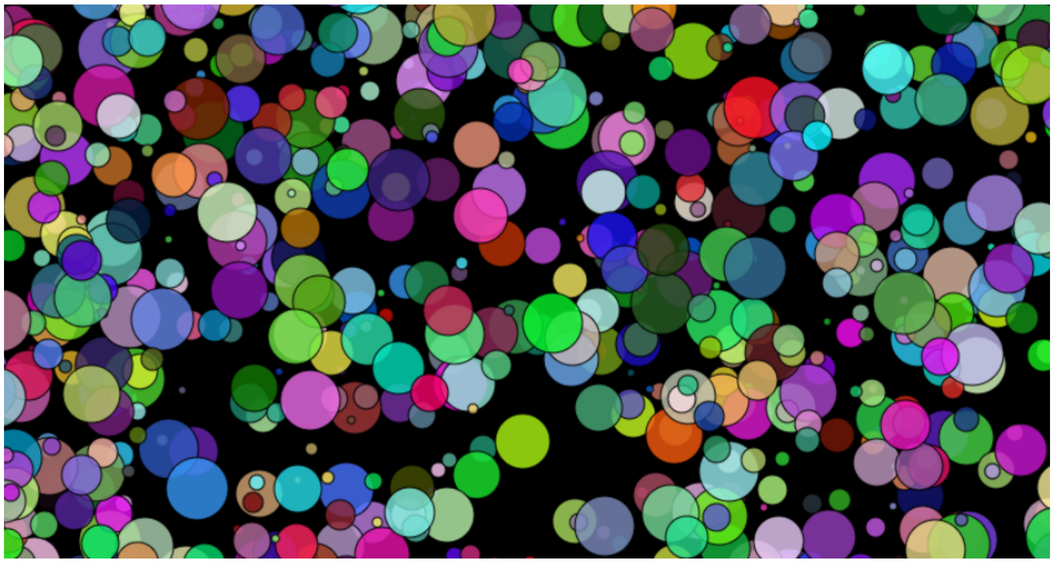
Animating Shapes

Workbook

Programming Tutorials by PyAngelo

https://www.pyangelo.com/tutorials/animating-shapes



Name:______________________

Task 1 - Variables

Initialising Variables

In Python, variables are used to store information. Variables often store information that changes as our program runs. Each variable is given a name and a value. To set the value stored by the variable we use the assignment operator (=). In the following program, line 4 takes the value 20 and stores it in the variable x.

```
setCanvasSize(640, 360)
background(128, 0, 128)
fill(255, 255, 0)
x = 20
circle(x, 180, 20)
```

In Python the assignment operator is the equals sign (=) but it is important to note that this does not have the same meaning as in mathematics. When you see a line of code like:

```
x = 20
```

You can read this as “x gets the value 20”. Now that this variable has been created and holds the value 20, we can use it in subsequent commands. When the computer reads the fifth line of code, it finds the “x” and then checks if a variable called x has been set. Since in our example we have already assigned x the value of 20, the program uses the value 20 instead of x (what would happen if we had not set the value of x?).

Type the program into the PyAngelo editor and click the start button. You will see the circle being drawn with its centre at (20, 180).

Now update the program to set x to 340 like so:

```
setCanvasSize(640, 360)
background(128, 0, 128)
fill(255, 255, 0)
x = 340
circle(x, 180, 20)
```

Now that x holds the value 340, the circle is drawn at (340, 80).

Note: Initialising a variable is a technical programming term that means create a variable and give it an initial value.

Updating Variables

Once a variable has been initialised, we can update the value it stores. For example the following program creates a variable called x and updates it numerous times:

```
setCanvasSize(640, 360)
background(128, 0, 128)
fill(255, 255, 0)
x = 20
circle(x, 180, 20)
x = 80
circle(x, 180, 20)
x = 140
circle(x, 180, 20)
x = 200
circle(x, 180, 20)
x = 260
circle(x, 180, 20)
x = 320
circle(x, 180, 20)
```

Notice that each time the circle is drawn, x has a new value and hence the circle is drawn in a new position.

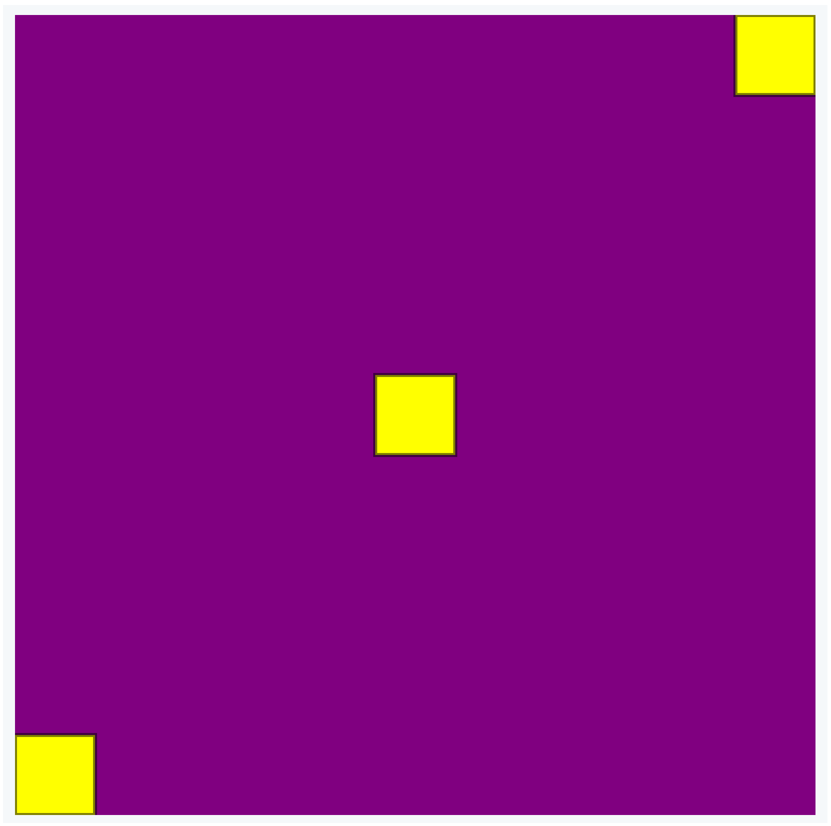
Task 1 Challenge

Your task is to :

- Create a canvas that is 400 pixels wide and 400 pixels high
- Set the background colour of the canvas to a colour of your choice
- Create a variable named x and set its value to 0
- Create a variable named y and set its value to 0

- Draw a square at (x, y) with a length and width of 40
- Give x a value of 180 and y a value of 180
- Draw a square at (x, y) with a length and width of 40
- Give x a value of 360 and y a value of 360
- Draw a square at (x, y) with a length and width of 40

Your final sketch should look like this when run:



Check Your Knowledge - Task 1

Question 1a

A variable has a ____________ and stores ________________.

Question 1b

What symbol is the assignment operator in Python?

____

Question 1c

How would you set a variable called score to have a value of zero?

____

Question 1d

What is the final value of the variable named cost when the following lines of code are run:

cost = 20

cost = 50

____

Check Your Knowledge Answers - Task 1

Question 1a

A variable has a **name** and stores **information**.

Question 1b

=

Question 1c

score = 0

Question 1d

50

Task 2 - Animation the Hard Way

Assignment Operator

```
x = 20
```

As we mentioned earlier, the line above takes the value on the right hand side of the assignment operator, and stores it in the variable on the left hand side.

```
x = x + 10
```

The line above does something similar. It takes the value on the right side which in this case is whatever is stored in the variable x plus 10. If x was 20, then after we add 10 we get the value 30, and this is then stored in the variable x. Hence the line above always adds 10 to the current value of x, whatever that is. Hence we can modify a program from Task 1 to use this type of assignment.

```
setCanvasSize(640, 360)
background(128, 0, 128)
fill(255, 255, 0)
x = 20
circle(x, 180, 20)
x = x + 60
circle(x, 180, 20)
x = x + 60
circle(x, 180, 20)
x = x + 60
circle(x, 180, 20)
x = x + 60
circle(x, 180, 20)
x = x + 60
circle(x, 180, 20)
```

sleep(seconds)

The sleep function pauses our program for the number of seconds we specify. Let's use the sleep function to create a one second delay after we draw each circle.

```
setCanvasSize(640, 360)
background(128, 0, 128)
fill(255, 255, 0)
x = 20
circle(x, 180, 20)
sleep(1)
x = x + 60
circle(x, 180, 20)
sleep(1)
x = x + 60
circle(x, 180, 20)
sleep(1)
x = x + 60
circle(x, 180, 20)
sleep(1)
x = x + 60
circle(x, 180, 20)
sleep(1)
x = x + 60
circle(x, 180, 20)
sleep(1)
```

Our First Animation

To create an animation we need to repeat the following steps over and over again:

- Clear the screen
- Update the position of the circle (the x variable in this example)
- Draw the circle
- Sleep for a small amount of time

```
setCanvasSize(640, 360)
fill(255, 255, 0)

background(128, 0, 128)
x = 20
circle(x, 180, 20)
sleep(1)

background(128, 0, 128)
x = x + 60
circle(x, 180, 20)
sleep(1)

background(128, 0, 128)
x = x + 60
circle(x, 180, 20)
sleep(1)

background(128, 0, 128)
x = x + 60
circle(x, 180, 20)
sleep(1)

background(128, 0, 128)
x = x + 60
circle(x, 180, 20)
sleep(1)

background(128, 0, 128)
x = x + 60
circle(x, 180, 20)
sleep(1)
```

Task 2 Challenge

Modify the program above so that the ball moves across the entire screen. Try adjusting the delay in seconds after each circle is drawn by modifying the sleep(1) command. What happens if you set the delay to 2 seconds? What happens if you set the delay to 0.1 seconds? Try modifying how much you add to x each time.

Check Your Knowledge - Task 2

Question 2a

What is the final value of the variable score after the following 3 lines of code are run?

```
score = 0

score = score + 100

score = score + 50
```

Question 2b

What does the following command do?

```
sleep(10)
```

Question 2c

Why do we need to clear the screen before drawing our objects in their new positions in order to create an animation?

Question 2d

What is the final value of the variable x after the following 2 lines of code are run?

```
x = 20

x = x + 60
```

Check Your Knowledge Answers - Task 2

Question 2a

150

Question 2b

It causes the program to sleep for 10 seconds before executing the next line of code

Question 2c

If we did not clear the screen, you would see all the drawings of the objects at their old positions as well as the new drawing in its current position. The clearing of the screen followed by drawing an object in a different position is what creates the illusion of motion.

Question 2d

80

Task 3 - Animation Loop

forever:

In PyAngelo there is a special command that does not exist in Python. It is the forever command and it must be followed by a colon. The colon indicates that there is a block of code that will be underneath this line and indented. In Python the normal way of indenting code is using the TAB key and having the TAB key set to 4 spaces.

With the forever loop, the block of code underneath and indented will be repeated forever! This is what we call an infinite loop because it never ends. That also means that any code written below this loop (code that is not indented) will never be reached because the program keeps repeating the indented block of code.

This infinite loop is not always a good idea as the program never ends. However, for animations it can be useful. In fact, when creating animations we call this type of infinite loop the animation loop.

A basic formula for animating an object is:

1. Clear the canvas
2. Update the position of the object
3. Draw the object in its new position
4. Sleep for a small amount of time

Did you notice in our last task, we repeated the following code over and over:

```
background(128, 0, 128)
x = x + 60
circle(x, 180, 20)
sleep(1)
```

Instead of typing the same 4 lines many times, let's see if we can rewrite our animation to take advantage of an animation loop using the forever command.

```
setCanvasSize(640, 360)
fill(255, 255, 0)
x = -1
speedX = 1
delaySeconds = 0.005

forever:
    # Step 1 - Clear the canvas
    background(128, 0, 128)
    # Step 2 - Update the position of the circle
    x = x + speedX
    # Step 3 - Draw the circle in its new position
    circle(x, 180, 20)
    # Step 4 - sleep for a small amount of time
    sleep(delaySeconds)
```

This is a much more efficient way to code our animation. The 4 steps of clearing the canvas, updating the position of the circle, drawing the circle, and sleeping for a small amount of time are repeated forever. We know it is these lines of code that are repeated because they are indented and underneath the forever: command.

Well done. You have just coded your first animation!

Task 3 Challenge

Modify the program above so that the ball starts at the bottom left of the screen and moves diagonally up and across the canvas. See if you can make the ball move faster across the screen.

Check Your Knowledge - Task 3

Question 3a

What special character is required immediately after forever, and what does it signify?

Question 3b

What is wrong with the following lines of code?

```
forever:
circle(50, 75, 10)
sleep(0.005)
```

Question 3c

What is the first step each iteration of the animation loop?

Question 3d

What is the last step each iteration of the animation loop?

Question 3e

Which key is recommended in PyAngelo to create indented code?

Check Your Knowledge Answers - Task 3

Question 3a

The colon character (:). This signifies that there will be an indented block of code below which due to the forever loop will be repeated infinitely.

Question 3b

There needs to be a block of code underneath the forever: command that is indented. The code should be written as:

```
forever:
    circle(50, 75, 10)
    sleep(0.005)
```

Question 3c

The first step in the animation loop is to clear the canvas.

Question 3d

The last step in the animation loop is to sleep for a small amount of time. This allows your browser to render all of the objects to the screen and also allows your browser to respond to user events such as mouse clicks or keyboard input.

Question 3e

The TAB key is recommended for creating indented code in PyAngelo. In the online editor the TAB key is configured to insert 4 spaces. If you type a colon at the end of a line and hit the ENTER key, the editor will automatically create an indentation for you.

Task 4 - Respawning Ball

In our previous task we managed to create an animation that moved the ball from left to right. However, when the ball went off the right edge of the screen it simply kept going further and further off the screen. Since the animation loop repeats forever the program does not end by itself and so a good idea might be to make the ball respawn from the left edge whenever it has gone off the right edge.

Checking if the ball has gone off the canvas

To respawn the ball means to reset its position once it has left the canvas. This means after we update the variable storing the position of the ball, we should check to see if that would take it off the right edge. If this is the case we can set the value of the variable to be such that the ball's position is now to the left side of the canvas.

If statements

In Python, if statements are used to take different actions based on a condition. This is why they are often referred to as conditional statements. The condition must be one that is either True or False. If the condition is True, then the block of code that is underneath and indented will be executed. If the condition evaluates to False, the indented block of code will be skipped and not run.

Here is an example if statement:

```
if x >= 660:
    x = -20
```

Note that the statement starts with the keyword if followed by a condition, and then finally a colon (:). The colon signifies there will be an indented block of code below that gets executed only if the condition evaluates to True. The above if statements checks if the x variable is greater than or equal to 660. If that is True, then the x variable gets the value -20. In our previous task we used the variable x to store the location of the centre of the circle. Hence the if statement would in effect cause the ball to respawn from the left edge.

Here is the complete program that uses if statements to respawn the ball:

```
setCanvasSize(640, 360)
fill(255, 255, 0)
x = -1
speedX = 1
delaySeconds = 0.005

forever:
    # Step 1 - Clear the canvas
    background(128, 0, 128)
    # Step 2 - Update the position of the circle
    x = x + speedX
    if x >= 660:
        x = -20
    # Step 3 - Draw the circle in its new position
    circle(x, 180, 20)
    # Step 4 - sleep for a small amount of time
    sleep(delaySeconds)
```

Types of checks

To check if a value is equal to another value we need to use the equivalence operator which is two equals signs (==). We do not use a single equals sign as this is used for assignment when we wish to set the value of a variable. Here is an example that checks if the variable x is equal to 50 and if so changes the fill colour to yellow.

```
if x == 50:
    fill(255, 255, 0)
```

Here is a list of the comparison operators you can use:

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equivalent to
!=	Not equal to

Task 4 Challenge

Modify the respawning ball program from above so that the ball starts at the bottom left of the screen and moves diagonally up and across the canvas. Add in another check to see if the ball has gone off the top of the canvas and if so respawn the ball at the bottom. You will need to create a new variable to keep track of the y position of the ball and add an additional if statement for the check.

Check Your Knowledge - Task 4

Question 4a

What special character is required at the end of an if statement, and what does it signify?

Question 4b

What is wrong with the following lines of code?

```
if y >= 380
y = -20
```

Question 4c

What is the value of score after the following code is run?

```
score = 50
timeLeft = 10
if timeLeft >= 10:
   score = score + 100
if timeLeft < 5:
   score = score - 50
```

Question 4d

Which operator is used to test if a variable is greater than a value?

Question 4e

Which operator is used to test if a variable is equivalent to a value?

Check Your Knowledge Answers - Task 4

Question 4a

The colon character (:). This signifies that there will be an indented block of code below which will only be executed if the condition of the if statement evaluates to True.

Question 4b

There needs to be a colon at the end of the line beginning with if. Also there needs to be a block of code underneath the if line that is indented. The code should be written as:

```
if y >= 380:
        y = -20
```

Question 4c

150. The first if condition evaluates to True and hence the 100 is added to the value of the score variable. The second if condition evaluates to False and hence the indented line of code decrementing score by 50 is not run.

Question 4d

>

Question 4e

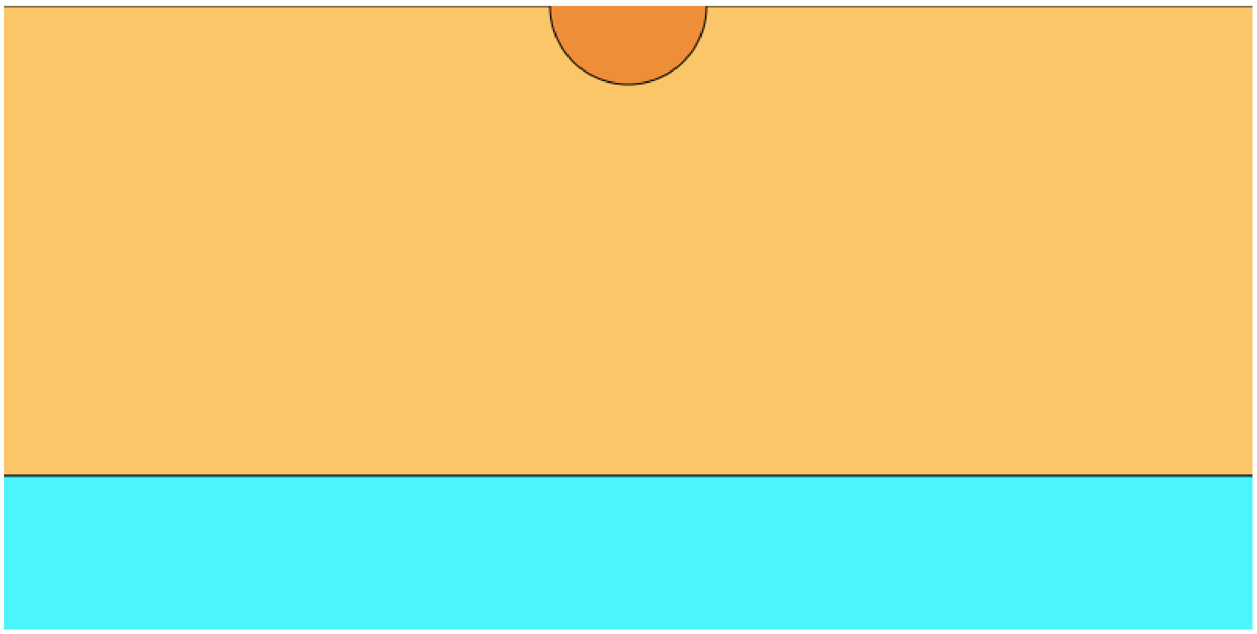
==

Task 5 - Sunrise and Sunset

Now it’s time to put our skills to the test. Our task is to create a graphical program that displays a sunrise followed by a sunset over and over again. Here is some starter code that we will work from:

```
setCanvasSize(800, 400)
x = 400
y = 400

forever:
    # Sky
    fill(250, 198, 104)
    rect(0, 100, 800, 300)

    # Sun
    fill(239, 142, 56)
    circle(x, y, 50)

    # Water
    fill(75, 245, 251)
    rect(0, 0, 800, 100)

    sleep(0.005)
```

If you type this into the PyAngelo editor and run the program you will see the following drawn to the canvas:



Coding the Sunset

To code the sunset we need to adjust the variables x and y. The variable x needs to increase each time through the animation loop in order to move the sun to the right. The variable y needs to decrease each time through the animation loop in order to move the sun downwards. See if you update the program to add this functionality.

Coding the Sunrise

In order to code the sunrise we need to first check when the sunset has finished. One way to do this is by checking the value of the x variable. If the value of the x variable is greater than 800, we know the sun has moved to the bottom right corner of the screen and will be hidden behind the water. The reason it will be hidden behind the water is the order that the objects are drawn. Since the water is drawn last, it will be drawn on top of the sun.

Your task is to add an if statement that checks if the sunset has completed and if so it should set the x and y variables both to 0 to position the sun ready to rise. As the sun is now meant to be rising, we should no longer decrement the y variable but instead increment it. A good way to do this is by creating a variable that holds the y direction of 1 or -1 and using this variable to update y.

```
yDir = -1
y = y + yDir
```

The code above decrements the value of y since yDir is a negative number. The line y = y +yDir can be used inside the animation loop. If we update the value of yDir to be positive 1, then the line y = y + yDir will now increment the value of y by 1 which is what we want for our sunrise. You should switch the value of the variable yDir when the sunset has finished. Note that you will also need an if statement to check when the sunrise has finished and then switch the value of yDir back to negative 1 to start the sunset once again.

Try to modify your sketch so that the sun continuously rises and sets by implementing the ideas we discussed above in your code. Once you've had a go, take a look at how I modified the code to incorporate these changes below:

```
setCanvasSize(800, 400)
x = 400
y = 400
yDir = -1

forever:
    background(0, 0, 0)
    # Sky
    fill(250, 198, 104)
    rect(0, 100, 800, 300)

    # Sun
    x = x + 1
    y = y + yDir
    if x > 1000:
        x = 0
        y = 0
        yDir = 1

    if x == 400:
        yDir = -1
    fill(239, 142, 56)
    circle(x, y, 50)
```

```
# Water
fill(0, 0, 0)
rect(0, 0, 800, 100)
fill(75, 245, 251)
rect(0, 0, 800, 100)

sleep(0.005)
```

Task 5 Challenge

Modify the sunrise and sunset sketch so that the colours fade to black as the sun sets, and brighten as the sun rises. One way to approach this is by using the alpha argument to the fill() function. The alpha argument lets you set the transparency in the range of 0 to 1. Specifying 0 means the colour is fully transparent and hence will not be seen. Setting the value to 1 means the colour is not at all transparent and nothing behind it will be seen. If you were to make the background black before drawing the sky and water, you could then set the alpha value to less than 1 to start to see the black background through the colours. If the alpha is set to 0 you would only see the black background. Take a look at the reference page for more information on the alpha argument to the fill() function. If you need help with this challenge then watch the associated video on the PyAngelo website:
https://www.pyangelo.com/tutorials/animating-shapes/sunrise-and-sunset

Check Your Knowledge - Task 5

Question 5a

What will the value of y be after the following lines of code are run?

```
y = 401

yDir = -1

y = y + yDir
```

Question 5b

What does the following lines of code do:

```
if y == 400:

        yDir = -1
```

Question 5c

What function does the fourth argument to the fill() function have and what range of values can you give it?

```
fill(255, 255, 0, 0.5)
```

Question 5d

What effect does the order objects are drawn in have in a PyAngelo sketch?

Check Your Knowledge Answers - Task 5

Question 5a

400

Question 5b

The code checks to see if the value of the variable y is equivalent to 400 and if so it updates the value of yDir to be -1. If the value of y is not equivalent to 400 then the value of yDir is not changed. The point of this code is to change the value of yDir to negative 1 so that the sun will start moving down the screen when the line y = y + yDir is executed

Question 5c

The fill() function, and other colour functions such as stroke() and background(), can be passed a fourth argument that specifies the alpha of the colour. The value specified can be between 0 and 1. A value of 0 means the colour is fully transparent and a value of 1 means the colour is not transparent at all. Hence a value of 0.5 would mean the colour is displayed but is partially transparent and hence colours below can be seen.

Question 5d

The order objects are drawn in is important. When you draw an object, it will be drawn on top of any previous objects on the canvas that it overlaps with.

Task 6 - mouseX and mouseY Variables

mouseX and mouseY

The mouseX and mouseY variables are automatically created and updated by PyAngelo. mouseX holds the x position of the mouse and mouseY holds the y position. We can use these variables wherever we would use variables we have created ourselves including for drawing shapes. The following sketch draws a circle wherever the mouse currently is:

```
setCanvasSize(640, 360)

forever:
    background(0, 0, 0)
    fill(255, 0, 128)
    circle(mouseX, mouseY, 20)
    sleep(0.005)
```

Do not clear the canvas in the animation loop

What do you think will happen if we move the background(0, 0, 0) function call to outside the forever loop? Type the following into your editor and run the sketch to see what happens:

```
setCanvasSize(640, 360)
background(0, 0, 0)

forever:
    fill(255, 0, 128)
    circle(mouseX, mouseY, 20)
    sleep(0.005)
```

Since the canvas is only cleared at the start of the program, every circle that is drawn stays on the canvas. Each time through the animation loop a new circle is drawn where the mouse is currently located. In fact, this could be the start of a paint program.

mouseIsPressed

PyAngelo also has a variable called mouseIsPressed. This variable is what is known as a boolean variable as it can have only two values, True or False. Since the variable is either True or False, we can use it in an if statement to check if the mouse is currently pressed. We can modify the program above to clear the screen whenever the mouseIsPressed like so:

```
setCanvasSize(640, 360)
background(0, 0, 0)

forever:
    fill(255, 0, 128)
    circle(mouseX, mouseY, 20)
    if mouseIsPressed:
        background(0, 0, 0)
    sleep(0.005)
```

Task 6 Challenge

Modify the above sketch so that a circle is only drawn when the mouse is being pressed. Ensure that the circle is still drawn at the location of the mouse.

Check Your Knowledge - Task 6

Question 6a

What is the name of the inbuilt variable that keeps track of the current x position of the mouse?

Question 6b

What does the following lines of code do:

```
if mouseIsPressed:
        fill(255, 255, 0)
```

Question 6c

What is the name of the inbuilt variable that keeps track of the current y position of the mouse?

Question 6d

What is the effect of moving the background() function call from inside the forever loop to above it?

Check Your Knowledge Answers - Task 6

Question 6a

mouseX

Question 6b

It sets the fill colour to yellow if the mouse is being pressed. If the mouse is not pressed then the fill() function is not called.

Question 6c

mouseY

Question 6d

If the background function is not called inside the forever loop then the canvas is not cleared. This has the effect that everything that is drawn on the canvas remains on the canvas.

Task 7 - if, elif, and else

If and Else

Earlier we learned how we could use an if statement to execute some code only if the specified condition was True. We can extend the if statement to run some code if the condition was not True by using the else clause.

```
setCanvasSize(640, 360)
fill(255, 0, 128)

forever:
    background(0, 0, 0)
    if mouseX > 320:
        fill(255, 255, 0)
    else:
        fill(255, 0, 255)
    circle(320, 180, 20)
    sleep(0.005)
```

The above sketch sets the fill colour to yellow if the mouseX variable is greater than 320, however if the mouseX is not greater than 320, meaning it is 320 or less, then the fill colour is set to purple. Notice that the “else:” line needs a colon to indicate that there will be a block of code underneath and indented.

The important thing to remember is that the indented block of code below the “else:” line is only executed if the original condition evaluates to False. With the new “else:” clause we can execute two different sections of code. What happens if we need more options?

Elif

We can add more options using the “elif” clause. An elif allows us to specify a new condition to evaluate. We can add as many elif conditions as required. The following program sets the fill colour to a different value based on the location of the mouse.

```
setCanvasSize(640, 360)
fill(255, 0, 128)

forever:
   background(0, 0, 0)
   if mouseX > 300:
      fill(255, 0, 0)
   elif mouseX > 200:
      fill(0, 255, 0)
   elif mouseX > 100:
      fill(0, 0, 255)
   else:
      fill(255, 255, 0)
   circle(320, 180, 20)
   sleep(0.005)
```

The combination of the starting if, as many elif's as required and an optional else are considered a single statement and as soon as the first matching condition is found, the indented block of code is executed and then the code skips to the end of the if statement and then executes the next line of code. This is important to understand because if you look at the first condition above, it checks if mouseX is greater than 300. The next condition checks if the mouseX is greater than 200. However, if the mouseX is greater than 300 it must also be greater than 200 but that does not matter because only the indented block associated with the first matching condition is executed. If no conditions evaluate to True then the "else:" clause will be executed.

Task 7 Challenge

Modify the above sketch so that the background colour is changed based on the y position of the mouse.

Check Your Knowledge - Task 7

Question 7a

How many elif clauses can you include in an if statement?

Question 7b

What will be the colour of the circle after the following code is run?

```
setCanvasSize(640, 360)

score = 10
if score > 20:
    fill(255, 0, 0)
elif score > 10:
    fill(0, 255, 0)
else:
    fill(0, 0, 255)
circle(100, 200, 20)
```

Question 7c

What will be the value of savings after the following code is run? What is the logic error in this code?

```
savings = 100
if savings > 20:
    savings = savings + 10
elif saving > 50:
    savings = savings + 20
```

Check Your Knowledge Answers - Task 7

Question 7a

As many as required. There is no limit on the number of elif clauses you can add to an if statement.

Question 7b

Blue. Since the score is not greater than 20 and also is not greater than 10, the else condition will be executed which sets the fill colour to (0, 0, 255) which is a blue colour.

Question 7c

The value of savings will be 110. The logic error is that the savings = savings + 20 line will never execute. This is because savings is greater than 20 and hence the first conditional block will be executed and the rest of the if statement will be ignored. One way to fix this would be to change the code to:

```
savings = 100
if savings > 50:
    savings = savings + 20
elif saving > 20:
    savings = savings + 10
```

Task 8 - Random Screensaver

Using Other People's Code

One advantage of using Python is that there is a lot of code that other people have written that we can use in our own programs. Code is usually bundled together into what is called a library or a module. One useful module is called random and it contains a bunch of functions that let us introduce randomness into our programs.

Random Library

One way to import a module is to use the from statement.

```
from random import *
```

This imports the entire random library into your code. The "*" is used to represent everything. If you only wish to use a single function you can specify that function instead of the "*". For example, in our next task we will be using the randint function which picks a random integer value. To import this function the following code is used.

```
from random import randint
```

By convention these statements are usually included at the top of your program.

randint()

The randint() function takes 2 arguments; the minimum number and the maximum number. It returns a random integer between these two numbers which you can store in a variable. An integer is a whole number such as 10, 0, or -3. An example of a number that is not an integer is 3.14 as it contains a decimal point.

```
from random import randint
num = randint(1, 10)
```

The above code selects a random number between 1 and 10 and stores it in the variable num.

A Screensaver

A screensaver is a program that fills the screen with moving images or patterns when the computer has been idle for a designated period of time. With older monitors this was in fact to save the screen. These days the monitors do not need to be saved! However, it is fun to build a screensaver and we will do so by using our new found friend, the randint() function. Type the following code into the PyAngelo editor and run it. You should see circles of random sizes being drawn in random positions.

```
from random import randint
setCanvasSize(640, 360)
background(0, 0, 0)
forever:
    fill(180, 0, 180)
    x = randint(0, 640)
    y = randint(0, 360)
    radius = randint(5, 30)
    circle(x, y, radius)
    sleep(0.1)
```

Take the time to read each line of code carefully and ensure you understand what is happening in the program.

Task 8 Challenge

Modify the screensaver program so that each circle is a random colour. To do this you will need to modify the fill() function. Remember that the fill() function takes 3 arguments, the amount of red, green, and blue. The amount is a number from 0 to 255. So one way to achieve random colours is to pick 3 random integers between 0 and 255 and use these in the fill() function. If you need help with this challenge then watch the associated video on the PyAngelo website: [Random Screensaver | Animating Shapes | PyAngelo]

Here are some ideas to experiment with your screensaver program:

- Add some transparency to the circles' colour
- Remove the stroke or border from each circle
- 50% of the time draw a circle and 50% of the time draw a square

width and height Variables

In PyAngelo, when you call the setCanvasSize() function, two variables are automatically created for you.

- width - this variable holds the width of the canvas
- height - this variable holds the height of the canvas

It is good programming practice to use these variables rather than hardcoding a number because if you at some stage decide to change the width and height of the canvas these variables are automatically updated and so you won't need to modify all the occurrences of your hard coded numbers. So in our screensaver program we should use the following lines of code when selecting a random position for our circle:

```
x = randint(0, width)
y = randint(0, height)
```

Check Your Knowledge - Task 8

Question 8a

What does the “*” represent in the following statement:

```
from random import *
```

Question 8b

What will be stored in the variable x after the following code is run?

```
from random import randint
x = randint(1, 100)
```

Question 8c

What statement imports only the randint() function from the random library?

Question 8d

What will the value of the inbuilt variables width and height be after the following code is run?

```
setCanvasSize(200, 100)
```

Check Your Knowledge Answers - Task 8

Question 8a

The “*” represents the entire library. Hence the “from” statement above imports the entire random library into the program.

Question 8b

The variable x will store a random integer between 1 and 100.

Question 8c

from random import randint

Question 8d

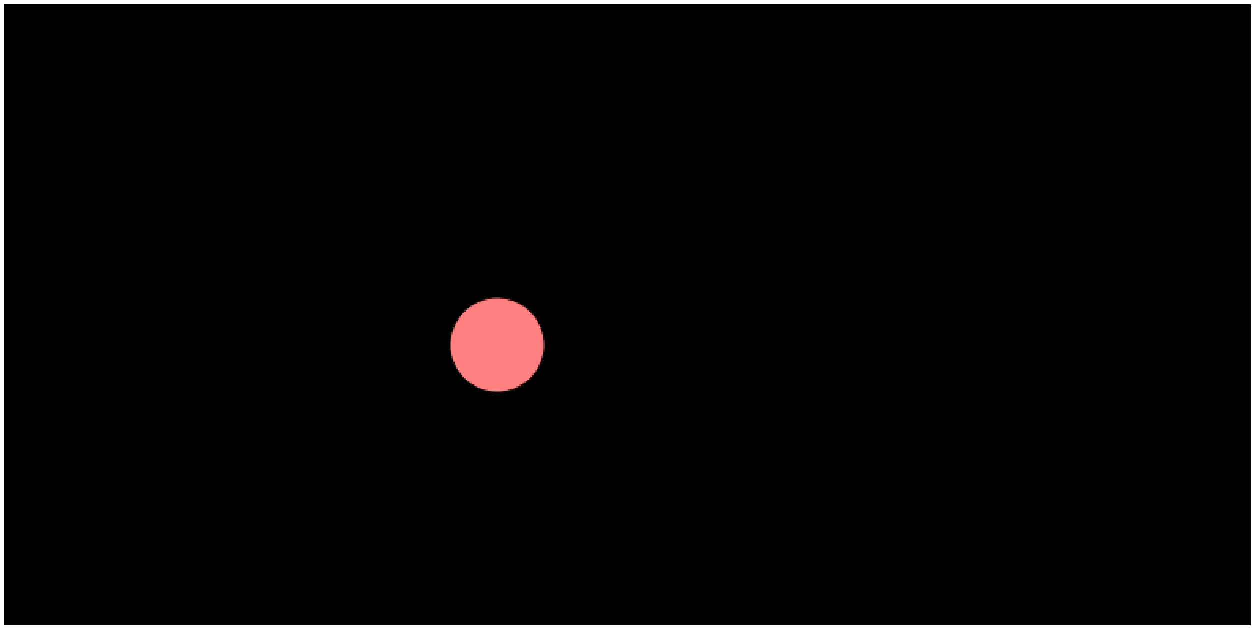
The variable width will store the value 200 and the variable height will store the value 100.

Task 9 - Bouncing Ball Animation

Once again it is time to put our skills to the test. Our task is to create the classic bouncing ball program. The ball should move around the screen and bounce off the edges of our canvas. Here is the starter code for this task:

```
setCanvasSize(800, 400)
fill(255, 128, 128)
x = 320
y = 180

forever:
    background(0, 0, 0)
    circle(x, y, 30)
    sleep(0.005)
```

If you type this into the PyAngelo editor and run the program you will see the following drawn to the canvas:



Task 9 Challenge

Modify the program so that the ball:

- Starts moving in a random direction
- Bounces off the edges of the canvas
- Changes to a random colour each time it bounces on the edge

If you need help with this challenge then watch the associated video on the PyAngelo website:
https://www.pyangelo.com/tutorials/animating-shapes/bouncing-ball-animation

Check Your Knowledge - Task 9

Question 9a

What does the following if statement check? Assume the variable y is used to hold the y position of the centre of the ball.

```
if y >= height - radius:
    yDir = yDir * -1
```

Question 9b

What does the following if statement check? Assume the variable y is used to hold the y position of the centre of the ball.

```
if y < radius:
    yDir = yDir * -1
```

Question 9c

Explain what the purpose of the following line of code is:

```
yDir = yDir * -1
```

Check Your Knowledge Answers - Task 9

Question 9a

The if statement is checking if the ball has moved past the top of the canvas.

Question 9b

The if statement is checking if the ball have moved past the bottom of the canvas.

Question 9c

This changes the sign of the yDir variable. Assuming the yDir variable is used to change the y-position of the ball, the line of code would reverse the vertical direction of the ball. For example if yDir was equal to 2, then the line of code would change it to -2 and if yDir was -2 then it would be changed to 2.

Task 10 - Two Bouncing Balls

As a last extension task in this series on animating shapes, see if you can create a program that has 2 bouncing balls moving around the canvas. Note that you can essentially duplicate the code used for a single ball but you must then rename all of the variables. One way to do this would be instead of having x and y variables, to have x1, x2, y1, and y2 variables. This might be OK for 2 balls but what if you have 100 balls? Whenever you find coding a lot of work there is usually a cleaner solution. However, this is still a good exercise for you to complete. Once you have done this the hard way you can move onto the next tutorial series called Animation with Sprites which will introduce you to the world of object oriented programming and show you a neat way to eliminate the need for so many variables. Combine this with the power of lists in Python and you can easily create a program with 100 bouncing balls!