2

The Fundamentals

Self-Review Questions

Self-review 2.1 What are the types of the following literals? 1, 1., 1.0, ”1”, ”1.”, ”1.0”, ’1’, ’1.’, 100000000000000000, 100000000000000000., 100000000000000000.0.

int, float, float, str, str, str, str, str, long, float, float.

Self-review 2.2 Explain the difference between the following two statements:

```
print 12
print "12"
```

The first print the integer literal 12, the second prints the string comprising the character 1 followed by the character 2.

Self-review 2.3 Highlight the literals in the following program:

```
a=3
b='1'
c=a-2
d=a-c
e="dog"
f=', went to mow a meadow.'
g='man'
print a,g,"and his",e,f,a,g,",",d,g,",",c,g,"and his",e,f
```

The literals are: 3, ’1’, 2, "dog", ’, went to mow a meadow.’, ’man’, "and his", ",", "and his".

Self-review 2.4 What is the output of the above program?

This is best answered by actually executing the code, even though the question was really about ‘mental execution’, i.e. thinking through what the Python system would do when executing the code.

```
|> python meadowSong.py
3 man and his dog , went to mow a meadow. 3 man , 2 man , 1 man and his dog , went to mow a mea
|>
```

The important issue here is that ‘mental execution’ is an important skill so that you know when an error has occurred and can debug the error.

Self-review 2.5 What are the types of the variables a, b, c, d, e, f, g in the above program?

int, str, int, int, str, str, str

Self-review 2.6 Is there a difference between the *type* of the expressions "python" and 'python'?

Self-review 2.7 What is the result of 4+4/2+2?

8. the 4/2 is evaluated first then the left addition 4+2, then the right addition 6+2.

Self-review 2.8 Add brackets as needed to the above expression to make the answer:

a. 2
b. 5
c. 6

1. $(4+4)/(2+2)$
2. $4+(4/(2+2))$
3. $(4+4)/2+2$

Self-review 2.9 What is iteration?

It is the repeating of some action or sequence of actions.

Self-review 2.10 What is the difference between the expressions c = 299792458 and c = 2.99792458 * 10 ** 8?

In the first c is of type int, in the second, c is of type float. The value represented is the same.

Self-review 2.11 What does this expression n < 140 mean? I.e. what test is undertaken and what is the value and type of the result?

It tests whether the value of the variable n is less than the literal 140 and delivers **True** if it is and **False** if it is not. The result of the expression if of type bool.

Self-review 2.12 What is the value of the variable val after the executing the expression val = 1 / 2? What is the type of val?

Self-review 2.13 What is the value of the variable val after the executing the expression val = 1 / 2.0? What is the type of val?

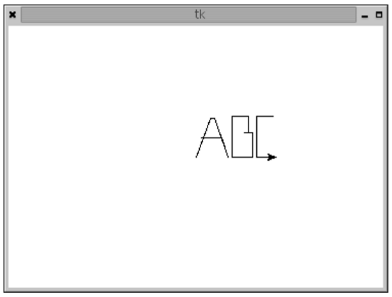
Self-review 2.14 What are the possible values contained in a variable of type bool?

Programming Exercises

Exercise 2.1 The program:

```
import turtle

scale = 4

## Letter A
turtle.down ( )
# Point upwards to begin
turtle.left ( turtle.heading ( ) + 90 )
turtle.right ( 20 )
turtle.forward ( 10 * scale )
turtle.right ( 70 )
turtle.forward ( 1 * scale )
turtle.right ( 70 )
turtle.forward ( 10 * scale )
turtle.backward ( 5 * scale )
turtle.right ( 90 + 20 )
turtle.forward ( 5 * scale )
#Move to right of letter and over 1 * scale
turtle.up ( )
turtle.backward ( 5 * scale )
turtle.left ( 110 )
turtle.forward ( 5 * scale )
turtle.left ( 70 )
turtle.forward ( 1 * scale )

## Letter B
turtle.down ( )
# Point upwards to begin
turtle.left ( turtle.heading ( ) + 90 )
turtle.forward ( 10 * scale )
turtle.right ( 90 )
turtle.forward ( 4 * scale )
turtle.right ( 90 )
turtle.forward ( 4 * scale )
turtle.left ( 90 )
turtle.backward ( 1 * scale )
turtle.forward ( 2 * scale )
turtle.right ( 90 )
turtle.forward ( 6 * scale )
turtle.right ( 90 )
turtle.forward ( 5 * scale )
# Move to right of letter
turtle.up ( )
turtle.right ( 180 )
turtle.forward ( 6 * scale )

## Letter C
turtle.down ( )
# Point upwards to begin
turtle.left ( turtle.heading ( ) + 90 )
turtle.forward ( 10 * scale )
turtle.right ( 90 )
turtle.forward ( 4 * scale )
turtle.backward ( 4 * scale )
turtle.left ( 90 )
turtle.backward ( 10 * scale )
turtle.right ( 90 )
turtle.forward ( 4 * scale )
# Move to right of letter
turtle.up ( )
turtle.forward ( 1 * scale )

# Pause
raw_input ( "Press any key to end." )
```

causes the following when executed:



The code for writing each individual letter is fairly easy to spot, because of the comments. There's a much easier way to divide this code up though: functions. The exercise then is to:

1. Modify the code so that each letter is drawn by a function: drawA ( ) for the letter A, for example.
2. Add the letters "D" and "E" as functions.
3. Amend the code to display the word "DECADE" rather than "ABCDE".

Exercise 2.2 Rewrite your answer to **??** (three colored circles one above the other) using a function.

Hint: The function should probably have three parameters.

```
import turtle

def drawAtWithColor ( x , y , c ) :
    turtle.up ( )
    turtle.goto ( x , y )
    turtle.down ( )
    turtle.color ( c )
    turtle.circle ( 20 )

drawAtWithColor ( 0 , 35 , 'red' )
drawAtWithColor ( 0 , -20 , 'yellow' )
drawAtWithColor ( 0 , -75 , 'green' )

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

Exercise 2.3 Rewrite your answer to **??** (three colored, filled circles one above the other) using a function.

Hint: The answer should be a trivial extension to the answer of the previous question.

```
import turtle

def drawAtWithColor ( x , y , c ) :
    turtle.up ( )
    turtle.goto ( x , y )
    turtle.down ( )
    turtle.color ( c )
    turtle.fill ( 1 )
    turtle.circle ( 20 )
    turtle.fill ( 0 )

drawAtWithColor ( 0 , 35 , 'red' )
drawAtWithColor ( 0 , -20 , 'yellow' )
drawAtWithColor ( 0 , -75 , 'green' )

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

Exercise 2.4 Rewrite your answer to question **??** (drawing a hexagon) using iteration.

```
import turtle

for i in range ( 6 ) :
    turtle.forward ( 50 )
    turtle.right ( 60 )

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

Exercise 2.5 Extend your answer to the previous question so that the number of sides to draw is a parameter – this is a programming solution for **??**. How many sides do you need before it looks like a circle?

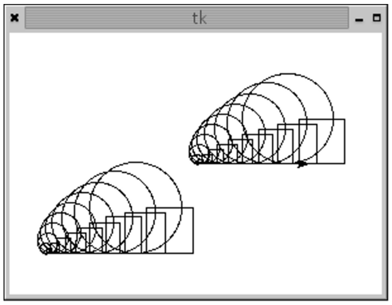
```
import turtle

circumference = 300

sides = input ( 'How many sides: ' )

length = circumference / sides
angle = 360 / sides

for i in range ( sides ) :
    turtle.forward ( length )
    turtle.right ( angle )

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

Experiment indicates that on a 1280×1024, 90dpi screen anything over about 20 sides cannot easily be distinguished from an actual circle.

Exercise 2.6 Rewrite your answer to **??** using a function.

```
import turtle

length = 10

def drawPair ( ) :
    global length
    turtle.forward ( length )
    length += 5
    turtle.left ( 120 )
    turtle.forward ( length )
    length += 5
    turtle.left ( 120 )

for i in range ( 10 ) :
    drawPair ( )

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

Exercise 2.7 Write a program to draw the following image:



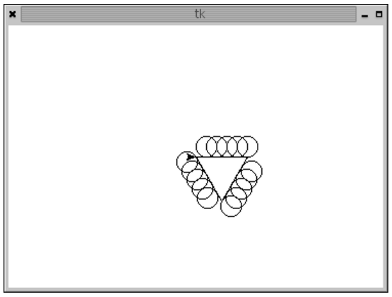
```
import turtle

def drawSquare ( x ) :
    for i in range ( 4 ) :
        turtle.forward ( x )
        turtle.left ( 90 )

def setPosition ( x , y ) :
    turtle.up ( )
    turtle.goto ( x , y )
    turtle.down ( )

def drawSequence ( ) :
    for i in range ( 10 ) :
        turtle.circle ( i * 4 )
        turtle.up ( )
        turtle.forward ( i )
        turtle.down ( )
        drawSquare ( i * 4 )
        turtle.up ( )
        turtle.forward ( i )
        turtle.down ( )
```

```
setPosition ( -120 , -70 )
drawSequence ( )
setPosition ( 0 , 0 )
drawSequence ( )

raw_input ( 'Press any key to terminate: ' )
```

Exercise 2.8 When the code:

```
import turtle

sides = 3
length = 5

for i in range ( sides ) :
    for j in range ( length ) :
        turtle.forward ( 10 )
        turtle.circle ( 10 )
    turtle.right ( 120 )

raw_input ( 'Press any key to end' )
```

is executed the result is:



i.e. it draws a triangle with circles along its edges. The *intention* was that modifying the variable sides would allow you to draw a square (by setting it to 4), a pentagon (setting it to 5) and so on. This doesn't currently work, however. The programmer clearly became distracted and left the program half finished. Your job is to make the code work as intended.

Hint: Remember that for any regular polygon, the external angles always add up to 360. If they didn't, it would either not be closed or it would overlap, or it would not be regular!

Challenges

Challenge 2.1 Go back and make sure you finish the programming exercise on creating polygons surrounded by circles. If necessary modify the program so that the drawing is undertaken by a function with parameters for side length and number of sides. Then add a further parameter to your function that allows the size of the circles to be adjusted. Finally, write a program that asks the user to enter values for side length, number of sides and size of circles – this is now a general purpose regular-polygon-surrounded-by-circles generator. Not entirely useful per se, but fun!

Challenge 2.2 Can you spot a way of making money given the exchange rates in the last version of the currency conversion program? The trick is to write a program to show how much you could lose or make by exchanging between these currencies.

What stops people (as opposed to banks) exploiting this for profit?

(If you are a bank, the opportunities here are called *currency arbitrage*, but it is a high-risk activity.)

Index