

Drawing a snowman

with Python Turtle





1. Setup

If you don't have a Python editor already installed on your computer, we will need to install one for this guide. 





I'd recommend using the 'Mu' Python editor as it is extremely beginner friendly. Head to the 'Mu' website here:

<https://codewith.mu> and follow the on-site instructions to download.

Once you have 'Mu' (or your editor of choice) installed, open it up. If you are asked which mode you would like to enter, click 'Python 3'.

 Save the file at this point somewhere memorable so that if anything happens you have a place to come back to. 

A few things to remember:

- It's important that you build the snowman in the order that the guide says, as each section builds on from the next 
- Upper and lowercase letters matter in Python, so be sure to copy the code from the yellow boxes carefully. 
- Green boxes are key programming concepts 
- Optional extras are included in orange 
- Remember, the most important thing is to have fun!

2. Let's Begin! ✨

Python Turtle allows us to draw things to the screen with the help of a 'turtle' which holds our pen and listens to our instructions on where to go and what to draw!

To start we need to add some code at the start of our project to tell the computer some important things such as:

- how to access the turtle drawing library
- what our pen is called
- where to draw our picture
- what color we would like the background

 Try it now!

Let's start by adding the turtle library to our project ↓

```
import turtle
```

Now let's tell our computer that we would like to see our design

on the screen while its drawing:

```
window = turtle.Screen()
```

Next, let's tell our computer what the pen is called. I've called mine 'pen', but if you'd like yours to be called 'bob' then that's fine too!

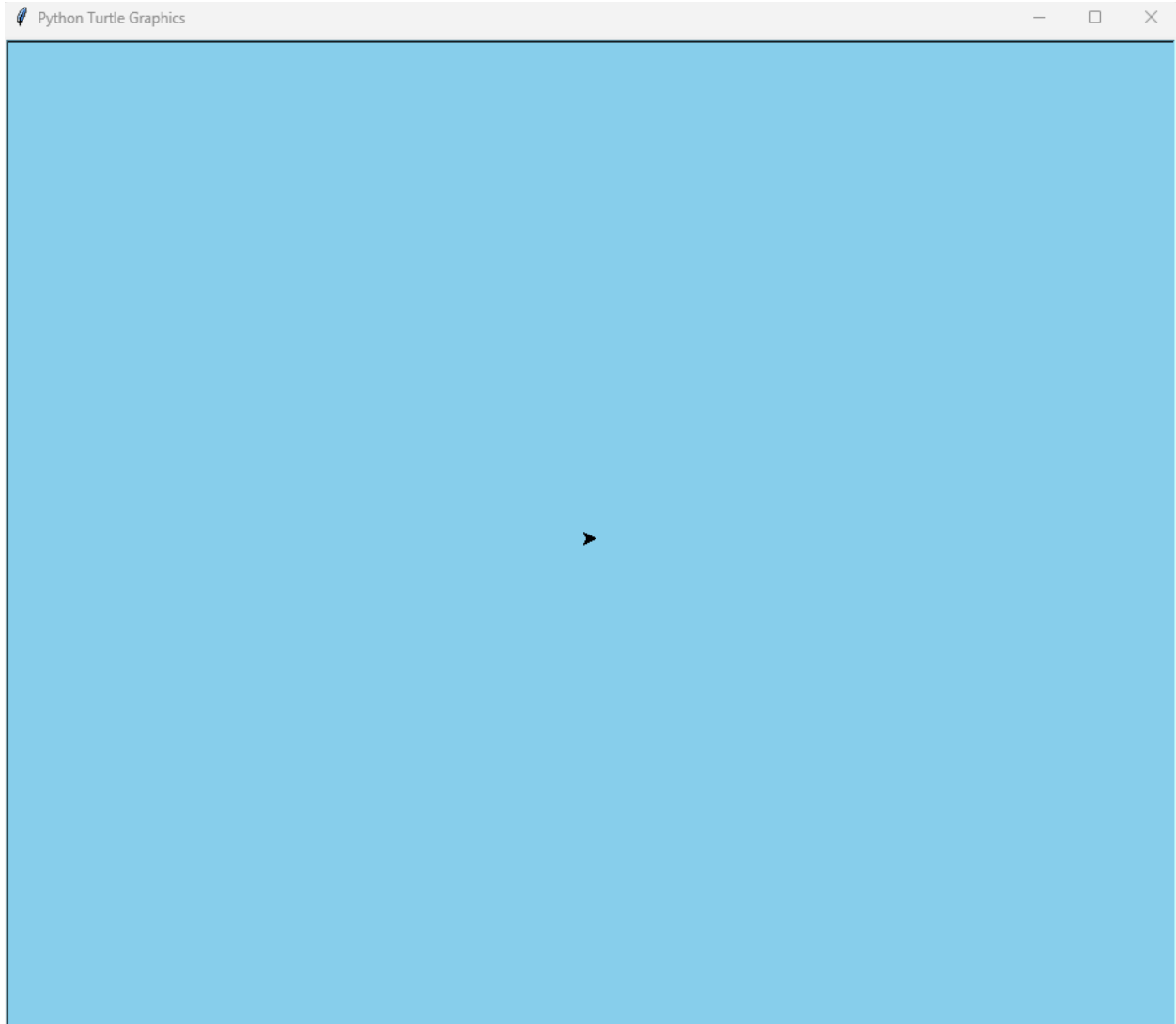
```
pen = turtle.Turtle()
```

Finally, let's change the background color. It's up to you what color you would like the background to be and you can change the color by changing the name within the speech marks. I chose 'skyblue'.

```
window.bgcolor('skyblue')
```

 Congratulations! 

You can now press the 'run' button in the top bar of your editor and you should see a blue screen. Our canvas is now ready to draw on!



- OPTIONAL (but highly recommended) ● Its useful to divide sections of the project with notes as you go along. This makes it easier to refer back to them later to fix mistakes or change something. Notes are ignored by the computer so they won't affect our project.
- To add a note, simply add a # at the start of the note such as:
#this is the setup section OR #draw body

3. Drawing the base of the snowman

We are ready to create the first circle that will be the base of our snowman! In this step, we will go through each line in detail to explain what the code is doing.

In future steps we won't need to do this as we will re-use this code.


Try it now!

First let's tell our turtle what color the pen should be 

```
pen.color('white')
```

Next our turtle needs to know what color to fill in our circle 

```
pen.fillcolor('white')
```

Next let's tell our turtle where to begin filling. This is important because we only want to color-in certain areas 

```
pen.begin_fill()
```

This next line tells the turtle to lift up the pen before we are

ready. Just think how messy it would get if the pen stayed down before we had arrived where we wanted to draw ↓

```
pen.penup()
```

Here is where we tell the turtle which position to go to with an x and y position on the screen ↓

```
pen.setposition(0,-395)
```

Then we tell the turtle to put the pen back down on the page as we are ready to draw ↓

```
pen.pendown()
```

Here is where we tell our turtle to draw the circle, the number that we input here is the radius of the circle ↓

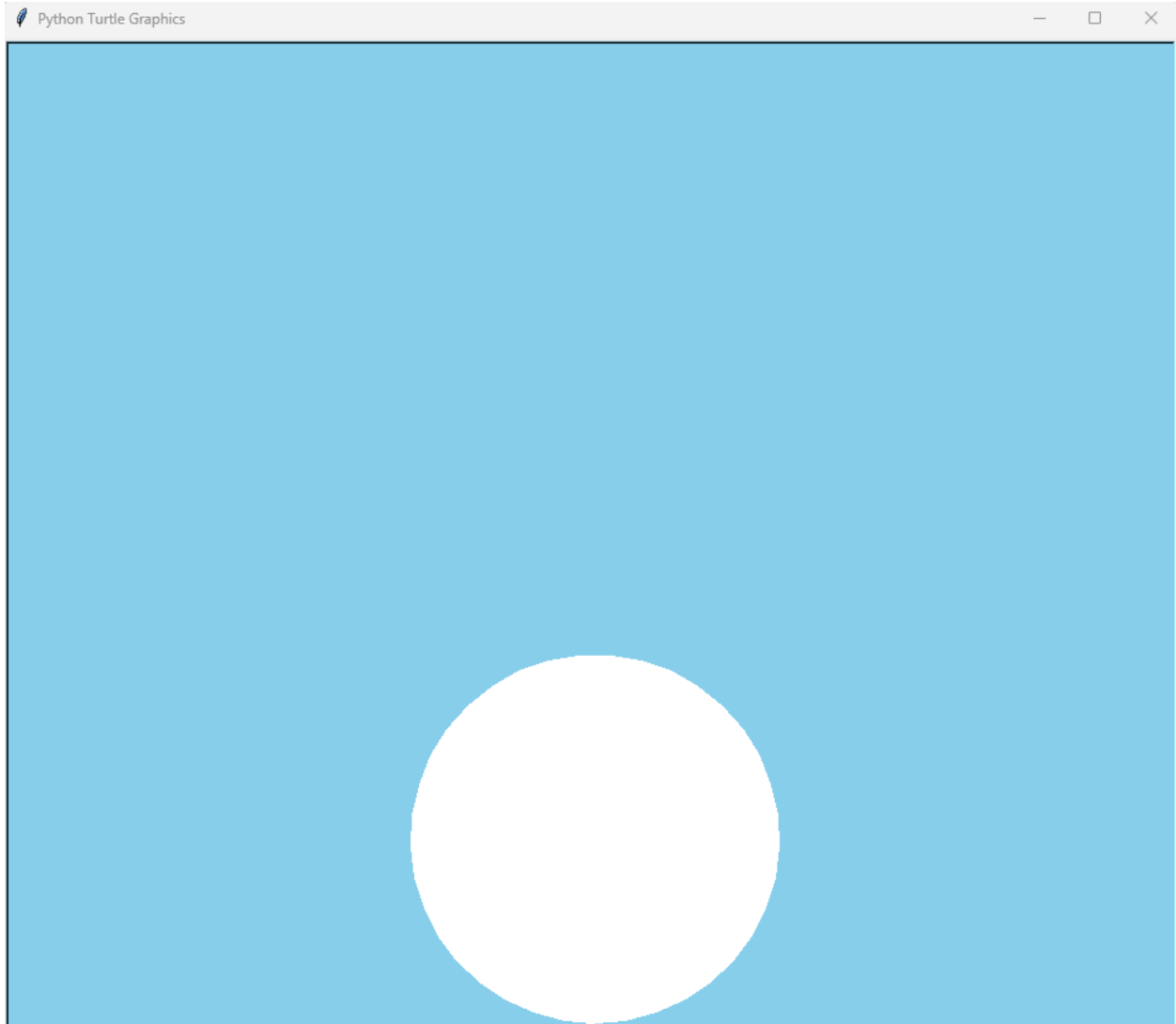
```
pen.circle(150)
```

Last but not least, we tell the turtle to stop filling in the shape ↓

```
pen.end_fill()
```

🌈 Congratulations! 🌈

Press the 'run' button and your turtle should draw the base of the snowman!



4. The Power of functions 💡

This design will have a lot of circles which could mean repeating ourselves over and over again...boring!

This is why programmers use functions! Understanding

these will make our snowman come to life much quicker!

A function is like a box where we can store code so that we can re-use it again in the future if we want to, without having to write it all out again.

In Python the word 'def' marks the start of the function.

This is followed by the function's name, in this case 'favourite_foods'.

After this we always end the first line of our function with '():'

Anything that we put after this first line is moved over to the right by using the 'tab' key on your keyboard. This tells the computer that this code is part of the instructions that we are putting inside the function box.

For example, if we wanted our code to list our favorite foods, we could write it like this ↓

```
def: favorite_foods():  
    print('pizza')  
    print('biscuits')  
    print('chips')
```

This code won't do anything at the moment, as we need to 'call' it. 'calling' a function is how we use the set of instructions that we have put inside it.

We can do this, simply by writing the name of the function, followed by brackets ↓

```
favorite_foods()
```

Our code would then write 'pizza biscuits chips' on the screen

because that is what the instructions inside `favorite_foods()` tell the computer to do.

Sometimes though, our favorite foods change and we need a way to make our set of instructions easily re-usable.

There is one final part to our function called 'arguments'. Arguments allow us to give information to the function so that it can use this information when the time comes.

Let's take our `favorite_foods()` function for example.

Our favorite foods change sometimes, so we want to give our function temporary names aka 'arguments' until we know what foods we actually want.

This would look like this ↓

```
def favorite_foods(food1, food2, food3):  
    print(food1)  
    print(food2)  
    print(food3)
```

We are telling our function that when we use it, we will tell it at that moment in time what our 3 favorite foods are with 3 arguments.

If we used this function, we would call it like this ↓

```
favorite_foods('pizza', 'biscuits', 'chips')
```

Our code would then write out the same line as before, 'pizza biscuits chips'.

If we wanted to change our minds however, we could change of the foods when we call the function like this ↓

```
favorite_foods('pizza', 'chocolate', 'chips')
```

This would write 'pizza chocolate chips' because we changed

the food2 argument to 'chocolate'.

Let's take what we have learnt about functions and apply it to our snowman!


Try it now!

At the moment, you should have the following code:

```
pen.color('white')
pen.fillcolor('white')
pen.begin_fill()
pen.penup()
pen.setposition(0,-395)
pen.pendown()
pen.circle(150)
pen.end_fill()
```


Now let's put our code inside of a function like this 

```
def draw_circle():
    pen.color('white')
    pen.fillcolor('white')
    pen.begin_fill()
    pen.penup()
    pen.setposition(0,-395)
    pen.pendown()
    pen.circle(150)
    pen.end_fill()
```

Because we want to re-use our function for other circles as well, we need to replace some of the information with arguments to make it reusable 

```
def draw_circle(x, y, radius, pen_color, fill_color):
    pen.color(pen_color)
    pen.fillcolor(fill_color)
    pen.begin_fill()
    pen.penup()
    pen.setposition(x,y)
    pen.pendown()
    pen.circle(radius)
    pen.end_fill()
```

In our code we have now replaced the specific values with arguments, which allows us to specify these when we call the function as you'll see below.

Now that we have the function finished, nothing will be printed to the screen unless we call it with the correct arguments passed in. Below your function, call it with the specific values that we had before 

(I have added an optional comment with the # symbol)

```
#draw snowmans base circle
draw_circle(0, -395, 150, 'white', 'white')
```

 Congratulations! 

Press the 'run' button and our function should follow the set of instructions that we have put inside it and draw the base of our snowman like it was before!

3. Drawing the body and head of the snowman


If you have gotten this far, well done!

Your hard work is paying off now that we are using functions to build our snowman. As you'll see, this next section will be much simpler because of it.

The body and head of our snowman are made of two more circles which is great, because we have a function that can draw circles!

We just need to pass our functions the correct arguments that correspond to the x and y position, the radius of the circle etc.

Try it now!

Below the last function that we called to draw the snowman's base, call another one with new arguments passed in like this 

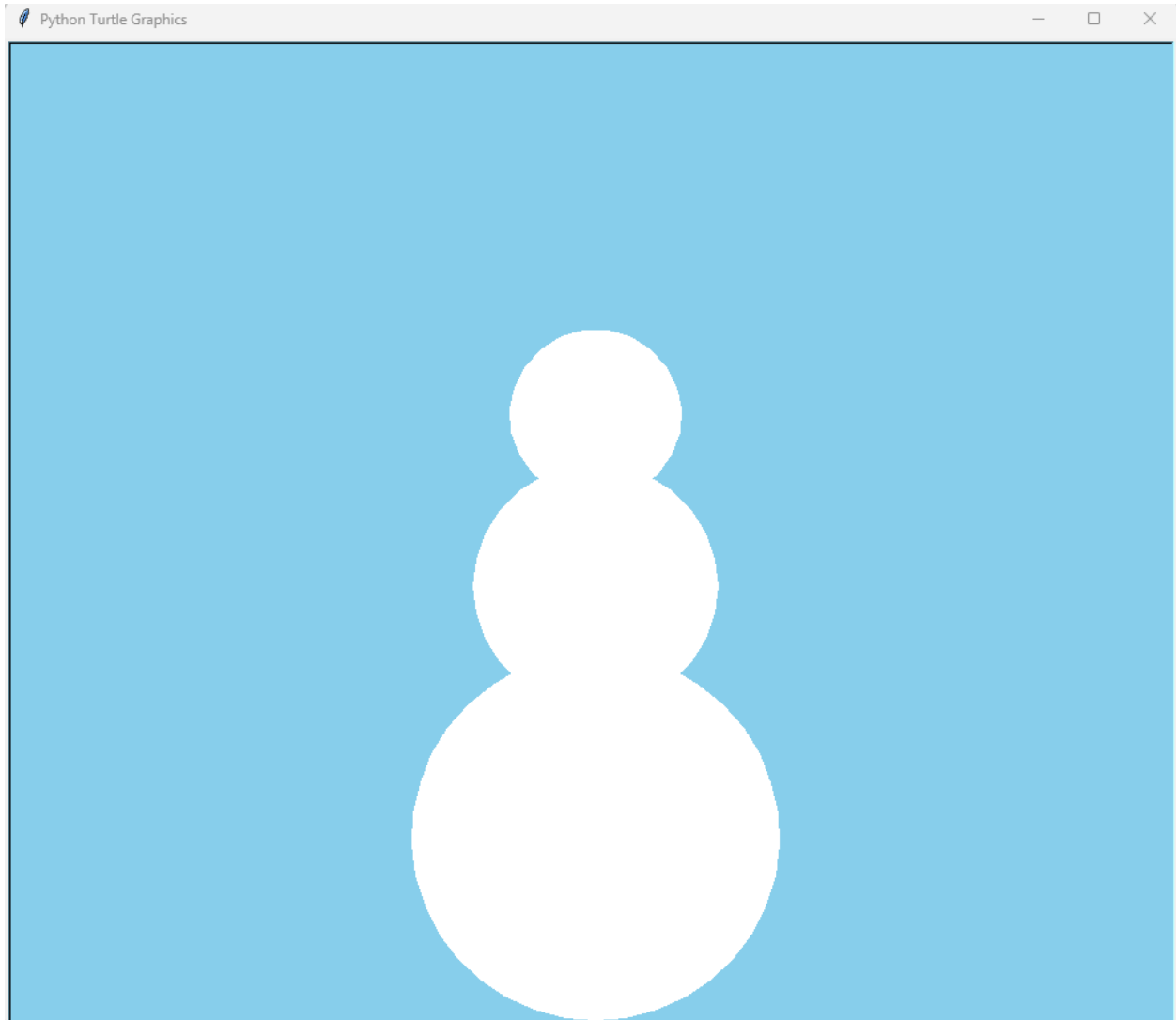
```
#draw body circle  
draw_circle(0, -140, 100, 'white', 'white')
```

Now let's add another underneath like before, which will draw a circle for the head 

```
#draw body circle  
draw_circle(0, 30, 70, 'white', 'white')
```

🌈 Congratulations! 🌈

Press the 'run' button and our function will do the hard work for us and draw our snowman's body and head!



- Optional ●

As we add more and more sections to our drawing, we may not want to wait for our snowman to draw each time we press run.

If you won't want to wait, you can add this line of code at the top of your project (just below `pen.bgcolor('skyblue')`) which will make it much faster ↓

```
pen.speen(0)
```

- You can also make the turtle visible with:

```
pen.shape('turtle')
```

- And hide the cursor all together with:

```
pen.hideturtle()
```

6. Adding a face and buttons

Let's call our function again a few more times to create the eyes, mouth and buttons of our snowman!

Try it now!

Below the last function that we called, call another one to draw the left eye ↓

```
#draw left eye  
  
draw_circle(-30, 110, 8, 'black', 'black')
```

And another for the right eye ↓

```
#draw left eye  
  
draw_circle(30, 110, 8, 'black', 'black')
```

Now lets call some more functions to draw circles for the mouth



```
#draw mouth circles  
  
draw_circle(-25, 75, 2, 'black', 'black')  
draw_circle(-15, 70, 2, 'black', 'black')  
draw_circle(-5, 68, 2, 'black', 'black')  
draw_circle(5, 68, 2, 'black', 'black')  
draw_circle(15, 70, 2, 'black', 'black')  
draw_circle(25, 75, 2, 'black', 'black')
```

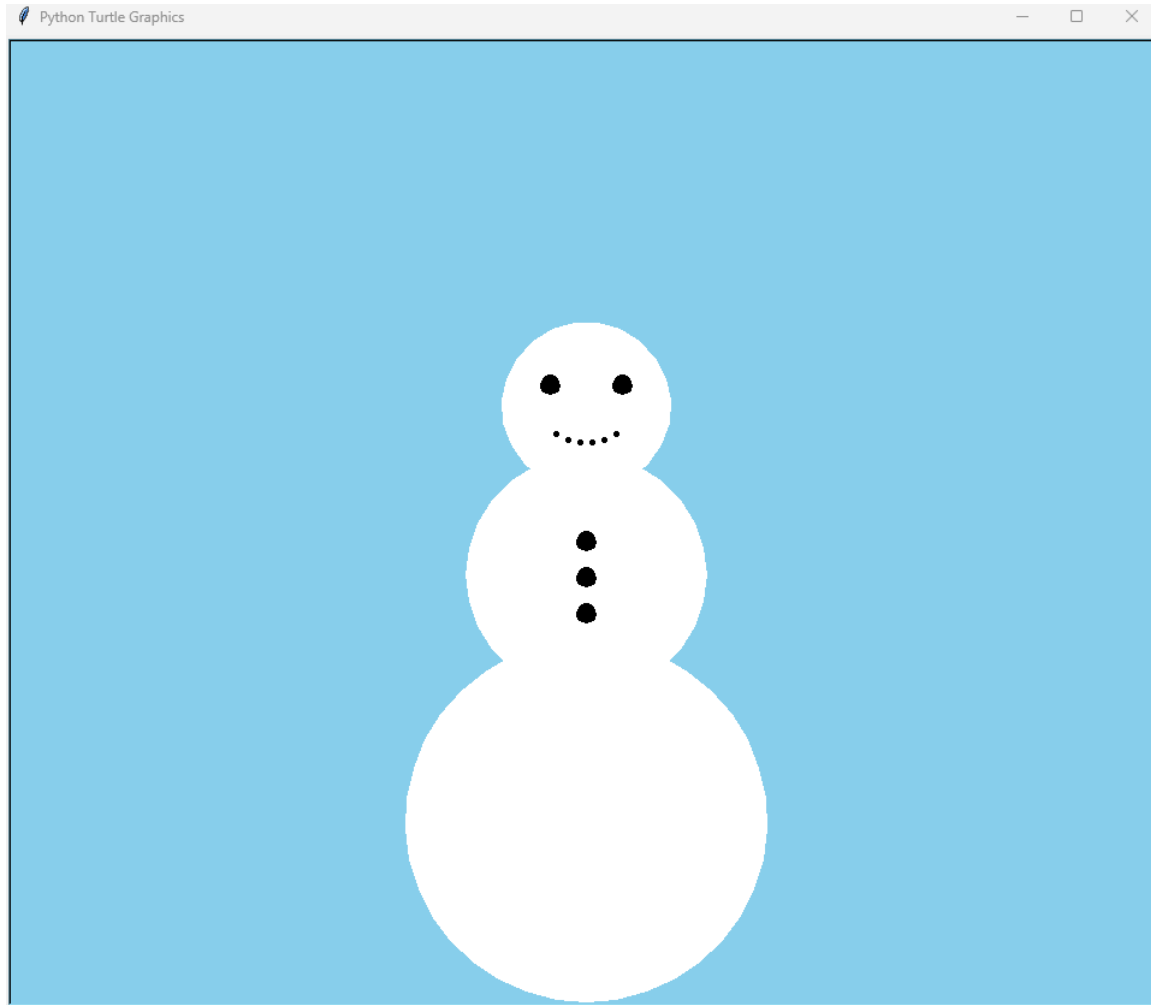
Let's use our circle function for the last time to add some buttons



```
#draw buttons  
  
draw_circle(0, -20, 8, 'black', 'black')  
draw_circle(0, -50, 8, 'black', 'black')  
draw_circle(0, -80, 8, 'black', 'black')
```

 Congratulations! 

Our snowman is taking shape! Press the 'run' button and watch our face and buttons appear!



7. Drawing the nose 🥕

Now let's add a carrot for our snowman's nose!

For this section we will guide our turtle manually as we cannot use our `draw_circle` function.

We do this with extra commands such as:

`pen.forward()` - this tells our turtle to move forwards in whichever direction it is facing. The number tells the turtle how far to move.

`pen.left()` and `pen.right()` - this command tells our turtle to turn left or right and the number is the amount of degrees to turn.

These commands are useful for any other projects that you'll work on in the future 🌻

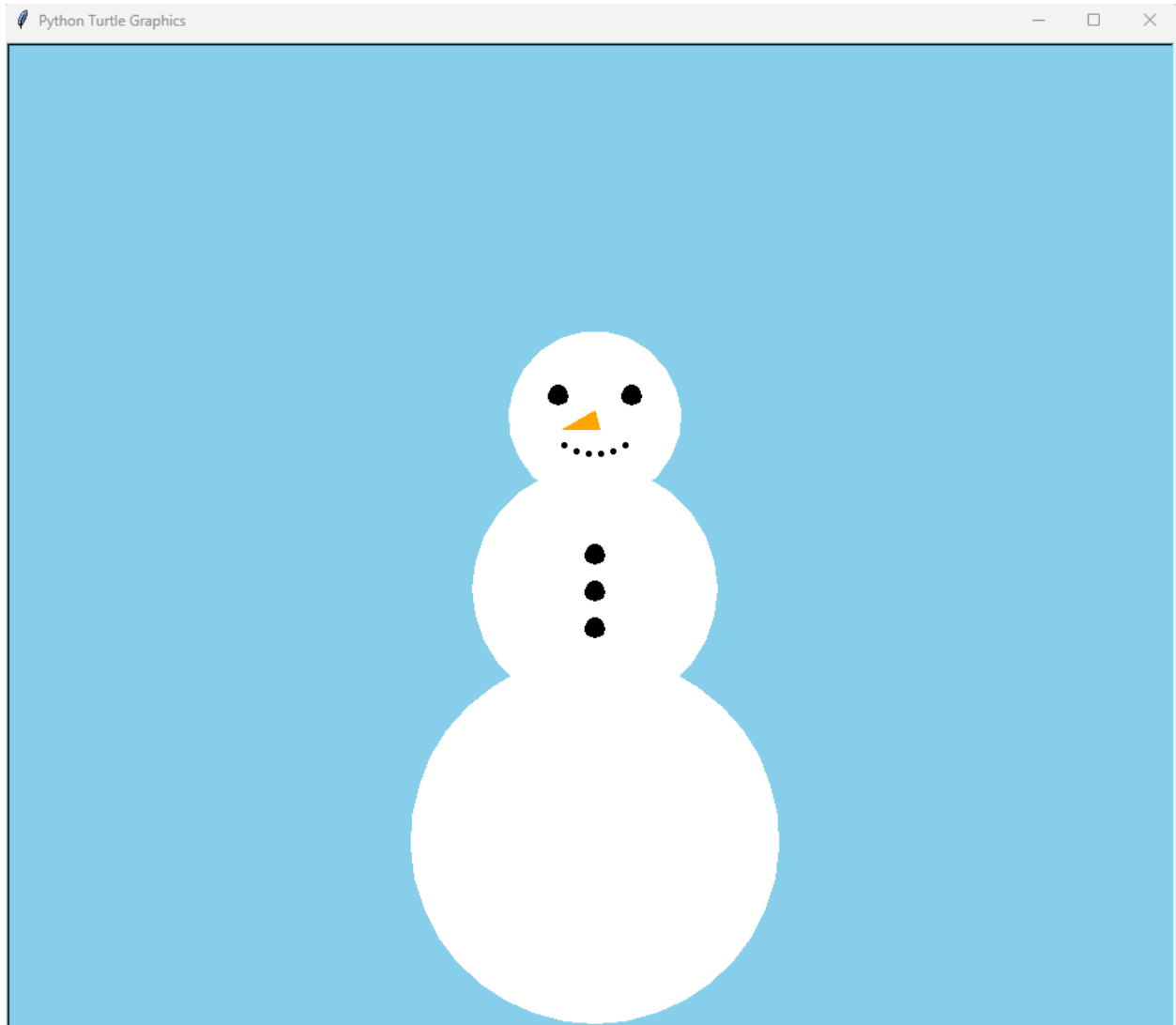
Try it now!

Below the functions that we called to draw the face and buttons, let's add this code for the nose 

```
#draw nose  
  
pen.color('orange')  
pen.fillcolor('orange')  
pen.penup()  
pen.setposition(0, 105)  
pen.begin_fill()  
pen.pendown()  
pen.right(150)  
pen.forward(30)  
pen.left(150)  
pen.forward(30)  
pen.goto(0, 105)  
pen.end_fill()
```

🌈 Congratulations! 🌈

Press the run button and you should see our nose drawn in the correct place!



8. Drawing the hat 🎩

Time to put a hat on our snowman!

We can't use our `draw_circle()` function, so we will guide our turtle manually again.

We also use a new command here ↓

`pen.pensize(5)` - This changes the thickness of our pen which is useful to remember for other projects that you work on.


 **Try it now!**

Below the code that we wrote to draw the snowman's nose, let's add some more code to draw our snowman's hat ↓

```
#draw hat

pen.color('black')
pen.fillcolor('black')
pen.begin_fill()
pen.penup()
pen.pensize(5)
pen.setposition(0, 150)
pen.pendown()
pen.forward(75)
pen.left(90)
pen.forward(10)
pen.left(90)
pen.forward(40)
pen.right(90)
pen.forward(60)
pen.left(90)
pen.forward(70)
```

```
pen.left(90)
pen.forward(60)
pen.right(90)
pen.forward(40)
pen.left(90)
pen.forward(10)
pen.left(90)
pen.goto(0, 150)
pen.end_fill()
```

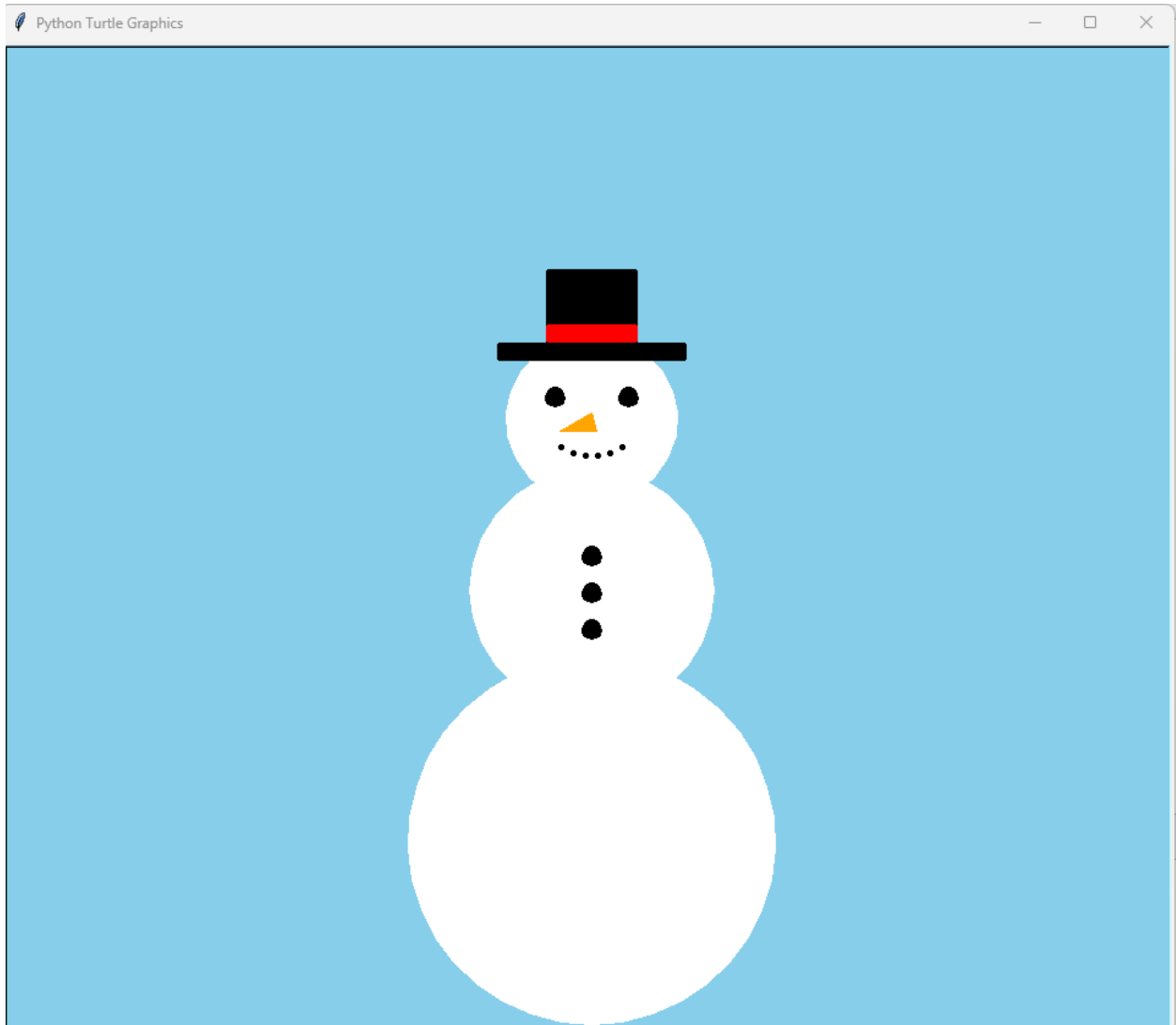
If you run the code now, you will see that we have given our snowman a top hat, but we need to finish it off with a piece of ribbon. Let's add this 

```
#draw ribbon

pen.color('red')
pen.fillcolor('red')
pen.begin_fill()
pen.penup()
pen.setposition(1, 165)
pen.pendown()
pen.forward(34)
pen.left(90)
pen.forward(10)
pen.left(90)
pen.forward(70)
pen.left(90)
pen.forward(10)
pen.setposition(1, 165)
pen.end_fill()
```

 Congratulations! 

Run your code and take a look at our new top hat, complete with ribbon!



9. Adding Arms 🙌


It's time to add the last part of our snowman, the arms!

We need to draw two arms, but we don't want all of our code out twice!

This would be a good place to create another function to draw each arm.


Let's take a look!

 Try it now!

Go to the top of your file just below the `draw_circle()` function that we added earlier. Lets add another function like this 

```
#draw buttons

def draw_arm(x, y, pen_color, fill_color):
    pen.color(pen_color)
    pen.fillcolor(fill_color)
    pen.pensize(7)
    pen.penup()
    pen.setposition(x, y)
    pen.pendown()
    pen.left(115)
    pen.forward(100)
    pen.left(45)
    pen.forward(30)
    pen.right(180)
    pen.forward(30)
    pen.left(90)
    pen.forward(30)
    pen.left(180)
    pen.forward(30)
    pen.right(130)
    pen.forward(30)
```

Now go to the bottom of the file and just below the code that we added to draw the snowman's hat, lets call our function to draw the right arm 

```
#draw right arm
draw_arm(70, -40, 'chocolate', 'chocolate')
```

Let's call our function again for the left arm ↓

```
#draw left arm  
draw_arm(-70, -40, 'chocolate', 'chocolate')
```

💡 D.R.Y 💡

In professional programming there is a principle called D.R.Y which stands for:

Don't
Repeat
Yourself

This is a best practice that developers use to keep code clean and avoid adding unnecessary code. The coolest part is that you have been using the D.R.Y principle here today, to prevent having to repeat many lines of code to draw circles or arms. 🎉

🌈 Well done! 🌈

You finished your design completely using Python code!

In the process you learnt about writing code in Python, the Python turtle library, functions and D.R.Y programming.

Remember that this is *your* design and you can change it however you'd like.

How about a different coloured hat, or ribbon, or buttons?

Why not make your drawing into a Christmas card for family and friends?

or maybe

Take what you have learnt and draw your own patterns and pictures!

 Happy Coding! 