Python

Turtle Module

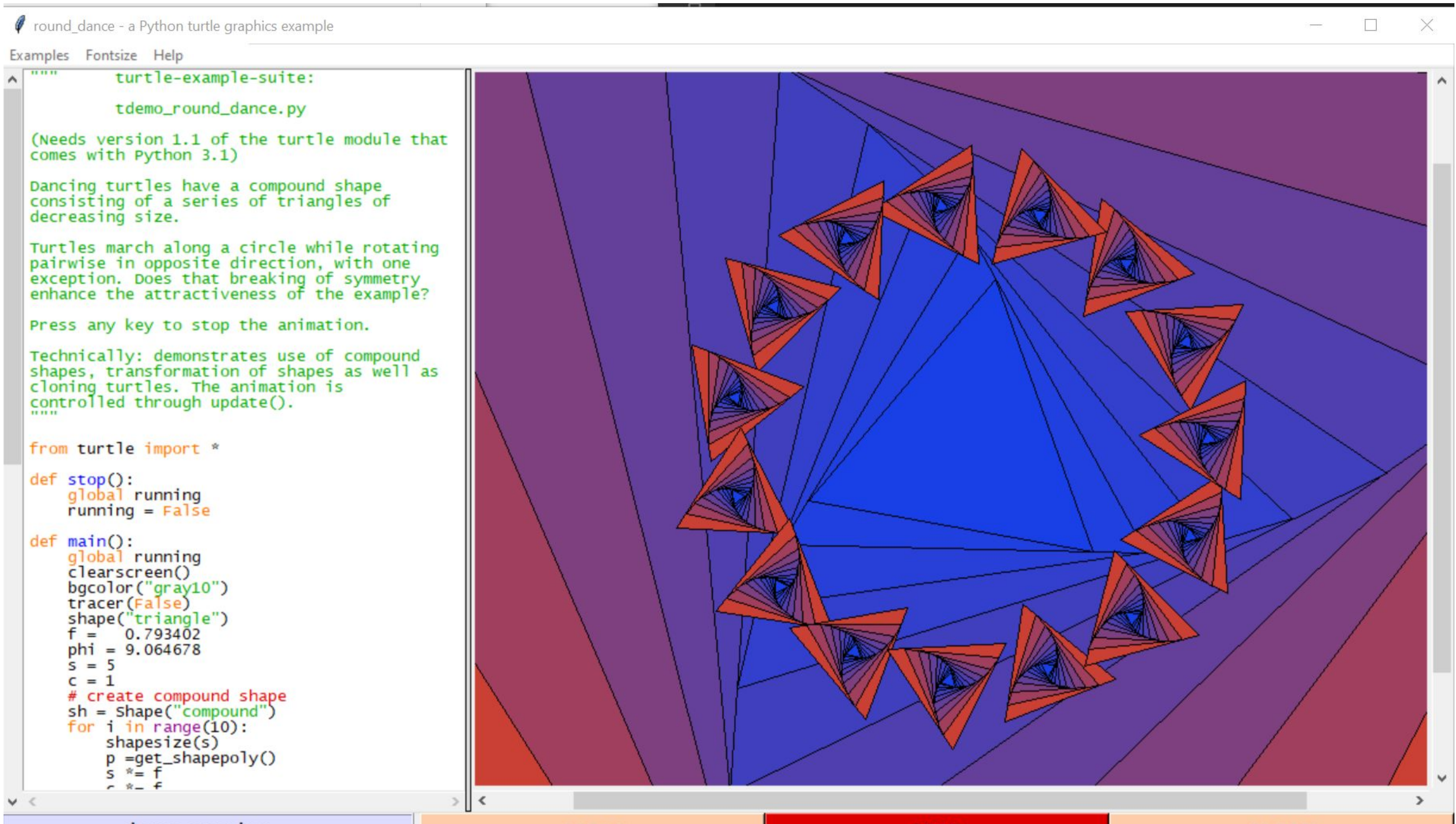
`turtledemo` — Demo scripts

Let’s start with some play by running a set of demo scripts.

https://docs.python.org/3/library/turtle.html#module-turtledemo

To run

```
python -m turtledemo
```

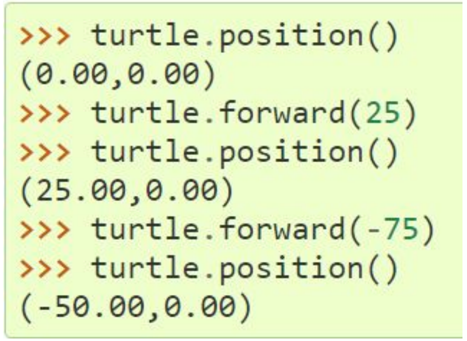


```
"""      turtle-example-suite:

         tdemo_round_dance.py

(Needs version 1.1 of the turtle module that
comes with Python 3.1)

Dancing turtles have a compound shape
consisting of a series of triangles of
decreasing size.

Turtles march along a circle while rotating
pairwise in opposite direction, with one
exception. Does that breaking of symmetry
enhance the attractiveness of the example?

Press any key to stop the animation.

Technically: demonstrates use of compound
shapes, transformation of shapes as well as
cloning turtles. The animation is
controlled through update().
"""

from turtle import *

def stop():
    global running
    running = False

def main():
    global running
    clearscreen()
    bgcolor("gray10")
    tracer(False)
    shape("triangle")
    f =   0.793402
    phi = 9.064678
    s = 5
    c = 1
    # create compound shape
    sh = Shape("compound")
    for i in range(10):
        shapesize(s)
        p =get_shapepoly()
        s *= f
        c *= f
```

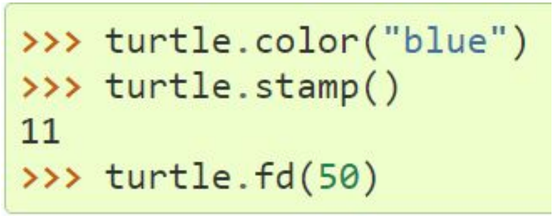
Move and draw

First execute ‘python’ and then import turtle!

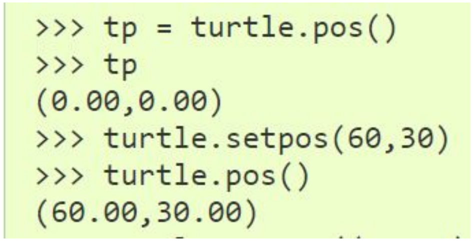
```
forward() | fd()
backward() | bk() | back()
right() | rt()
left() | lt()
goto()
setx(), sety()
setheading()
home()
circle()- turtle.circle(50)
dot() - turtle.dot(20, "blue")
stamp()
clearstamps()
undo()
speed()
```



```
>>> turtle.position()
(0.00,0.00)
>>> turtle.forward(25)
>>> turtle.position()
(25.00,0.00)
>>> turtle.forward(-75)
>>> turtle.position()
(-50.00,0.00)
```



```
>>> turtle.color("blue")
>>> turtle.stamp()
11
>>> turtle.fd(50)
```



```
>>> tp = turtle.pos()
>>> tp
(0.00,0.00)
>>> turtle.setpos(60,30)
>>> turtle.pos()
(60.00,30.00)
```

Speed

- “fastest”: 0
- “fast”: 10
- “normal”: 6
- “slow”: 3
- “slowest”: 1

https://docs.python.org/3/library/turtle.html

Tell Turtle’s state

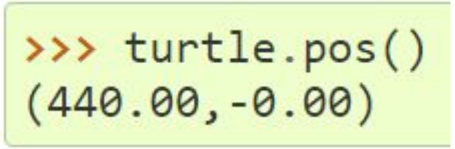
`position()` | `pos()`

`xcor()`

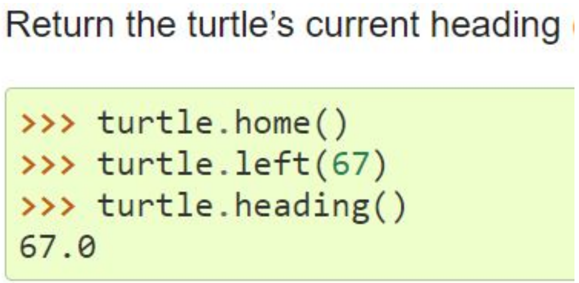
`ycor()`

`heading()`

`distance()`



```
>>> turtle.pos()
(440.00,-0.00)
```



Return the turtle’s current heading

```
>>> turtle.home()
>>> turtle.left(67)
>>> turtle.heading()
67.0
```

Pen control

Pen control

Drawing state

- `pendown()` | `pd()` | `down()`
- `penup()` | `pu()` | `up()`
- `pensize()` | `width()`
- `pen()`
- `isdown()`

Color control

- `color()`
- `pencolor()`
- `fillcolor()`

Filling

- `filling()`
- `begin_fill()`
- `end_fill()`

More drawing control

- `reset()`
- `clear()`
- `write()`

`fillcolor(colorstring)`

Set fillcolor to *colorstring*, which is a Tk color specification string, such as `"red"`, `"yellow"`, or `"#33cc8c"`.

```
>>> turtle.color("black", "red")
>>> turtle.begin_fill()
>>> turtle.circle(80)
>>> turtle.end_fill()
```

Turtle state

Visibility

`showturtle()` | `st()`
`hideturtle()` | `ht()`
`isvisible()`

Appearance

`shape()`
`resizemode()`
`shapesize()` | `turtlesize()`
`shearfactor()`
`settiltangle()`
`tiltangle()`
`tilt()`
`shapetransform()`
`get_shapepoly()`

Methods of TurtleScreen/Screen

Window control

- `bgcolor()`
- `bgpic()`
- `clear()` | `clearscreen()`
- `reset()` | `resetscreen()`
- `screensize()`
- `setworldcoordinates()`

Animation control

- `delay()`
- `tracer()`
- `update()`

Thanks