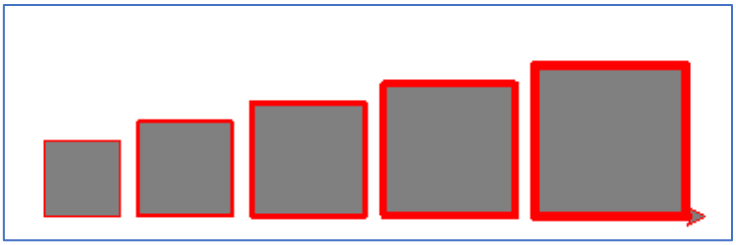
- **up**() – поднять перо. Используется в случае, если необходимо переместить Черепаху без оставления следа рисования
- **down**() – опустить перо. Используется для того, чтобы при перемещении Черепаха оставляла за собой след (в виде линии)

Давайте ещё улучшим нашу программу таким образом, чтобы черепашка рисовала нам следующую картинку:



```
from turtle import *
leo = Turtle()
dlina = 40
for i in range(5):
    leo.width(i + 1) # устанавливаем толщину пера
    for j in range(4):
        leo.forward(dlina)
        leo.left(90)
    leo.up()
    dlina += 10
    leo.forward(dlina)
    leo.down()
mainloop()
```

Здесь мы добавили с вами команду **width**(число) которая изменяет толщину пера, которым рисует черепашка. Улучшим программу ещё больше таким образом, чтобы черепашка рисовала нам следующую картинку:



```
from turtle import *
leo = Turtle()
dlina = 40
for i in range(5):
    leo.width(i + 1)
    leo.color('red', 'grey')  # цвет пера и цвет заливки
    leo.begin_fill()  # включаем закрашивание области
    for j in range(4):
        leo.forward(dlina)
        leo.left(90)
    leo.end_fill()  # выключаем закрашивание области
    leo.up()
    dlina += 10
    leo.forward(dlina)
    leo.down()
mainloop()
```

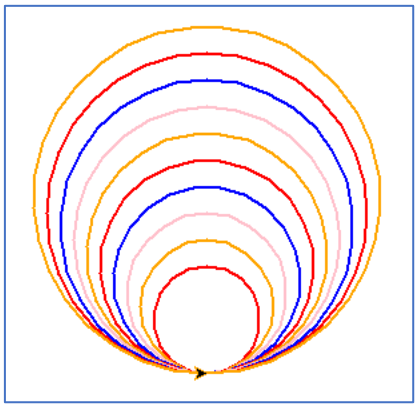
Здесь мы с вами использовали 3 новые команды:

- **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 «#»;
- **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. Незамкнутые области также закрашиваются. При этом первая и последняя точка соединяются по прямой линии.

Часть 2. Рисование окружностей и дуг

Теперь давайте напишем новую программу, в которой черепашка будет нам рисовать следующее изображение

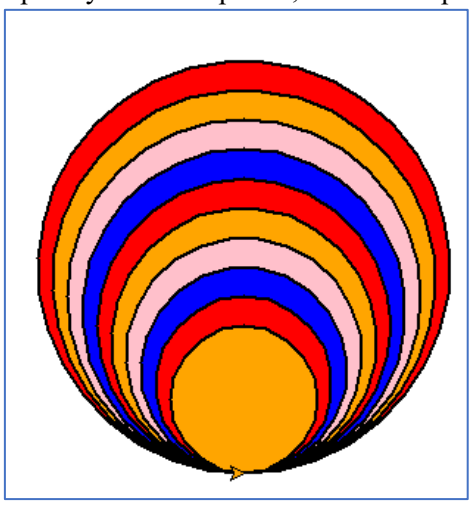


```
from turtle import *
leo = Turtle()
colors = ['red', 'orange', 'pink', 'blue']  # список цветов
leo.width(2)
radius = 40  # стартовый радиус окружности
leo.speed(10)  # скорость рисований
for i in range(10):
    leo.pencolor(colors[i % 4])  # установка цвета пера, цвет заливки не меняется
    leo.circle(radius)  # рисование окружности с нужным радиусом
    radius += 10
mainloop()
```

В данной программе мы использовали 3 новые команды:

- **speed**(число) – изменяет скорость рисования черепашки;
- **pencolor**(цвет) – изменяет цвет пера, цвет заливки не меняется;
- **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.

Давайте ещё доработаем нашу программу таким образом, чтобы она рисовала нам следующую картинку:

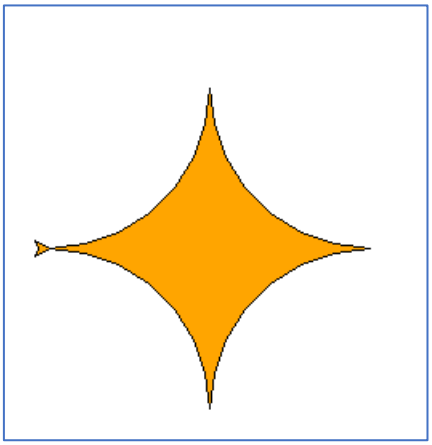


```
from turtle import *
leo = Turtle()
colors = ['red', 'orange', 'pink', 'blue']
leo.width(2)
radius = 140  # стартовый радиус окружности
leo.speed(10)
for i in range(10):
    leo.begin_fill()
```

```
    leo.fillcolor(colors[i % 4])  # установка цвета заливки
    leo.circle(radius)
    leo.end_fill()
    radius -= 10
mainloop()
```

Здесь появляется новая команда **fillcolor**(цвет) – цвет заливки. Цвет пера при этом не меняется.

Теперь научимся рисовать дуги. Давайте напишем программу, которая будет рисовать нам следующее изображение



```
from turtle import *
leo = Turtle()
leo.speed(10)
leo.fillcolor('orange')
leo.begin_fill()
for i in range(4):
    leo.circle(100, 90)  # рисование дуги радиусом 100 и углом 90 градусов
    leo.right(180)
leo.end_fill()
mainloop()
```

Здесь мы снова использовали команду circle, только передали ей не один параметр, а два.

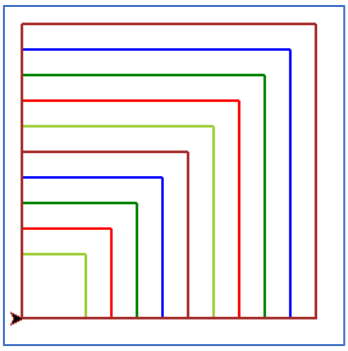
- **circle**(r, a) – рисование дуги. Рисование дуги радиусом r и углом, равным a. Отсчет угла идет из текущей позиции исполнителя. Аналогично радиус может быть как положительным, так и отрицательным. При положительном радиусе дуга рисуется против часовой стрелки, а при отрицательном по часовой стрелке.

В заключении хотелось бы сказать, что у библиотеки turtle существует ещё множество команд, которые позволяют рисовать более сложные изображения. С ними вы можете ознакомиться в документе, который лежит у вас рядом с этим файлом, а также можно прочитать официальную документацию. Также, если вы хотите посмотреть видо, посвященные использованию библиотеки turtle, то можете посмотреть следующий плейлист.

Задачи

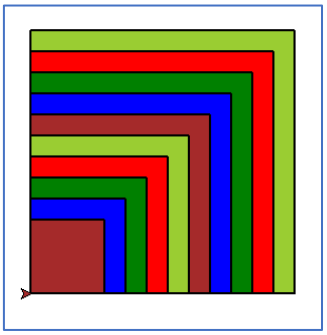
№ 1

Нарисуйте следующее изображение (количество цветов и квадратов может быть другим, главное, чтобы был принцип вложенности кубиков).



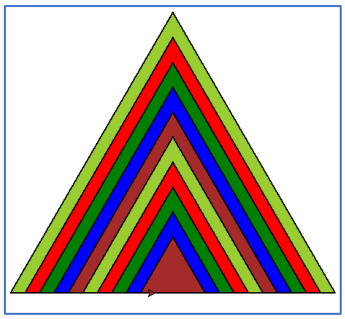
№ 2

Доработайте результат задачи № 1, чтобы она выводила следующую картинку



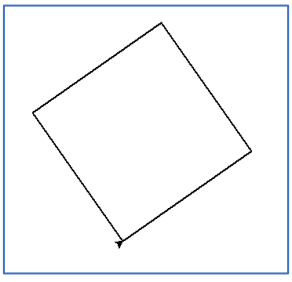
№3

Напишите алгоритм, рисующий следующий рисунок:



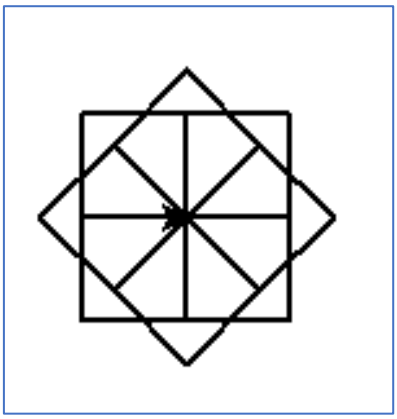
№4

Нарисуйте квадрат, стороны которого не параллельны осям координат



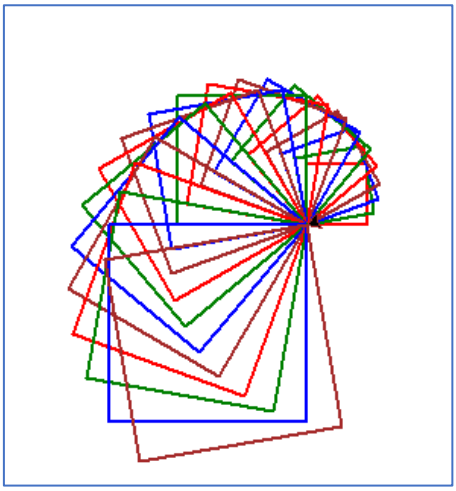
№ 5

Нарисуйте следующую картинку (рисуется 8 одинаковых квадратов. Как только нарисовали очередной квадрат, повернули черепашку на 45 градусов)



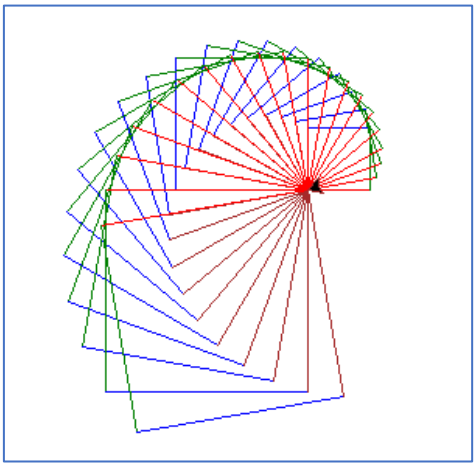
№6

Доработайте предыдущую программу таким образом, чтобы она рисовала следующий рисунок (угол поворота при рисовании следующего квадрата установите равным 10 градусов, количество отрисованных квадратов разного размера – произвольное, цвета квадратов также произвольные)



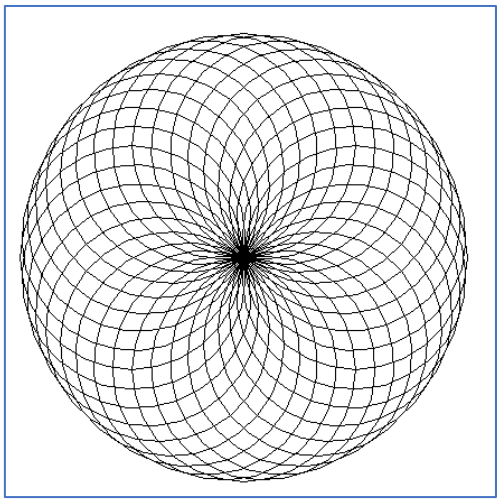
№7

Доработайте предыдущую программу, чтобы она выдавала следующий результат (стороны квадрата рисуются разными цветами)



№8

Измените код предыдущей программы так, чтобы в результате черепашки рисовала следующее изображение (**количество окружностей** – 36, **угол поворота черепашки** – 10 градусов). Окружности также можете сделать разноцветные



№ 9*

Нарисуйте домик. Дом должен иметь крышу, хотя бы одно окно, дверь. При желании можете добавить свои элементы. Ниже приведён пример дома.