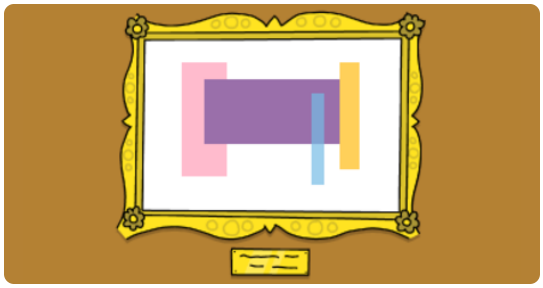
Projects

Modern Art

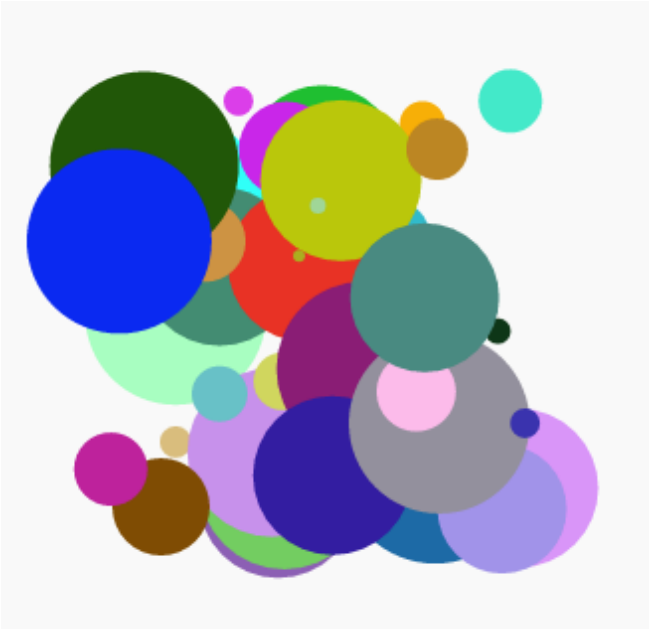
Code your own computer-generated modern art.

Python



Step 1 Introduction

In this project you will create computer generated modern art. You will use functions to write code that you can use over and over again.



Additional information for club leaders

If you need to print this project, please use the **Printer friendly version** (https://projects.raspberrypi.org/en/projects/modern-art/print).

Club leader notes

Introduction:

This project introduces functions through a colourful modern art generator. Functions are used to package useful turtle graphics code which can then easily be used to create funky art.

Online Resources

This project uses Python 3. We recommend using **trinket** (https://trinket.io/) to write Python online. This project contains the following Trinkets:

- **'Modern Art' starting point – jumpto.cc/modern-go** (http://jumpto.cc/modern-go)

There is also a trinket containing a sample solution to the challenges:

- **'Modern Art' Finished – rpf.io/modern-finished** (https://rpf.io/modern-finished)

Offline Resources

This project can be **completed offline** (https://www.codeclubprojects.org/en-GB/resources/python-working-offline/) if preferred. You can access the project resources by clicking the 'Project Materials' link for this project. This link contains a 'Project Resources' section, which includes resources that children will need to complete this project offline. Make sure that each child has access to a copy of these resources. This section includes the following files:

- modern-art/modern-art.py
- modern-art/snippets.py

You can also find a completed version of this project's challenges in the 'Volunteer Resources' section, which contains:

- modern-art-finished/modern-art.py

(All of the resources above are also downloadable as project and volunteer `.zip` files.)

Learning Objectives

- Functions;

This project covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (http://rpf.io/curriculum):

- **Combine programming constructs to solve a problem.** (https://www.raspberrypi.org/curriculum/programming/builder)

Challenges

- Turtle art - define a new function to complete the turtle art generator.
- More modern art - create a new function that calls other functions to generate modern art.

Frequently Asked Questions

- To avoid having to wait for earlier code to run when adding to the project children can comment out code using a '#' at the beginning of a line.
- If they do want all of their code to run then `clear()` can be used to clear the screen.

Project materials

Project resources

- **.zip file containing all project resources** (https://projects-static.raspberrypi.org/projects/modern-art/0806a645c4f4613cf8ac9df4ff6c2b72a1ba5b3c/en/resources/modern-art-project-resources.zip)
- **Online Trinket containing 'Modern Art' starter resources** (http://jumpto.cc/modern-go)
- **modern-art/modern-art.py** (https://projects-static.raspberrypi.org/projects/modern-art/0806a645c4f4613cf8ac9df4ff6c2b72a1ba5b3c/en/resources/modern-art-modern-art.py)
- **modern-art/snippets.py** (https://projects-static.raspberrypi.org/projects/modern-art/0806a645c4f4613cf8ac9df4ff6c2b72a1ba5b3c/en/resources/modern-art-snippets.py)

Club leader resources

- **.zip file containing all completed project resources** (https://projects-static.raspberrypi.org/projects/modern-art/0806a645c4f4613cf8ac9df4ff6c2b72a1ba5b3c/en/resources/modern-art-volunteer-resources.zip)
- **Online completed Trinket project** (https://trinket.io/python/47bbc2fc2b)
- **modern-art-finished/modern-art.py** (https://projects-static.raspberrypi.org/projects/modern-art/0806a645c4f4613cf8ac9df4ff6c2b72a1ba5b3c/en/resources/modern-art-finished-modern-art.py)

Step 2 Random colours

- Open this trinket: **jumpto.cc/modern-go** (http://jumpto.cc/modern-go).
- You can set the colour of a turtle by saying how much red, green and blue you would like from 0 to 255.

  Add the following code to get a purple turtle:

  Purple is made by mixing together red and blue.

Error - bad color sequence: (150, 0, 150)

Do you get the error `bad color sequence: (150, 0, 150)` when running your code.

This is because trinket uses a different colour mode to other Python editors. It can be fixed by changing the `colormode` to `255`.

```
from turtle import *

colormode(255)

shape("turtle")
color(150,0,150)
```

- Try some different numbers to get different colours.

  Remember each number can be from 0 to 255.

- How about choosing a random colour?

  Update your code to choose a random number between 0 and 255 for the red, green and blue values:

- Click 'Run' a few times to get different coloured turtles.

- That's fun, but it's a lot to remember and type every time you want to set a turtle to a random colour and it's not very easy to read.

  In Python we can write `def` to define a function that we can call whenever we need to set the turtle to a random colour.

  You've been calling functions already, `color()` and `randint()` are functions that have been defined for you.

  Let's put the random colour code into a function using def:

Make sure you indent the code inside the function. Functions are usually placed at the top of the script after the imports.

- If you 'Run' your code now you don't get a random coloured turtle. That's because you have defined your function, but not called it yet.
- Add a line to call your new function:

Notice that your new code is much easier to understand because the complex part is in the function. It's easy to work out what `randomcolour()` does.

Step 3 Random place

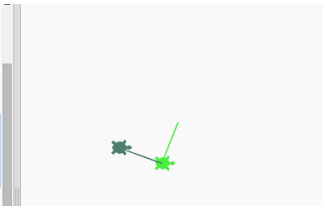
Let's create another function to move the turtle to a random place on the screen. The center of the screen is (0,0) so we'll place turtles in a square area around the centre.

- Add a `randomplace()` function:

```
def randomcolour():
  red = randint(0, 255)
  green = randint(0, 255)
  blue = randint(0, 255)
  color(red, green, blue)

def randomplace():
  x = randint(-100, 100)
  y = randint(-100, 100)
  goto(x, y)

shape("turtle")
randomcolour()
```

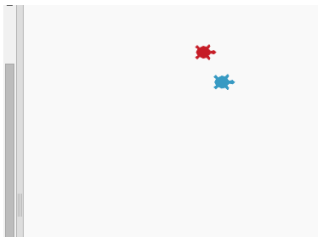
- Try your new function by calling it and then calling `stamp()`, you can call it more than once:

```
def randomplace():
  x = randint(-100, 100)
  y = randint(-100, 100)
  goto(x, y)

shape("turtle")
randomcolour()
randomplace()
stamp()
randomcolour()
randomplace()
stamp()
```



- Ooops, the turtle draws when it moves. Let's put the pen up at the beginning and down at the end so that the turtle doesn't draw while it's moving:

```
def randomplace():
  penup()
  x = randint(-100, 100)
  y = randint(-100, 100)
  goto(x, y)
  pendown()

shape("turtle")
randomcolour()
randomplace()
stamp()
randomcolour()
randomplace()
stamp()
```



  Did you notice that you only had to 'fix' the code in one place? That's another good thing about functions.

- Now test your code a few times.

Step 4 Challenge: Turtle art

Can you define a `randomheading()` function that will make the turtle point in a random direction and make the following code work?

```
shape("turtle")

for i in range(30):
  randomcolour()
  randomplace()
  randomheading()
  stamp()
```

Hints:

- `setheading(<number>)` will change the direction the turtle is facing in.

- `<number>` should be between 1 and 360 (the number of degrees in a circle)

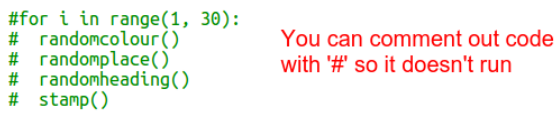
- You can use `randint(1, 360)` to choose a number between 1 and 360.

Step 5 Create rectangle modern art

Now let's create some modern art by drawing lots of rectangles of different sizes and colours.

- First add the following code to the bottom of your script, after your challenge code, to clear the screen after your turtle art and point the turtle in its usual direction:

- You can comment out your turtle art code by placing a `#` at the beginning of each line so that it doesn't run while you are working on rectangle art. (Then you can uncomment it later to show off all of your work.)



- Now let's add a function to draw a random-sized, random-coloured rectangle at a random location!

  Add a `drawrectangle()` function after your other functions:

```
def drawrectangle():
  hideturtle()
  length = randint(10, 100)
  height = randint(10, 100)
  begin_fill()
  forward(length)
  right(90)
  forward(height)
  right(90)
  forward(length)
  right(90)
  forward(height)
  right(90)
  end_fill()
```

Look in `snippets.py` for some helper code if you want to save some typing time.

- Add the following code at the bottom of `main.py` to call your new function:

Run your script a few times to see the height and width change.

- The rectangle is always the same colour and starts at the same location.

Now you'll need to set the turtle to a random colour and then move it to a random place. Hey, didn't you already create functions to do that? Awesome. You can just call them from the beginning of the drawrectangle function:

Wow that was a lot less work, and it's much easier to read.

- Now let's call `drawrectangle()` in a loop to create some cool modern art:

- Gosh that was a bit slow wasn't it! Luckily you can speed the turtle up.

  Find the line where you set the shape to 'turtle' and add the highlighted code:

```
shape("turtle")
speed(0)
```

  `speed(0)` is the fastest or you can use numbers from 1 (slow) to 10 (fast.) Experiment until you find a speed you like.

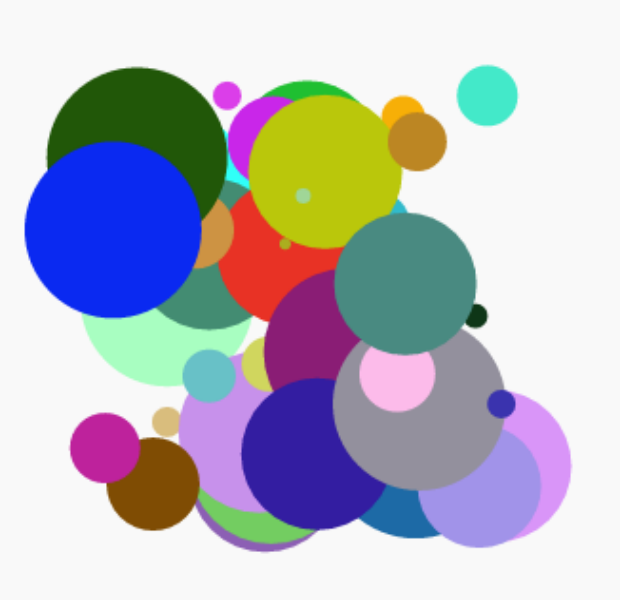
Step 6 Challenge: More modern art

Can you create a function that draws a shape and calls your `randomcolour()` and / or `randomplace()` functions?

You can call your function from inside a `for` loop as you did in the rectangle art to generate modern art.

Ideas:

- Turtles have a function called dot that takes a radius (distance from the centre to the edge of the circle) as input. E.g. turtle.dot(10) You could create a `drawcircle()` function that draws a circle with a random radius.



- Look in `snippets.py` for example code to draw stars with the turtle.