Turtle Graphics Crib Sheet

This is a summary of the Turtle graphics commands, some of which you have used before and some of which will be new to you.

For each of these, the parameters (the stuff in brackets) represents a value that you need to supply – either as a number or as a variable. For example: forward(*distance*) means that you need to replace *distance* with a number or a variable.

For angles, remember that 90degrees is a right-angle, 180degrees is a straight line and 360degrees takes you full-circle to where you started from.

```
# move relative to where the turtle is now
forward(distance)  # or: fd(distance)
backward(distance) # or: bk(distance) back(distance)
right(angle)       # or: rt(angle)
left(angle)        # or: lt(angle)

# move absolute (0,0 is the centre of the screen)
goto(xpos, ypos)   # or: setpos(xpos, ypos) setposition(xpos, ypos)
setx(xpos)         # move to given x position while keeping y position unchanged
sety(ypos)         # move to given y position while keeping x position unchanged
setheading(angle)  # or seth(angle). 0 means right/east. 90 means up/north
home()             # move back to the centre of the screen and point right
                   # ..leaves a trail if isdown()
clear()            # clear the screen. The turtle stays where it was.
reset()            # do home() then clear()

# shapes
circle(radius)
circle(radius, extent) # extent is an angle: 90 will draw a quarter-circle
circle(radius, extent, steps) # draw a regular polygon with steps sides
dot(size, colour)  # more on colours below..
stamp()            # stamp the turtle shape at the current position. More below..

# miscellaneous
undo()             # undo the last action - can be called repeatedly
speed(speed)       # 1 is slowest, 6 is normal, 10 is fast. 0 is fastest (yes, 0)
reset()            # clear the screen, move back to screen centre, point right
clear()
write("Hello")     # print some text at the turtle position
```

You can find out the current position of the turtle (for example, you might store these in variables so that your program can draw something somewhere else then come back to the current position).

```
# find out the turtle's state
(xpos, ypos) = position()          # or: pos()
angle = towards(xpos, ypos)        # the angle between turtle's current position
                                   # and the co-ordinate xpos,ypos
xpos = xcor()                      # like position() but only returns the
                                   # x co-ordinate
ypos = ycor()                      # like position() but only returns the
                                   # y co-ordinate
angle = heading()                  # current direction turtle is facing
how_far = distance(xpos, ypos)     # the distance from turtle's current position
                                   # to the co-ordinate xpos,ypos
```

To start with, the turtle leaves a thin black line behind as it moves. You can change that.

```
# pen control
pendown()         # or: pd() down(). When the pen is down, turtle leaves a trail.
penup()           # or: pu() up(). When the pen is up, turtle leaves no trail.
pensize(width)    # or: width(width). Change the width of the line
isdown()          # return True if the pen is currently down.
pencolor(colour)  # more on colours below..
```

There are three different ways to specify colours:

- By name: "red", "blue", "green", "turquoise", "yellow", "gold"
- With a Red/Green/Blue hex triplet, for example: pencolor("#ff8040") (ask me for more details..)
- With a Red/Green/Blue tuple, for example: pencolor(1, 0.5, 1.0). Each of those three numbers must be between 0 and 1. The first is the amount of red, the second is the amount of green and the third is the amount of blue. (1,0,0) is red, (1,1,1) is white (0,0,0) is black, (1,1,0) is yellow (ask me for more details..)

To start with, the turtle is a narrow arrow shape; you can see the shape and the direction in which it is pointing. You can change that.

```
# shapes
showturtle()      # or: st(). Make turtle visible
hideturtle()      # or: ht(). Make turtle invisible
isvisible()       # return True if turtle is currently visible
shape(newshape)   # newshape is the new shape for turtle
getshapes()       # list the shapes that can be used for shape()
```

You can use your turtle shape as a "stamp". You won't notice the stamp until you move the turtle elsewhere...

```
# stamp
stamp()           # stamp here
fred=stamp()      # stamp here and remember where the stamp was
clearstamp(fred)  # delete the stamp made at location fred
clearstamps()     # delete all stamps
clearstamps(1)    # delete the last stamp
clearstamps(3)    # delete the last 3 stamps
```

If you create a closed shape you can fill it in...

```
# fill
fillcolor(colour) # set the fill colour to colour
color(colour)     # set the pencolor and fillcolor to colour
begin_fill()      # call this before starting to draw a shape to be filled
end_fill()        # fill the shape
filling()         # return True if currently filling
```

You can have several turtles on the screen at the same time and control them independently. Here's an example...

```
# multiple turtles
t1 = Turtle()
t2 = Turtle()
t1.shape("turtle")
t2.shape("square")
t1.forward(100)
t2.left(90)
t2.forward(50)
```

This is almost (but not quite) everything you can do. Here are a couple of little sample programs.

```
# red and yellow star, from the Python Turtle documentation
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```

```
# squares
from turtle import *

penup()
color('red', 'yellow')

# draw a square using current turtle direction
def do_square(x,y,size):
    begin_fill()
    penup()
    goto(x,y)
    pendown()
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)
    # back where I started
    end_fill()

# set of colours to use
colours = ["#c0c0c0", "#808080", "#404040", "#202020", "#101010", "red",
"orange", "yellow", "green", "blue", "#9000f3", "violet"]

# draw some coloured squares
for i in range(0,12):
    color('red', colours[i])
    do_square(45*i-200, 45*i-200, 40)
```

```
# draw some flowers
from turtle import *
from random import *

def petal(length):
    # remember where I started and what direction I was pointing in
    orient=heading()
    posn=pos()
    pendown()
    forward(length)
    left(90)
    forward(length/20)
    left(180)
    forward(2*length/20)
    penup()
    # Go back to where I started
    setpos(posn)
    setheading(orient)

def flower(x, y, length, petals):
    for pet in range(0,petals):
        goto(x, y)
        petal(length)
        right(360/petals)

# set of colours to use
colours=["red", "green", "blue", "yellow", "orange", "violet", "turquoise"]
speed(11)
penup()

for i in range(0,50):
    x=randint(-400, 400)
    y=randint(-400, 400)
    length=randint(15,60)
    numpets=randint(5,20)
    color(colours[i%7])
    flower(x,y,length, numpets)
```

Installing Python at home

Python is *completely free* to download and to use. If you want to install it on a computer at home (if it's not yours, ask the owner's permission first) you can download it from http://www.python.org. Go to the "downloads" page and look for a link like "Latest Python 3 Release Python 3.4.1". If you try to do this and get stuck, ask me.