

Getting Started

Self-Review Questions

Self-review 1.1 Why is the following the case?

```
>>> print 4 + 9
13
>>>
```

Because the Python system evaluates the expression $4 + 9$ which means adding 4 to 9 giving the result 13 which is the printed out.

Self-review 1.2 Why is the following the case – rather than the result being 13?

```
>>> print "4 + 9"
4 + 9
>>>
```

Because "4+9" is a string and so it is printed as is, there is no expression to be evaluated.

Self-review 1.3 Why does the expression $2 + 3 * 4$ result in the value 14 and not the value 24?

Because the $*$ operator has higher precedence than the operator $+$. The $3 * 4$ sub-expression is evaluated first giving 12 and then the expression $2 + 12$ is evaluated resulting in 14.

Self-review 1.4 What is a module and why do we have them?

A module is a collection of bits of program that provide a useful service.

Self-review 1.5 What is a library and why do we have them?

The answer "A place where books are held, so that they can be borrowed" is clearly correct but not the one wanted. The answer wanted is "A collection of bits of software that we can make use of in our programs."

Self-review 1.6 What does the statement `turtle.demo ()` mean?

The statement causes the function `demo` from the `Turtle` module to be executed.

Self-review 1.7 What is an algorithm?

It is a procedure/process for achieving a desired goal.

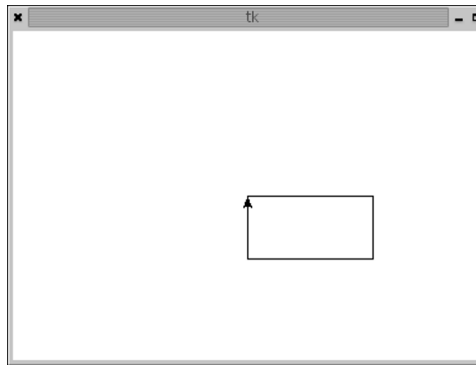
Self-review 1.8 Why are algorithms important in programming?

Programming is all about implementing algorithms in software so algorithms are critical and core to programming.

Self-review 1.9 What shape does the following program draw?

```
turtle.forward ( 100 )
turtle.right ( 90 )
turtle.forward ( 50 )
turtle.right ( 90 )
turtle.forward ( 100 )
turtle.right ( 90 )
turtle.forward ( 50 )
```

A rectangle 100 unit wide and 50 units deep:



Self-review 1.10 The Turtle module provides a function to draw circles. If this function was not available, what algorithm would you use to draw a circle?

Draw a large number of very short straight lines changing the direction of travel by 360° divided by the number of lines. So if the number is 3 we have a triangle. If the number is 4 we have a square. If the number is 5 we have a pentagon, with 6 a hexagon. If we have 100 sides then for a small 'circle' it may not be noticeable that the lines are straight.

Self-review 1.11 What does the function `turtle.goto` do?

Hint: Use the Python help system to help you!

Help on function `goto` in `turtle`:

```
turtle.goto = goto(*args)
Go to the given point.
```

If the pen is down, then a line will be drawn. The turtle's orientation does not change.

Two input formats are accepted:

```
goto(x, y)
go to point (x, y)
```

```
goto(x, y)
go to point (x, y)
```

```
Example:
>>> turtle.position()
[0.0, 0.0]
>>> turtle.goto(50, -45)
>>> turtle.position()
[50.0, -45.0]
```

Self-review 1.12 What is `stdin`?

Self-review 1.13 What is ‘script mode’?

Programming Exercises

Exercise 1.1 Estimate the size of your computer screen by importing the Turtle module and causing the turtle to move 1 pixel (though you may want to try 10 pixels!).

Using the program:

```
import turtle

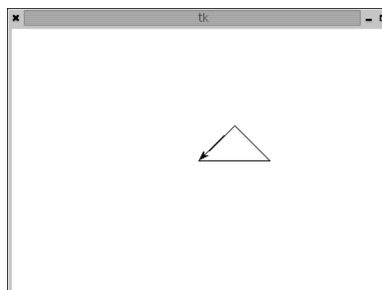
turtle.forward ( 1279 )
turtle.right ( 90 )
turtle.forward ( 512 )

raw_input ( 'Press return to terminate the program: ' )
```

and the window manager to resize and reposition the drawing window, we saw that the screen width was 1280. By making the window wider than the screen and then locating one end of the horizontal line at one side of the screen, we saw that the down line was right on the other edge of the screen. This trick doesn’t work so easily for vertical measurement. Hence we drew half what we thought was the screen height and saw whether it was half the height. It was so we estimate the height to be 1024.

Exercise 1.2 Experiment with the ‘square spiral’ program, varying the lengths of various lines. Which values make spirals and which do not? Which values make the nicest-looking spiral?

Exercise 1.3 Write a program to draw a right-angled triangle looking something like:



```
import turtle
```

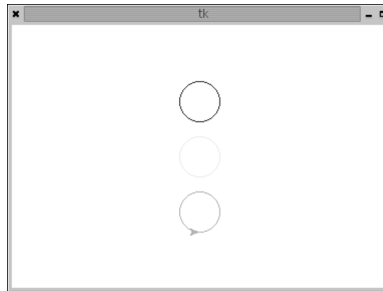
```
turtle.forward ( 70 )
turtle.left ( 135 )
turtle.forward ( 50 )
turtle.left ( 90 )
turtle.forward ( 50 )
```

```
# Prompt to stop the program from terminating and hence the display from disappearing.
```

```
raw_input ( 'Press return to terminate the program: ' )
```

Exercise 1.4 Write a program to draw a red circle, then a yellow circle underneath it and a green circle underneath that. The result should look something like:

It is true that in this monochrome rendition, the colors just appears in various shades of grey, but the colors will work on your computer – unless you have a monochrome screen!



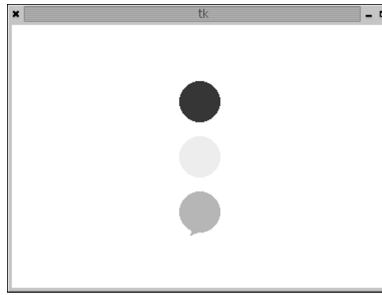
Hint: We used the function `turtle.goto` as well as the functions `turtle.up`, `turtle.down`, `turtle.color` and `turtle.circle` in our solution.

```
import turtle
```

```
turtle.up ( )
turtle.goto ( 0 , 35 )
turtle.down ( )
turtle.color ( 'red' )
turtle.circle ( 20 )
turtle.up ( )
turtle.goto ( 0 , -20 )
turtle.down ( )
turtle.color ( 'yellow' )
turtle.circle ( 20 )
turtle.up ( )
turtle.goto ( 0 , -75 )
turtle.down ( )
turtle.color ( 'green' )
turtle.circle ( 20 )
```

```
raw_input ( 'Press return to terminate the program: ' )
```

Exercise 1.5 Amend your program from the previous question so that the circles are filled and hence the result looks something like:



These circles are definitely filled with the right color when we run the program on our computers even though they just look greyish in this book.

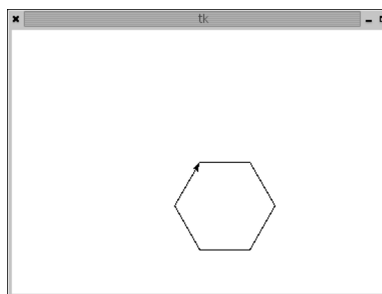
Hint: We used the function `turtle.fill` in our solution.

```
import turtle

turtle.up ()
turtle.goto ( 0 , 35 )
turtle.down ()
turtle.color ( 'red' )
turtle.fill ( 1 )
turtle.circle ( 20 )
turtle.fill ( 0 )
turtle.up ()
turtle.goto ( 0 , -20 )
turtle.down ()
turtle.color ( 'yellow' )
turtle.fill ( 1 )
turtle.circle ( 20 )
turtle.fill ( 0 )
turtle.up ()
turtle.goto ( 0 , -75 )
turtle.down ()
turtle.color ( 'green' )
turtle.fill ( 1 )
turtle.circle ( 20 )
turtle.fill ( 0 )

raw_input ( 'Press return to terminate the program: ' )
```

Exercise 1.6 Write a program to draw a regular hexagon. The result should look something like:



Hint: Regular hexagons have six sides, of equal length, which meet at an internal angle of 120° .

```
import turtle
```

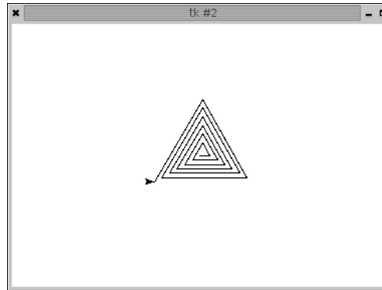
```
turtle.forward ( 50 )
turtle.right ( 60 )
turtle.forward ( 50 )
turtle.right ( 60 )
turtle.forward ( 50 )
turtle.right ( 60 )
turtle.forward ( 50 )
turtle.right ( 60 )
turtle.forward ( 50 )
turtle.right ( 60 )
turtle.forward ( 50 )
```

```
raw_input ( 'Press return to terminate the program: ' )
```

Exercise 1.7 Write a program to draw the first letter of your name.

Challenges

Challenge 1.1 Write a program to create a triangular spiral, as in:



Hint: This is easiest if you work with equilateral triangles – it gets very complex otherwise.

```
import turtle
```

```
turtle.forward ( 10 )
turtle.left ( 120 )
turtle.forward ( 15 )
turtle.left ( 120 )
turtle.forward ( 20 )
turtle.left ( 120 )
turtle.forward ( 25 )
turtle.left ( 120 )
turtle.forward ( 30 )
turtle.left ( 120 )
turtle.forward ( 35 )
turtle.left ( 120 )
turtle.forward ( 40 )
turtle.left ( 120 )
turtle.forward ( 45 )
```

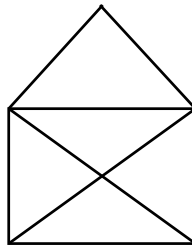
```

turtle.left ( 120 )
turtle.forward ( 50 )
turtle.left ( 120 )
turtle.forward ( 55 )
turtle.left ( 120 )
turtle.forward ( 60 )
turtle.left ( 120 )
turtle.forward ( 65 )
turtle.left ( 120 )
turtle.forward ( 70 )
turtle.left ( 120 )
turtle.forward ( 75 )
turtle.left ( 120 )
turtle.forward ( 80 )
turtle.left ( 120 )
turtle.forward ( 85 )
turtle.left ( 120 )
turtle.forward ( 90 )
turtle.left ( 120 )
turtle.forward ( 95 )
turtle.left ( 120 )

raw_input ( 'Press return to terminate the program: ' )

```

Challenge 1.2 Write a program to draw this 'envelope' figure:



Hint: You may want to start by doing some trigonometry to sort out all the angles and lengths. Pythagoras' Theorem will almost certainly come in useful: Pythagoras' Theorem states that for a right-angled triangle:

$$\text{hypotenuse} = \sqrt{x^2 + y^2}$$

where hypotenuse is the longest side of the triangle and x and y are the lengths of the other two sides.

Hint: To find the square root of a number in Python, you need to import the module `math` and call the function `math.sqrt`: to find the square root of 2, you import the `math` module, then write `math.sqrt(2)`.

```
#!/usr/bin/env python
```

```
import turtle
import math
```

*# Assume the house body is a square and the roof is a right-angled triangle (half a square).
We
can then get the various diagonals using Pythagoras' Theorem with all angles being multiples
of 45.*

It seems almost totally unreasonable not to use variables to solve this problem.

```
sideLength = 40
halfDiagonalLength = 0.5 * math.sqrt ( 2 * ( sideLength ** 2 ) )
```

Create the right orientation.

```
turtle.left ( 90 )
```

Draw the house.

```
turtle.forward ( sideLength )
turtle.right ( 135 )
turtle.forward ( halfDiagonalLength )
turtle.right ( 90 )
turtle.forward ( halfDiagonalLength )
turtle.left ( 135 )
turtle.forward ( sideLength )
turtle.left ( 135 )
turtle.forward ( halfDiagonalLength )
turtle.right ( 90 )
turtle.forward ( halfDiagonalLength )
turtle.left ( 135 )
turtle.forward ( sideLength )
turtle.right ( 135 )
turtle.forward ( halfDiagonalLength )
turtle.right ( 90 )
turtle.forward ( halfDiagonalLength )
turtle.right ( 45 )
turtle.forward ( sideLength )
```

Prompt to stop the program from terminating and hence the display from disappearing.

```
raw_input ( 'Press return to terminate the program: ' )
```

Challenge 1.3 Write a program to draw the following figure:

