COMP1021
Introduction to Computer Science

Controlling the Turtle Animation

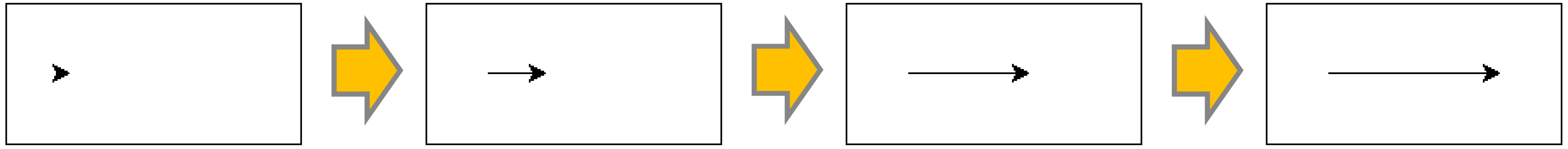
Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Control the speed of the turtle animation using `turtle.speed()`
  2. Turn on/off the turtle animation using `turtle.tracer()`
  3. Update the turtle window, while the animation is turned off, using `turtle.update()`

Controlling the Turtle Animation

- As you know, the turtle, by default, uses animation to show every drawing operation
  - For example, if you tell a turtle to move forward by 100 pixels you will see the turtle gradually move from its original position to the destination position 100 pixels away



Using turtle.speed()

- You can control the speed of the turtle animation using `turtle.speed()`
- It accepts one argument, which is a value from 0 to 10:
  - A value of 0 means ‘as fast as it can’
  - A value from 1 to 10 means an animation speed from very slow (=1) to very fast (=10)
- If you do not provide any value the function will return the current speed of the turtle, for example:

```
current_speed = turtle.speed()
```

An Example Using Speed Control

```
import turtle

def draw(): # Draw a square
    for _ in range(4):
        turtle.left(90)
        turtle.forward(200)

draw()
turtle.reset()
turtle.speed(10)
draw()
turtle.reset()
turtle.speed(1)
draw()
turtle.reset()
turtle.speed(0)
draw()

turtle.done()
```

- This example draws four squares, with each square drawn using a different turtle speed

Draw a square with normal speed (i.e. speed = 3)

Draw a square with a very fast turtle speed of 10 (but not the fastest, which is 0)

Draw a square with the slowest turtle speed

Draw a square with the fastest turtle speed

Using turtle.tracer()

- In case you don't want to see turtle animation at all you can turn it off using `turtle.tracer()`
- The following technique is the fastest possible method to draw something using turtle graphics

```
# Turn off the animation mode
turtle.tracer(False)
```

`False` *means no animation from now on*

. . . *draw whatever you want using the turtle* . . .

```
# Turn the animation mode back on
turtle.tracer(True)
```

`True` *means turning on the animation mode AND updating the screen*

An Example Using turtle.tracer()

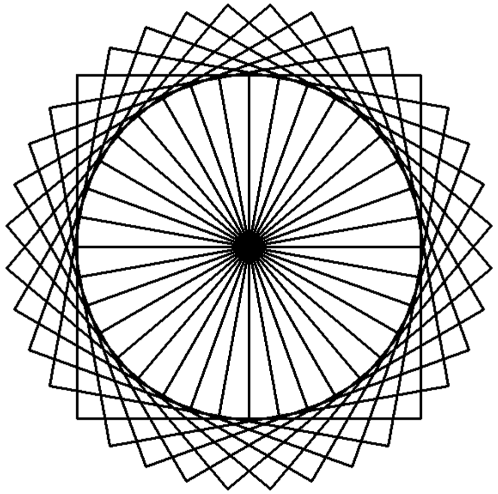
```
import turtle

def draw(): # Draw a square
    for _ in range(4):
        turtle.left(90)
        turtle.forward(200)

turtle.width(3)

turtle.tracer(False)
for _ in range(36):
    draw()
    turtle.left(10)
turtle.tracer(True)

turtle.done()
```

Turn off animation mode

Draw 36 squares around a point

Turn on animation mode and update the screen

- This example shows 36 squares without any turtle animation



Using turtle.update()

- In the previous example, the turtle screen is updated when the turtle tracer is turned on again
- In some situations, you may want to update the turtle screen without turning on the animation
- To do that you use the `turtle.update()` function
- `turtle.update()` is typically used after the tracer has been turned off

An Example Using turtle.update() 1/2

```
import turtle
import time

def draw(): # Draw a square
    for _ in range(4):
        turtle.left(90)
        turtle.forward(200)

turtle.width(3)
turtle.tracer(False)
```

} *Turn off animation mode*

- This example draws 4 squares every half a second without showing animation

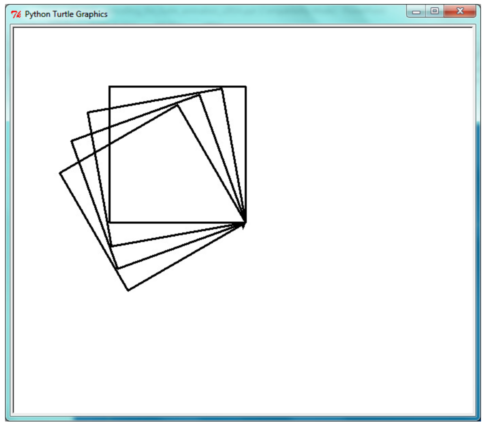
An Example Using turtle.update() 2/2

```
for i in range(36):
    draw()
    turtle.left(10)

    if (i + 1) % 4 == 0:
        # Update the turtle screen
        turtle.update()

        time.sleep(1)

turtle.done()
```

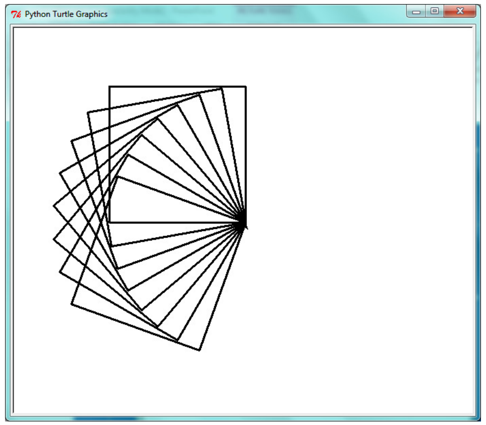
Update the turtle screen after drawing every 4 squares

Wait for a second before showing the next set of squares

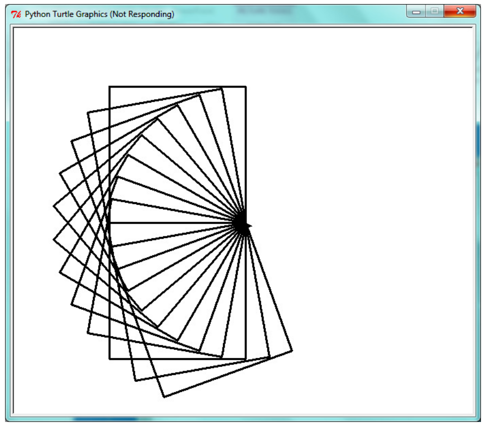
Running the Example



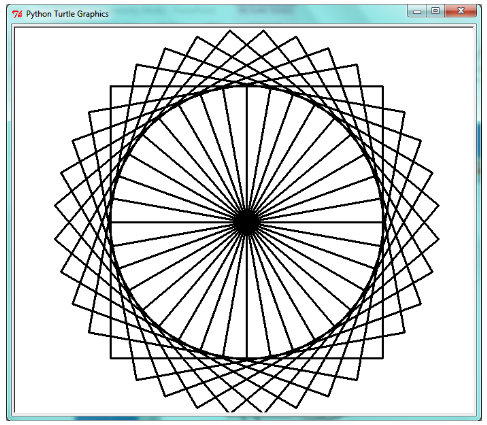
1 second later



1 second later

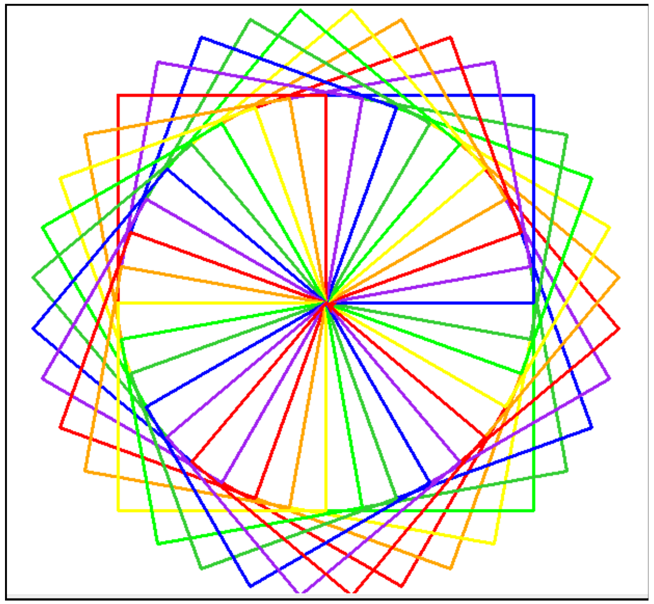


After updating the screen a few times



Extending the Example

- By extending the previous example, we can make a more colourful picture by drawing a set of squares using rainbow colours, like this:



```
def draw_squares():
    colors = ["red", "orange", "yellow", "green",
              "lime green", "blue", "purple"]

    for i in range(36):
        turtle.color(colors[i % len(colors)])
        draw()
        turtle.left(10)
```

Repeat the rainbow colours when drawing the squares

Rotating the Squares

- We then make an animation (not the turtle drawing animation) by rotating the squares using an infinite loop like this:

```
while True:
    turtle.clear()
    draw_squares()
    turtle.update()

    turtle.left(10)
    time.sleep(0.05)
```

Clear the turtle window and draw a new set of squares

Rotate a little bit and wait for a short while before drawing the next set of squares