Transition from Scratch to Python using to Turtle Graphics

Practical Sheet

Contents

1 Turtle Graphics Concepts

Turtle graphics is implemented in lots of languages, notably Logo and in some robot systems. Turtle graphics is available in both Scratch and Python; it is one area of overlap between the two systems, allowing the same problems to set and solved in both a visual and a textual language.

The essential idea of turtle graphics is to draw a picture by moving a 'turtle' around on a paper. The main concepts are:

- **Movement**: the turtle moves and turns. We can see or hide the turtle.
- **Pen**: The turtle holds a pen: the pen has thickness and colour. The turtle draws as it moves. The pen can be up (no drawing) or down (drawing).
- **Location**: The turtle is at an (x, y) location, and facing is some direction

There is a list of Python Turtle functions at the end of this sheet.

The picture drawing is animated – i.e. you watch the turtle moving. This is good for debugging. Animation slows everything down; you can switch it off and go faster.

Although turtle graphics is most associated with patterns, any picture can be drawn. On the other hand complex animation (e.g. games) and graphical user interfaces (i.e. a file saving dialog) is not part of turtle graphics.

2 First Turtle Program

The simplest turtle programs do not require loops, if statements or even variables.

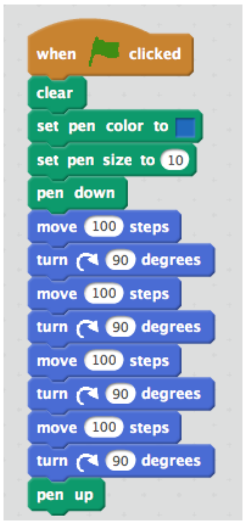
Exercise 1: Copy and run the following program.

```
from turtle import * # Always start with this

# Added code starts here
pencolor('blue')
pensize(10)

forward(100)
right(90)
forward(100)
```

```
right(90)
forward(100)
right(90)
forward(100)
right(90)

# Added code ends here
done()          # Always end with this
```

Exercise 2: Use Scratch to implement the similar program shown here:

What are the differences between the two programs?

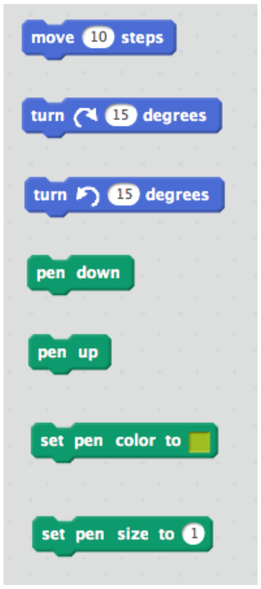


```
when 🏴 clicked
clear
set pen color to 🟦
set pen size to 10
pen down
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
pen up
```

Exercise 3: Adapt the program (in either / both language). Here are some suggestions

- Change the pen colour
- Draw two squares, of different sizes.

Exercise 4: Complete a table of translations from Scratch to Python by writing the Python equivalent again the blocks shown below.



```
move 10 steps
turn ↻ 15 degrees
turn ↺ 15 degrees
pen down
pen up
set pen color to 🟩
set pen size to 1
```

3 Exploring More Turtle

3.1 Filling Shapes

In Python, shapes can be filled. There is no direct equivalent in Scratch

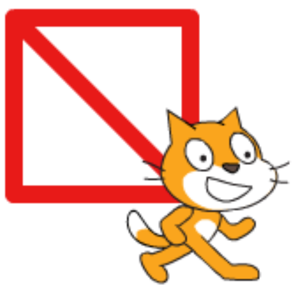
```
from turtle import * # Always start with this

# Added code starts here
pencolor('red')
pensize(5)

fillcolor('yellow')
begin_fill()  # start filling
forward(200)
left(90)
forward(200)
left(90)
forward(200)
left(90)
forward(200)
left(90)
end_fill()    # fill shapes drawn since start

# Added code ends here
done()        # Always end with this
```

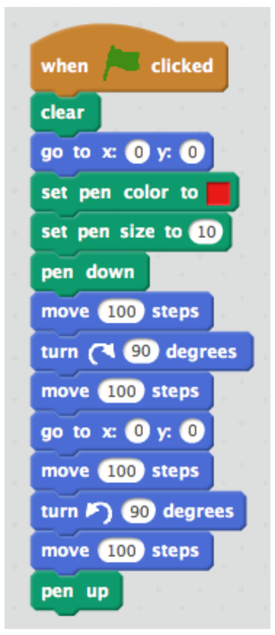
Exercise 5: Use Python to draw and fill one or more different shapes (not just squares) in different colours.

3.2 Co-ordinates and Direction

The turtle has X and Y co-ordinates. In the simplest use of turtle this is ignored, but more complex command allow the position to be set and queried.

Consider the following program (on the right) which draws a square with a diagonal.





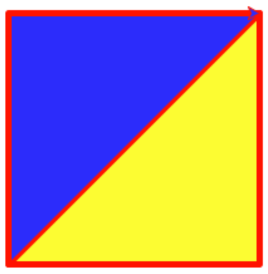
```
when ⚑ clicked
clear
go to x: 0 y: 0
set pen color to 🟥
set pen size to 10
pen down
move 100 steps
turn ↻ 90 degrees
move 100 steps
go to x: 0 y: 0
move 100 steps
turn ↺ 90 degrees
move 100 steps
pen up
```

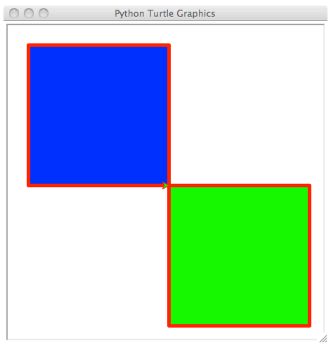
Exercise 6: Explain why it is harder to draw this using just left, right and forward. Is it possible?

Exercise 7: Draw the diagonal line program in Python. You may also read about the following command in the appendix of this sheet (you don't need them all for this example):

- xcor(), ycor()
- goto(x, y)
- heading(), setHeading()
- distance(), towards()

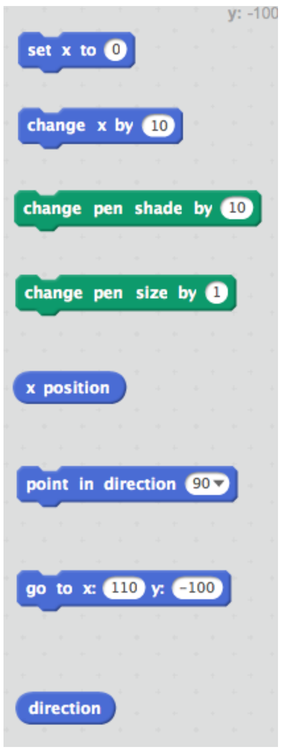
In Python the shapes can be filled as shown below. Consider the other picture: is goto() need to draw this conveniently?





__Exercise 8__: Suggest some pictures that require the use of co-ordinates. Create problems that can be solved in either Python or Scratch.

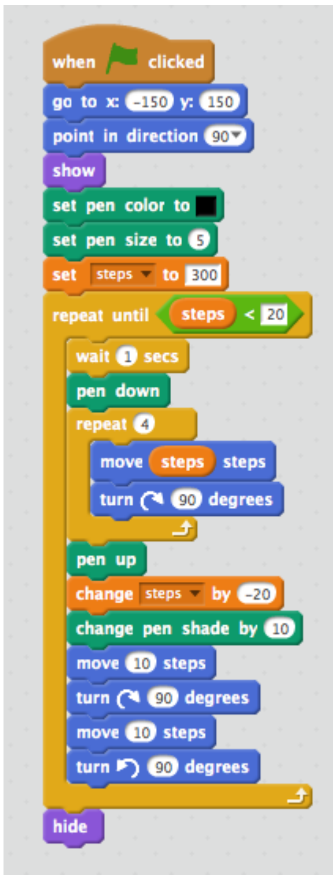
__Exercise 9__: Add the following blocks to the table of Python equivalents. Write the Python translation alongside.



y: -100
set x to 0
change x by 10
change pen shade by 10
change pen size by 1
x position
point in direction 90
go to x: 110 y: -100
direction

4 Control Structures in Python and Scratch

Many programs written using Python Turtle can also be implemented in scratch. Here is an example of a translation from scratch to python.

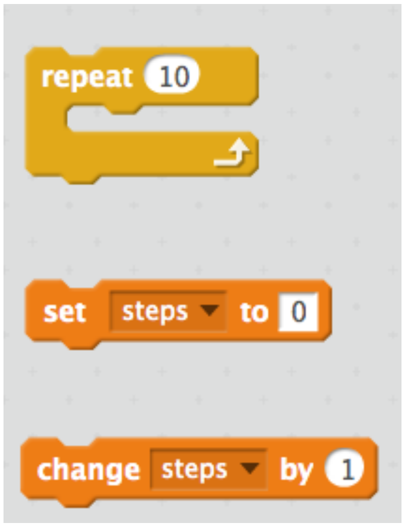


```
when clicked
go to x: -150 y: 150
point in direction 90
show
set pen color to
set pen size to 5
set steps to 300
repeat until steps < 20
    wait 1 secs
    pen down
    repeat 4
        move steps steps
        turn 90 degrees
    pen up
    change steps by -20
    change pen shade by 10
    move 10 steps
    turn 90 degrees
    move 10 steps
    turn 90 degrees
hide
```

```
from turtle import *

penup()
goto(-150, -150)
setheading(90)
colormode(1.0)

r = 0
pencolor(r,0,0)

pensize(5)
steps = 300
while steps > 20:
    pendown()
    for x in range(0,4):
        forward(steps)
        right(90)
    penup()
    steps = steps - 20
    r = r + 0.05
    pencolor(r, 0, 0)
    forward(10)
    right(90)
    forward(10)
    left(90)

done()
```

Exercise 8: Implement the two versions of the program.

Exercise 9: Examine the differences between the programs (other than filling and pen colours). Complete a table of equivalents:



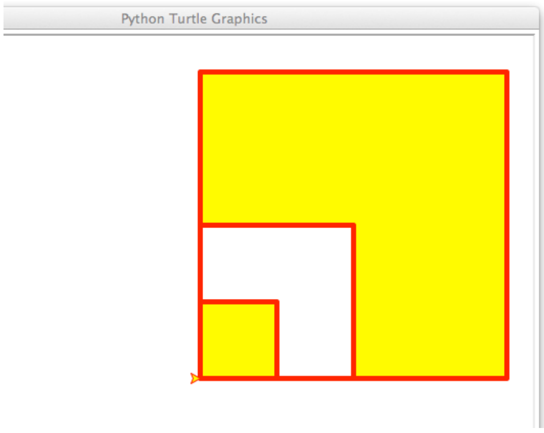
```
repeat 10

set steps to 0

change steps by 1
```

5 Using Functions in Turtle Programs

Because it takes quite a lot of commands to do anything, turtle graphics is good for teaching functions.

Exercise 10: Copy the following function and call it in a program to draw 3 or more squares of different sizes, as shown opposite.

```
def square(side):
    forward(side)
    left(90)
    forward(side)
    left(90)
    forward(side)
    left(90)
    forward(side)
    left(90)
```



Exercise 11: Create the square function in Scratch.

Exercise 12: More functions. Try the following enhancements:

- Enhance the square function to have a colour argument and fill the square with the given colour.
- Create a rectangle function.
- Create a similar function but instead of a square draw a regular polygon with a given number of sides, each of a given length.

5.1 Polygons

The square function shown above can be rewritten using a loop. Two possible versions are shown below:

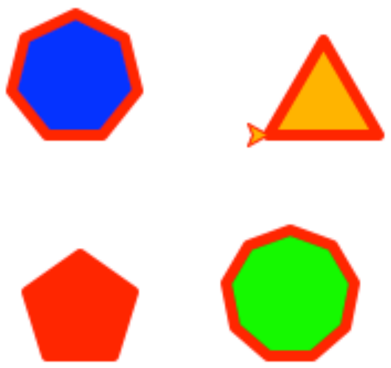
```
def squareL(side):
    for x in range(0,4):
        forward(side)
        left(90)
```

```
def squareL(side):
    count = 0
    while count < 4:
        forward(side)
        left(90)
        count = count + 1
```

Exercise 13: Create a similar function in Scratch.

Exercise 14: Using either Python or Scratch, write a function polygon that draws a N-sided regular polygon. Since the number of sides varies, this function requires the use of a loop. Remember that for N sides, turn 360/N (so when N is 4, turn 90, N is 5 turn 72, N is 6 turn 60 etc). Here is an example of using this function:

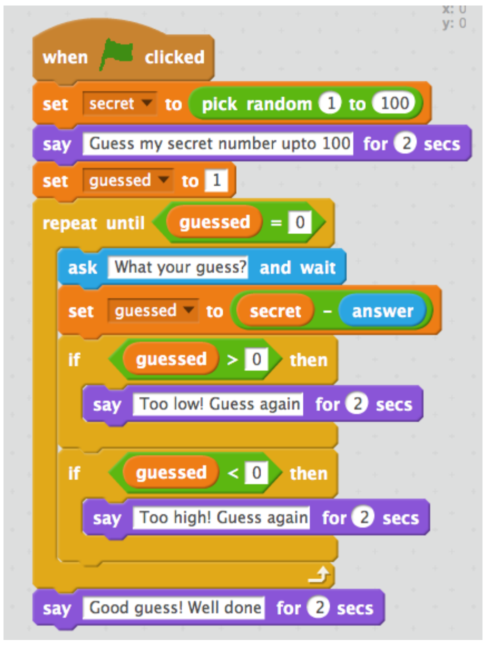
```
pencolor('red')
pensize(5)
penup()
goto(-50, -50)
pendown()
polygon(5, 30, "red")
penup()
goto(-50, 50)
pendown()
polygon(7, 25, "blue")
penup()
goto(50, -50)
pendown()
polygon(9, 20, "green")
penup()
goto(50, 50)
pendown()
polygon(3, 50, "orange")
```



6 Non Turtle Programs in Scratch and Python

Some programs that do not use the pen and turtle graphics can also be written in the same way in Scratch and Python, though obviously this is not true of all programs.

Exercise 15: Number Guessing: translate the following Scratch programming into Python. What are the additional language features that this example introduces? Complete another translation table.



```
when [flag] clicked
set secret to pick random 1 to 100
say Guess my secret number upto 100 for 2 secs
set guessed to 1
repeat until guessed = 0
    ask What your guess? and wait
    set guessed to secret - answer
    if guessed > 0 then
        say Too low! Guess again for 2 secs
    if guessed < 0 then
        say Too high! Guess again for 2 secs
say Good guess! Well done for 2 secs
```

Exercise 16: Suggest other problems that could be implemented in both Scratch and Python.

7 Appendix: Turtle Graphics Function Reference

The Python turtle graphics package has both

- A functional interface
- An object-oriented interface

We will use the functional interface; a few capabilities are not available as a result. The full documentation is found in chapter 23 of the Python standard library.

Function	Description and Example
Move and Draw	
forward(), backward()	Move the turtle a distance: `forward(10)`
right(), left()	Turn by an angle: `right(90)`
goto()	Move to an (x, y) position: `goto(10, 50)`
home()	Move to the home position: `home()`
circle()	Draw a circle or arc. Examples: • `circle(100)` – draw a circle of radius 100 • `circle(50, 90)` – draw an arc of radius 50, angle 90
dot()	Draw a dot with diameter & colour: `dot(20, 'blue')`
Turtle Position	
setheading()	Set the direction of the turtle: `setheading(90)`. In standard mode, 0 is East, 90 is North, 180 is West and 270 is South.
heading()	Get the heading: `angle = heading()`
xcor(), ycor()	Get the x or y coordinates: `currentX = xcor()`
distance()	Calculate the distance from the current position to some point: `dist = distance(50, 75)`
towards()	Calculate the angle from the current position to the given co-ordinates: `towards(100, 100)`
Pen and Turtle	
pendown(), penup()	Put the pen down / up. When the pen is down, moving the turtle draws a line.
pensize()	Set the pen size: `pensize(10)` for a thick pen.
isdown()	Returns true if the pen is up.
showturtle(), hideturtle()	Show or hide the turtle. It is faster to draw without the turtle visible. (*Note: it is also possible to switch off animation altogether*)
pencolor()	Set the pen colour: `pencolor('green')`

Function	Description and Example
Filling and Clearing	
fillcolor()	Set the fill colour; colours most easily entered as string though other formats supported: `fillcolor('blue')`.
filling()	Test whether shapes being drawn are to be filled.
begin_fill(), end_fill()	These command bracket drawing commands in which shapes should be filled.
reset()	Reset everything.
clear()	Clear the picture but do not reset the turtle.
write()	Write a text string: `write("Hello")` writes to the current position (which does not change) and aligns the string so that the turtle is at the left hand of the string.
Screen	
bgcolor()	Set the background colour of the screen: `bgcolor('pink')`
bgpic()	Set a picture as the background, using a file name.
screensize()	Set the screen size to e.g. 500 width and 300 high: `screensize(500, 300)`
textinput	Input text using a dialog: `textinput("Title", "Prompt")`
numinput	Input a number
window_height, window_width	Get the window height or width