Lesson 8

Introduction to User-Defined Functions

We've used lots of different functions up to now. ***print()*** is a function which prints text to the output console, ***append()*** is a function which adds an item to a list, and so on. Someone else has defined these functions for us, and they allow us to easily perform tasks in Python.

We can also write our own functions. These allow us to bundle up code that does a specific job (especially if that job might be repeated a lot of times) and put it outside of our "main code". Often it reduces the amount of code we have to write, but more importantly it should always make our code neater and easier to follow.

Defining a Function

First of all we need to write the code that will be executed when the function is called. Below is a function that asks the user their favourite colour and prints out the result.

```
def FavouriteColour():
    colour = input("What is your favourite colour? ") # ask user favourite colour
    print("My favourite colour is " + colour)
```

- def - this tells the computer you want to **define** a new function.
- FavouriteColour(): - this is the name of the function you are defining, followed by a **colon** (“**:**”) with the code inside the function **indented**.

The indented code is all of the code that will be executed when the ***FavouriteColour()*** function is called in our main code.

Calling a Function

After we have defined the function in our code, we need to call the function in the **main** part of our code. We need to define a main function which will be used to call our user-defined functions and will have all of the rest of our code.

```
def main(): # define main function
    FavouriteColour() # call the function
```

At the moment, we are just calling ***FavouriteColour()*** inside our ***main()*** function but in the future we will have more than just one function call inside ***main()***.

The final step is to call the ***main()*** function so the code will be executed.

```
if __name__ == '__main__':
    main() # call main() function
```

The if statement will execute the ***main()*** function if the python interpreter is running that module (the source file) as the main program (you don't need to worry about this!).

So if we put all of our code together, it will look like this:

```
def FavouriteColour():
    colour = input("What is your favourite colour? ") # ask user favourite colour
    print("My favourite colour is " + colour)

def main(): # define main function
    FavouriteColour() # call the function

if __name__ == '__main__':
    main() # call main() function
```

Tasks:

1. Copy and run the code above.
2. Define a function that asks the user to input **two numbers**.
3. Add the numbers together and print the result.

4. If the number is odd, print “odd number. Or else, print “even number”.

   Note: to check whether a number is odd or even we need to use the modulo operator (%).

```
if number%2 == 0: # 2 divides evenly into number
  # even number
else: # 2 does not divide evenly into number
  # odd number
```

5. Call the function in main().

Snowstorm Project

We can also draw more advanced shapes using Tina the Turtle, along with **loops** and **functions**. The goal of this project is to draw snowflakes on the screen, that we can later customize.

Tasks:

1. Import the Turtle module, initialize Tina the Turtle, **set the pen size to 6** and **the pen colour to white**.

2. Make a coloured background by getting the turtle window and setting a colour for it.

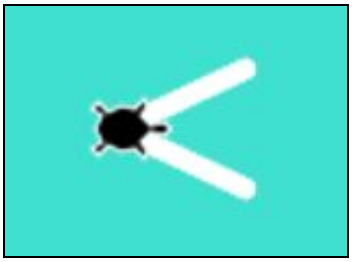
```
screen = turtle.Screen()

screen.bgcolor("turquoise")
```

3. Define a function called **vshape**, to draw the first bit of the snowflake in the shape of a 'v'.

```
def vshape():
```

4. Place the instructions to draw the 'v' shape into the function.

```
tina.right(25)      # turn right 25 degrees
tina.forward(50)    # go forward 50 pixels
tina.backward(50)   # go backwards 50 pixels
tina.left(50)       # turn left 50 degrees
tina.forward(50)    # go forward 50 pixels
tina.backward(50)   # go backwards 50 pixels
tina.right(25)      # turn right 25 degrees
```



5. Declare and call your ***main()*** function which will be used to call the ***vshape()*** function.

```
def main(): #define main function
    vshape() # call function

if __name__ == '__main__':
    main() # call main function
```

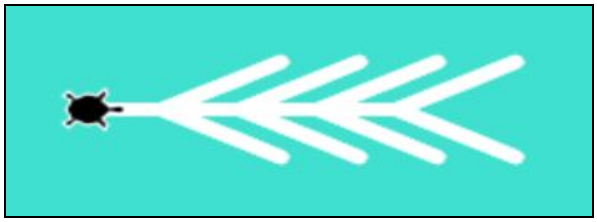
Run your code to see what happens!

6. Define a new function called ***snowflakeArm()***, to turn the ‘v’ shape into one ‘arm’ of the snowflake.

```
def snowflakeArm():
```

7. Place the instructions to draw the snowflake arm by drawing a ‘v’ multiple times in different locations.

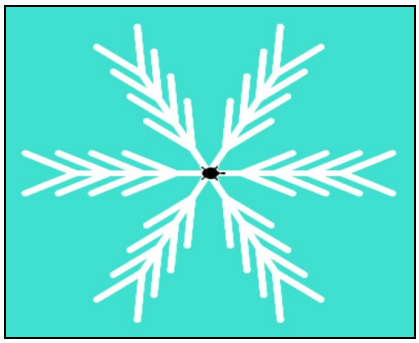
```
for i in range(0,4): # for loops runs 4 times
    tina.forward(30) # go forward 30 pixels
    vshape() # draw 'v' shape

tina.backward(120) # go back 120 pixels (to centre)
```



Call ***snowflakeArm()*** in your main function (take out ***vshape()***) and run your code!

8. Define another function to draw a whole snowflake, using a for loop to repeat the snowflake arm that you just made.

```
def snowflake():
    for i in range(0,6): # for loops runs 6 times
        snowflakeArm() # draw snowflake arm
        tina.right(60) # turn right 60 degrees
```



Call ***snowflake()*** in your main function (take out ***snowflakeArm()***) and run your code!

9. Run your code!

We can use the random module to make a multi-coloured snowflake by choosing a random colour from a list of different colours.

Tasks:

1. Import the random module at the top of the program.

```
import random
```

2. Delete the line that sets the pen colour (so we can use different colours).

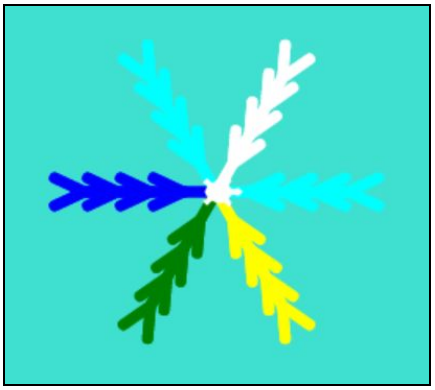
```
tina.pencolor("white") # DELETE THIS LINE
```

3. Set up a list of colours that our snowflake arms can be and insert it after the line that sets the background.

```
colours = ["blue","purple","cyan","white","yellow","green", "red","pink","orange"]
```

4. Pick a random colour from the list for each snowflake arm. This involves adding an extra line in the for loop in the ***snowflake()*** function.

```
def snowflake():
    for i in range(0,6):
        tina.color(random.choice(colours))
        snowflakeArm()
        tina.right(60)
```



Note: If you want each snowflake to be a single colour, move this line above the for loop.

5. Run your code!

Now we are going to use the random module to fill the screen with coloured snowflakes of different sizes and in different positions on the screen.

Tasks:

1. To vary the size of the snowflakes, we need to add a 'size' variable to the ***snowflake()*** function.

```
def snowflake(size):
    for i in range(0,6):
        tina.color(random.choice(colours))
        snowflakeArm(size)
        tina.right(60)
```

2. We also need to add this new 'size' variable to the ***vshape()*** and ***snowflakeArm()*** functions, to replace the numbers in **forward** and **backward** commands.

```
def vshape(size):
    tina.right(25)
    tina.forward(size)
    tina.backward(size)
    tina.left(50)
    tina.forward(size)
    tina.backward(size)
    tina.right(25)
```

```
def snowflakeArm(size):
    for i in range(0,4):
        tina.forward(size)
        vshape(size)
    tina.backward(size*4)
```

3. In your ***main()*** function, use a for loop to draw several snowflakes, each with its own size and position chosen at random.

```
for i in range(0,10): # for loop runs 10 times
    size = random.randint(5,30) # picks a random size between 5 and 30
    tina.pensize(random.randint(1,7)) # picks a random pen size between 1 and 7
    x = random.randint(-150,150) # picks a random x-coordinate between -150 and 150
    y = random.randint(-150,150) # picks a random y-coordinate between -150 and 150
    tina.penup() # don't draw when turtle moves
    tina.goto(x,y) # go to random position
    tina.pendown() # start drawing again
    snowflake(size) # draw snowflake
```

4. Save and run your code!

Extra Tasks:

1. Make **every second snowflake** you draw a 10-arm snowflake. At the moment you are drawing a 6-arm snowflake by having the for loop run 6 times and turning by 60°, so you will need to change the for loop to run 10 times and ensure the turtle turns right 36° when you want to draw the 10-arm snowflake.

   Note: This will require you to test whether the current value of the for loop used to draw the 10 snowflakes is on an odd or even index. If its odd, draw the 6-arm snowflake, or else if its even, draw the 10-arm snowflake.

   Note: You will also have to pass in something to your ***snowflake()*** function to know when to change the values in the for loop and the turn angle. One way of accomplishing this may be to use a **boolean** (a variable that can be **True** or **False**). Declare a boolean at the top of your code.

   ```
   odd = False # Boolean variable set to False at the start
   ```

   Then, if the index of the for loop is even(explained above), set “odd” to “False”. Or else if the index is odd, set “odd” to “True”.

   ```
   for i in range(0,10):
       if i%2 == 0: # no remainder, so index is even
           odd = False # set boolean to False
       else: # one remainder, so index is odd
           odd = True # set boolean to True

       # Rest of code to draw snowflakes
   ```

You can then pass your “odd” variable into your ***snowflake()*** function and you it to conditionally draw your snowflakes.

```
def snowflake(size, odd):
    if odd == True: # if "odd" boolean is True
        for i in range(0,6): # draw 6-arm snowflake
            tina.color(random.choice(colours))
            snowflakeArm(size)
            tina.right(60)
    else: # else "odd" boolean is False
        for i in range(0,10): # draw 10-arm snowflake
            tina.color(random.choice(colours))
            snowflakeArm(size)
            tina.right(36)
```

```
snowflake(size, odd) # draw snowflake
```