

## Pygame lesson 1

Pygame is used for... making games. It allows you to draw graphics, import graphics, create classes of objects, check the interaction between the user and the screen and between objects on the screen.

We will start with setting up a basic screen, ready for our first simple games.

### Setting up a screen – the basic loop

Type up this code. It won't do much, but we need to start somewhere. You should play around with the colours a bit. The only thing this allows you to do is click on the 'close' button.

```
# Pygame base template - source http://programarcadegames.com/
import pygame

# Define some colors - note that they are tuples
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = (255, 0, 0)

pygame.init()

# Set the width and height of the screen - again, a tuple
size = (700, 500)
screen = pygame.display.set_mode(size)
pygame.display.set_caption("My first game")

# Loop until the user clicks the close button.
done = False

# Used to manage how fast the screen updates - we will set it inside the loop
clock = pygame.time.Clock()

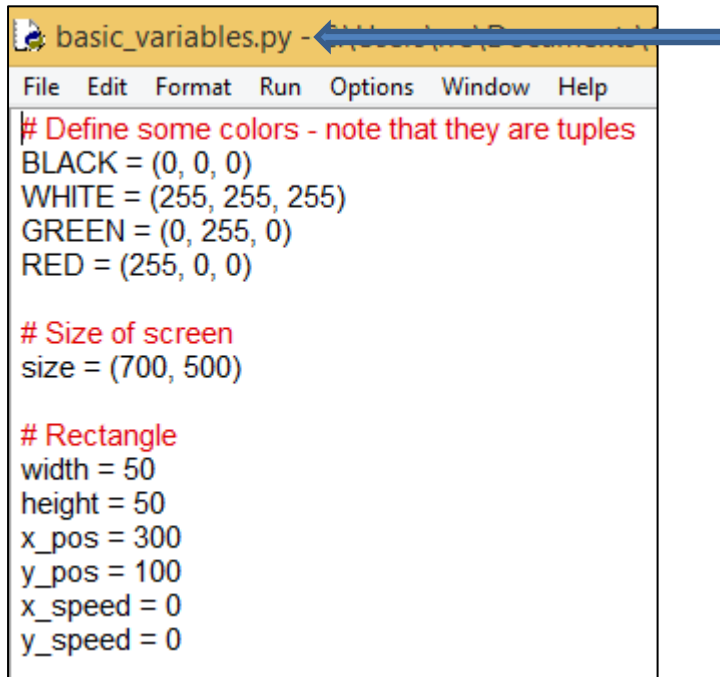
# ----- Main Program Loop -----
while not done:
    # --- Checking for events - key presses
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    # --- Game logic should go here eventually


    # Clear the screen and set it to white.
    screen.fill(WHITE)
    # --- Drawing code should go here
    # --- This updates the screen and sets the frame rate
    pygame.display.flip()
    clock.tick(60)

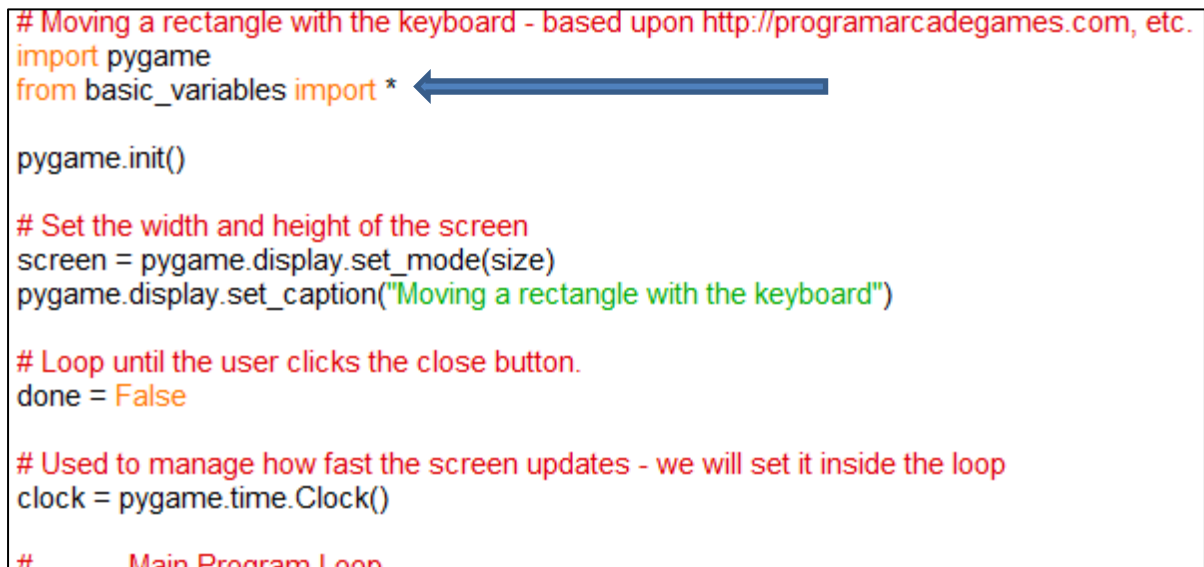
# Close the window and quit - it is important to quit the game
pygame.quit()
```


## Moving a rectangle around with the keyboard

The first part of the game is similar, but I've created an external file with the colour, etc., and imported these into the main program to make it a bit tidier.



```
basic_variables.py -   
File Edit Format Run Options Window Help  
# Define some colors - note that they are tuples  
BLACK = (0, 0, 0)  
WHITE = (255, 255, 255)  
GREEN = (0, 255, 0)  
RED = (255, 0, 0)  
  
# Size of screen  
size = (700, 500)  
  
# Rectangle  
width = 50  
height = 50  
x_pos = 300  
y_pos = 100  
x_speed = 0  
y_speed = 0
```



```
# Moving a rectangle with the keyboard - based upon http://programarcadegames.com, etc.  
import pygame  
from basic_variables import *   
  
pygame.init()  
  
# Set the width and height of the screen  
screen = pygame.display.set_mode(size)  
pygame.display.set_caption("Moving a rectangle with the keyboard")  
  
# Loop until the user clicks the close button.  
done = False  
  
# Used to manage how fast the screen updates - we will set it inside the loop  
clock = pygame.time.Clock()  
  
# Main Program Loop
```

And the rest:

```
# ----- Main Program Loop -----
while not done:
    # --- Event Processing
    for event in pygame.event.get(): # check if key pressed
        if event.type == pygame.QUIT:
            done = True

        elif event.type == pygame.KEYDOWN: # It was an arrow key.
            if event.key == pygame.K_LEFT:
                x_speed = -1
            elif event.key == pygame.K_RIGHT:
                x_speed = 1
            elif event.key == pygame.K_UP:
                y_speed = -1
            elif event.key == pygame.K_DOWN:
                y_speed = 1

    # --- Game logic will go here eventually
    x_pos = x_pos + x_speed
    y_pos = y_pos + y_speed

    # Clear the screen and set it to white.
    screen.fill(WHITE)

    # --- Drawing code
    pygame.draw.rect(screen, RED, [x_pos, y_pos, width, height])

    # --- This updates the screen and sets the frame rate
    pygame.display.flip()
    clock.tick(60)

# Close the window and quit - it is important to quit the game
pygame.quit()
```

## Changing the way the rectangle moves

Instead of it constantly moving, and the keyboard being used to simply change the direction, you can change this so it only moves when you click on an arrow on the keyboard

```
elif event.type == pygame.KEYDOWN: # It was an arrow key.
    if event.key == pygame.K_LEFT:
        x_speed = -3
    elif event.key == pygame.K_RIGHT:
        x_speed = 3
    elif event.key == pygame.K_UP:
        y_speed = -3
    elif event.key == pygame.K_DOWN:
        y_speed = 3

# --- Game logic |
x_pos = x_pos + x_speed
y_pos = y_pos + y_speed
x_speed = 0
y_speed = 0
```

## Drawing other shapes

```
# Drawing various shapes
import pygame
from basic_variables import *

pygame.init()

# Set the width and height of the screen
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Drawing shapes with Pygame")
done = False

# ----- Main Program Loop -----
while not done:
    # --- Event Processing
    for event in pygame.event.get(): # check if key pressed
        if event.type == pygame.QUIT:
            done = True

    # Clear the screen and set it to white.
    screen.fill(WHITE)

    # ----- Drawing code -----
    # Draw on the screen a green line from (0, 0) to (100, 100) that is 5 pixels wide.
    pygame.draw.line(screen, GREEN, [0, 0], [100, 100], 5)

    # --- This updates the screen and sets the frame rate
    pygame.display.flip()
    clock.tick(60)

# Close the window and quit - it is important to quit the game
pygame.quit()
```

You can now draw a range of other shapes, and patterns. Just put the following in the 'drawing code' section above.

Parallel lines using a WHILE loop

```
# ----- Drawing code -----  
# Draw on the screen several lines from (0, 10) to (100, 110)  
# 5 pixels wide using a while loop  
y_offset = 0  
while y_offset < 100:  
    pygame.draw.line(screen, RED, [0, 10+y_offset], [100, 110+y_offset], 5)  
    y_offset = y_offset + 10
```

Parallel lines using a FOR loop

```
# ----- Drawing code -----  
# Draw on the screen several lines from (0,10) to (100,110)  
# 5 pixels wide using a for loop  
for y_offset in range(0, 100, 10):  
    pygame.draw.line(screen, RED, [0, 10+y_offset], [100, 110+y_offset], 5)
```

**Challenge** – produce a 'star' with all lines starting at the middle point

Producing patterns with sin and cos

```
# ----- Drawing code -----  
# For this code, make sure to have a line that says "import math"  
# at the top of your program.  
for i in range(200):  
    radians_x = i / 20  
    radians_y = i / 6  
    x = int(75 * math.sin(radians_x)) + 200  
    y = int(75 * math.cos(radians_y)) + 200  
    pygame.draw.line(screen, BLACK, [x,y], [x+5,y], 5)
```

Ellipse

```
# ----- Drawing code -----  
# Draw an ellipse, using a rectangle as the outside boundaries  
pygame.draw.ellipse(screen, BLACK, [20,20,450,200], 2)
```

Polygon

```
# ----- Drawing code -----  
# This draws a triangle using the polygon command  
pygame.draw.polygon(screen, BLACK, [[100,100], [50,300], [200,400]], 5)
```

## Drawing with functions

More complicated shapes can be drawn with functions, for example:

Added to the top of the program:

```
# drawing functions
def draw_tree():
    pygame.draw.rect(screen, BROWN, [60, 400, 30, 45])
    pygame.draw.polygon(screen, GREEN, [[150, 400], [75, 250], [0, 400]])
    pygame.draw.polygon(screen, GREEN, [[140, 350], [75, 230], [10, 350]])

# ----- Main Program Loop -----
```

Added to the loop:

```
# ----- Drawing code -----
draw_tree()
```

Added to the basic\_variables.py file:

```
BROWN = (165, 42, 42)
```

This tree is a bit silly as it will only draw in one place. Instead, it would be better to create a tree function where we pass it a starting location. This will allow us to create trees wherever we want.

```
# drawing functions
def draw_tree(x, y):
    pygame.draw.rect(screen, BROWN, [x, y, 30, 45])
    pygame.draw.polygon(screen, GREEN, [[x-50,y],[80+x,y], [x+15,y-150]])
    pygame.draw.polygon(screen, GREEN, [[x-50,y-30],[80+x,y-30], [x+15,y-180]])
```

```
# ----- Drawing code -----
draw_tree(50, 400)
draw_tree(150, 420)
```

**Challenge** – draw a scene featuring trees, a little house and a couple of snowmen, all of which are drawn by functions