Checking qualifications

Demonstrate your knowledge of:

- → creating classes;
- → the basics of Pygame;
- → lists and their methods.

Checking qualifications

How do you create a game scene ?

How do you set a background color for it?

How can you set a specific color?

Checking qualifications

A game scene with a colored background

Command	*Purpose*
`window = display.set_mode((500, 500))`	Creates a window with the following size: (width, length).
`window.fill(<color>)`	Fills the background with the specified color
`pygame.display.update()`	Updates the content of the game window
`clock = pygame.time.Clock()`	Creates a game timer.
`clock.tick(40)`	Sets the scene refresh rate to ~40 FPS

Note. The color can be set using the RGB palette.

Checking qualifications

What is a game loop ?

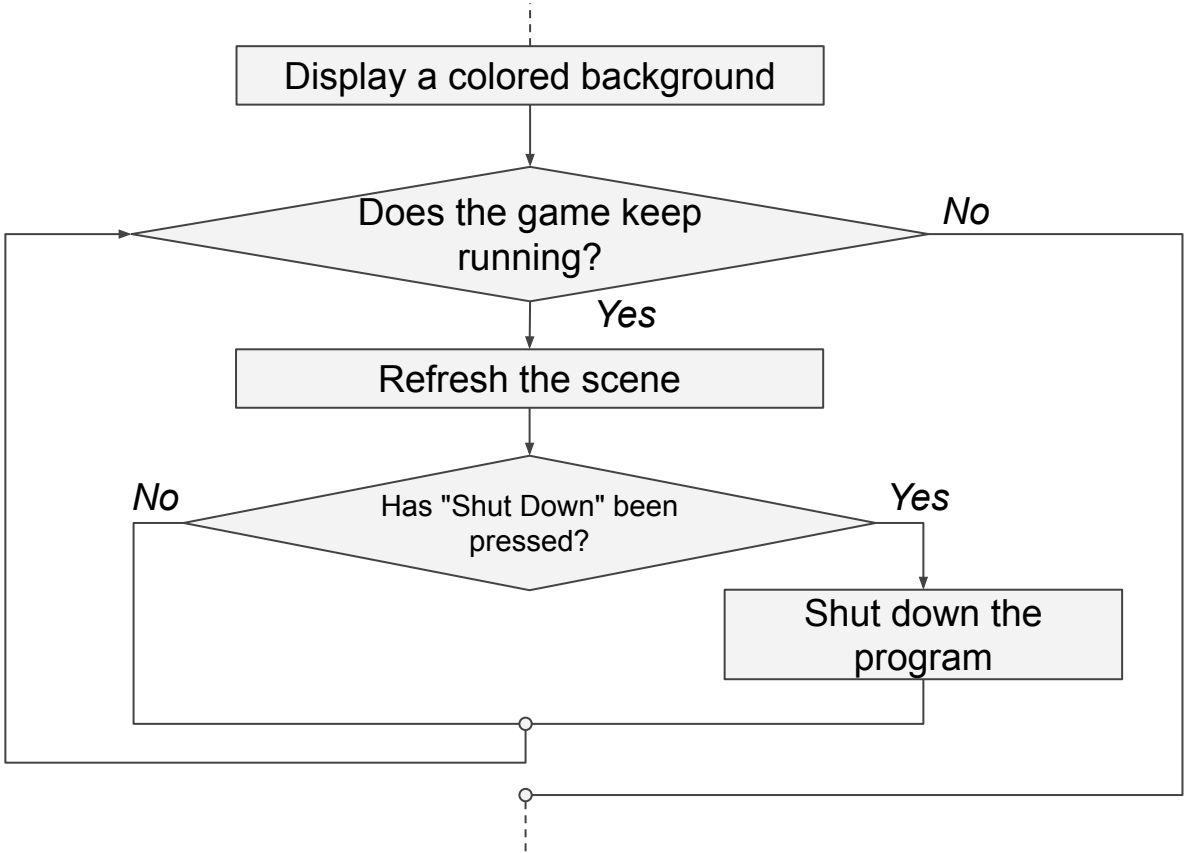
How do we create one?

What will be the condition for completing the preparation of the game template?

Checking qualifications

The flowchart for a game loop

The loop ends when you click on the "Shutdown" button



Checking qualifications

What is a class?

How can you create a derived class ?

Checking qualifications

Creating classes

To create a class, you need to:

- list the **properties** that define the features of a class instance in the constructor;
- list the **methods** for managing the instance.

```
class Class name():
    def __init__(self, Value):
        self.Property name = Value
    def Method name(self):
        Action with the object and properties
        Action with the object and properties
```

A special function of the **constructor** that creates an instance of a class with the specified properties.

__init__

Two lower underscores.

Checking qualifications

Superclass and derived class

Supposing the superclass has already been written, to create a derived class we then need to:

- specify the name of the superclass when creating a derived class;
- add the required methods to the derived class.

```
class Derived class name ( Superclass name ):
    def Method name (self):
        Action with the object and properties
```

*A variant where **new properties are not introduced**.*

*The derived class is only being supplemented with a **new method.***

Checking qualifications

What is a list?

How can you create a list and fill it with elements?

What other methods of working with lists do you know?

Checking qualifications

A list is a structure for storing different types of data in a well-ordered manner .

Example: A list of enemy sprites.

```
enemies = [sprite1, sprite2, sprite3, sprite4]
```

```
for e in enemies:
    e.draw()
```

→ Displaying each sprite from the enemies list.

Checking qualifications

Working with lists

Command	*Purpose*
`enemies = list()`	Declare an empty list
`enemies.append(sprite1)`	Add an item to the end of the list (it can be any type)
`enemy in enemies`	Search for the occurrence of an element in the list (returns True or False)

`for enemy in enemies:`
`    Command1`
`    Command2`
	Iterate through the `enemies` list items.
	
	“For each item (`enemy`) of the list (`enemies`), execute Command1, Command2”

`len(enemies)`	Determine the length of the `enemies` list

Checking qualifications

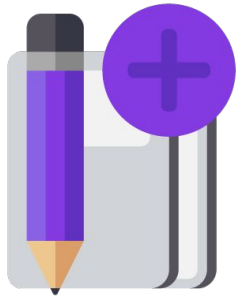
Qualifications confirmed!

Great, you are ready to brainstorm and complete your work task!

Checking qualifications

Brainstorm:

Game objects "Arkanoid"



Let's create a game template

The result will be a game scene with a background, ball and platform.

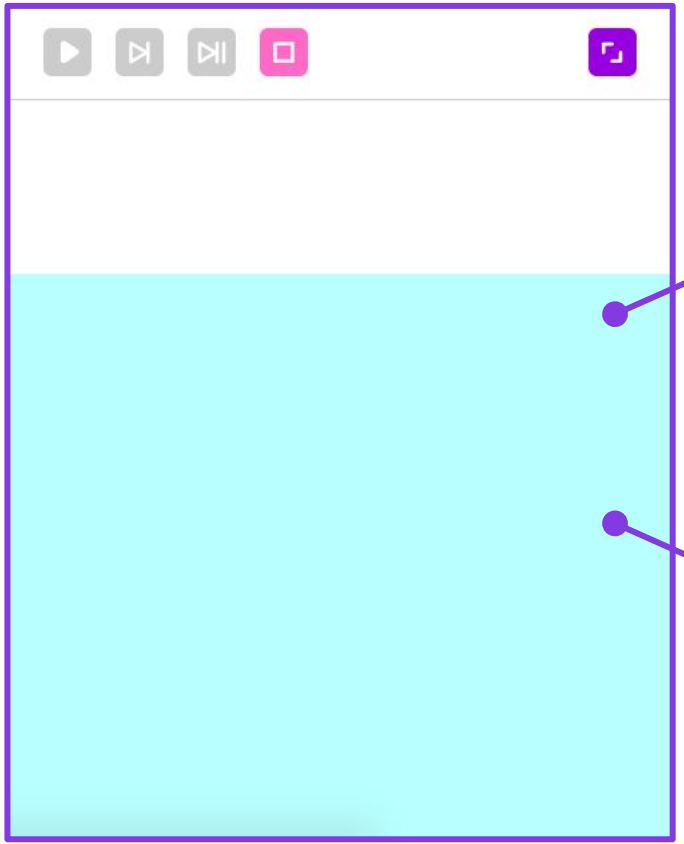
Brain storm

Checklist

- [ ] Create a game scene with dimensions of 500×500
- [ ] Set the scene's background color using the RGB palette
- [ ] Create game timer (pygame.time.Clock () object.)
- [ ] Create a game loop and set the frame rate to 40 frames/sec.
- [ ] Create a class for the image sprite.
- [ ] Create and display sprites: ball, platform and monsters.

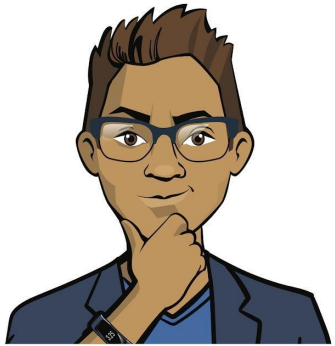
Let's create the game space and a class for working with pictures.

How do you program the template for the game "Arkanoid"?



Color blue (200, 255, 255)

Scene size 500x500



Brain storm

Program flowchart:

Result: a game scene filled with color.

Connect Pygame modules

Create background objects

Fill the scene with color

Create a game timer

Game loop:

Refresh rate ~40 FPS

Update the scene

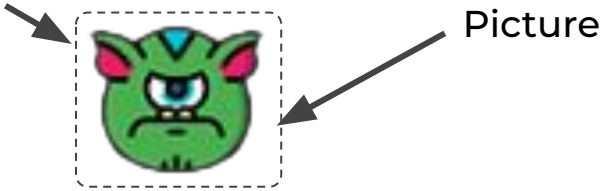
You have already implemented a similar template for the Fast Clicker game.

Brain storm

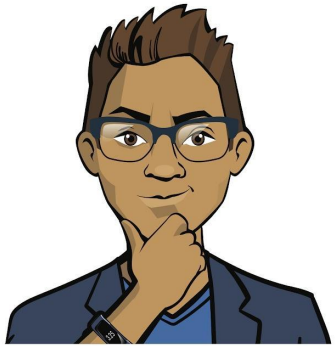
Creating sprites with pictures

According to the terms of reference, the appearance of the sprites should be defined by pictures:

Rectangular area



Maybe we can use the class we already made for the Fast Clicker project?



Brain storm

Creating sprites with pictures

According to the terms of reference, the appearance of the sprites should be defined by pictures:

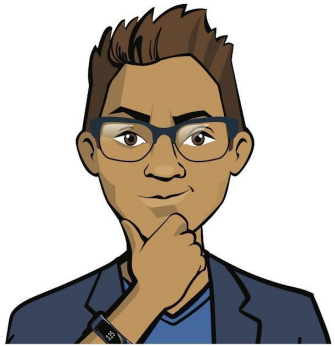
Rectangular area

Picture

Used to track object coordinates.

Responsible for external representation.

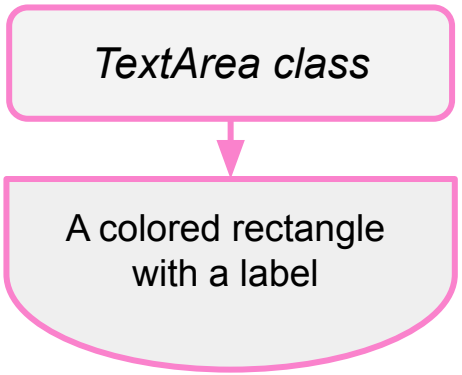
We have an Area class that sets an invisible frame for the sprite! How do we add its external appearance?



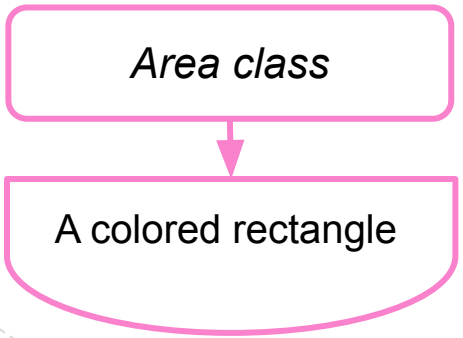
Brain storm

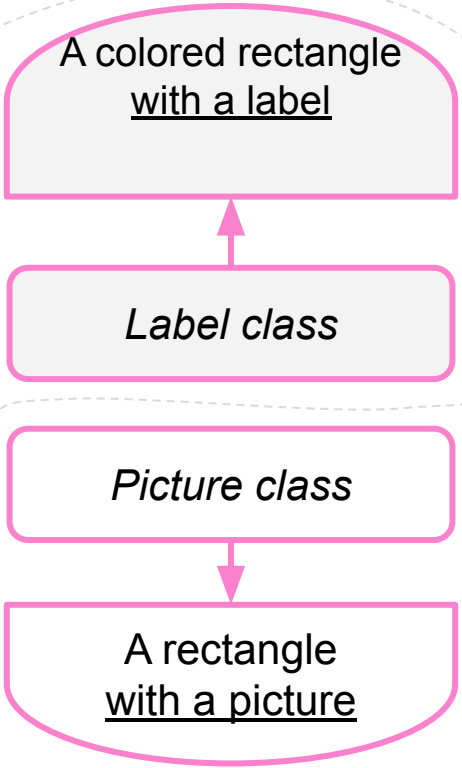
Creating sprites with pictures

Earlier we discussed a scheme with class inheritance:



It was implemented in the first project.

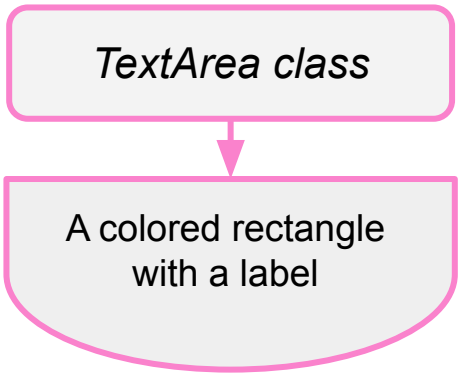




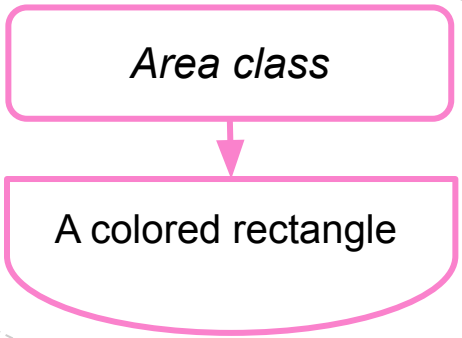
This solution was proposed for Fast Clicker.

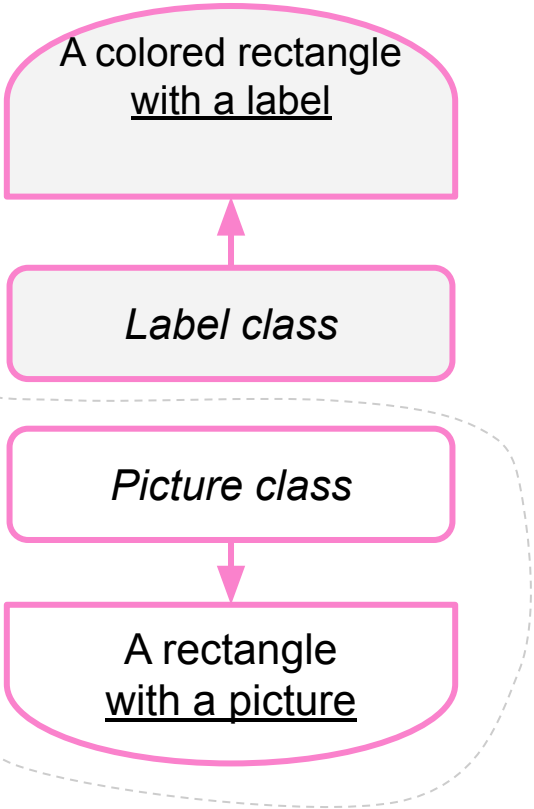
Creating sprites with pictures

Earlier we discussed a scheme with class inheritance:



It was implemented in the first project.



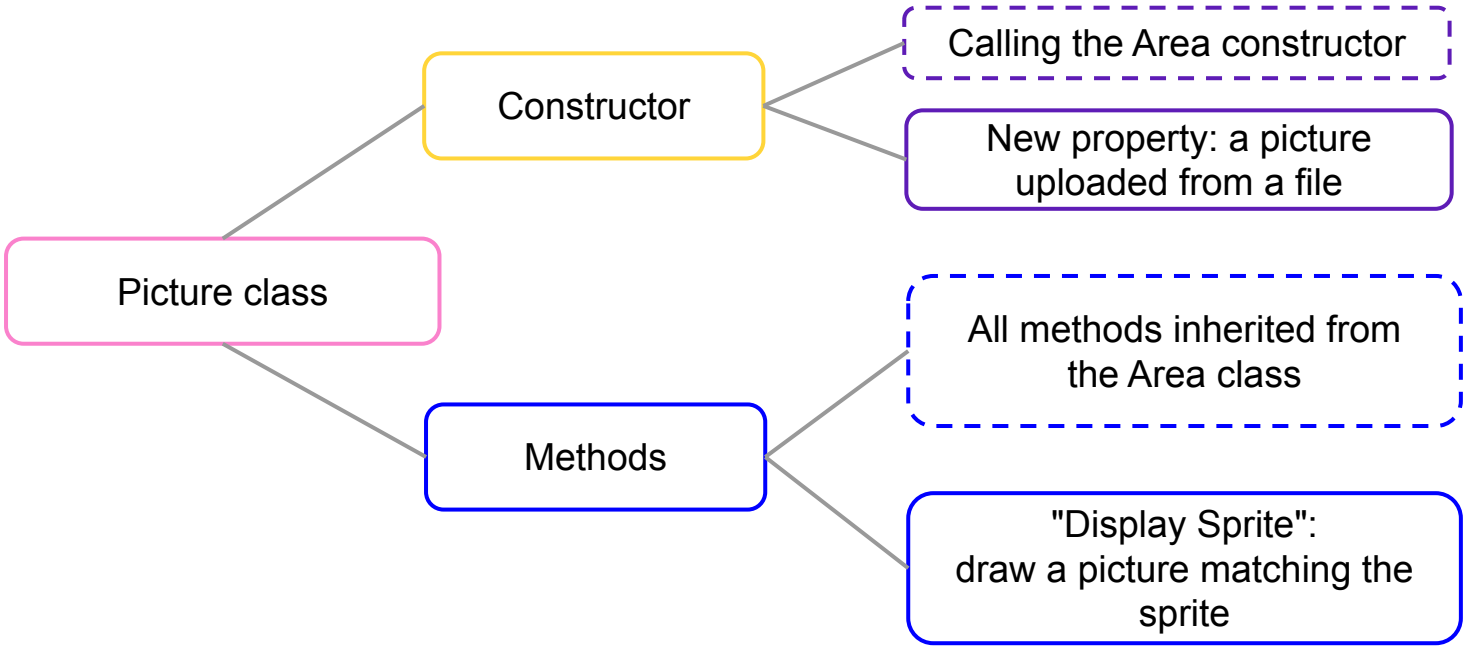


This hierarchy is suitable for "Arkanoid"!

Picture class

Area already describes a rectangular area. Let's copy this class from our previous project.

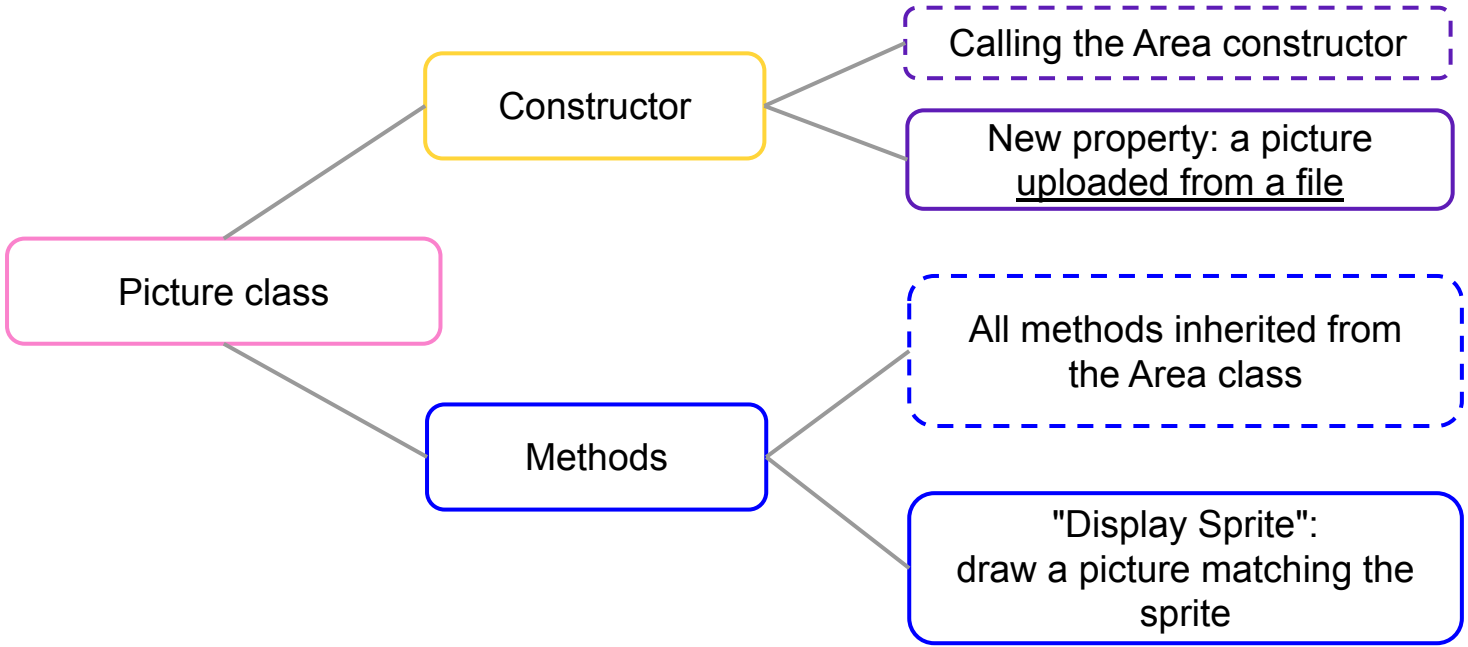
Let's create a derived Picture class that displays an image.



Picture class

Area already describes a rectangular area. Let's copy this class from our previous project.

Let's create a derived Picture class that displays an image.

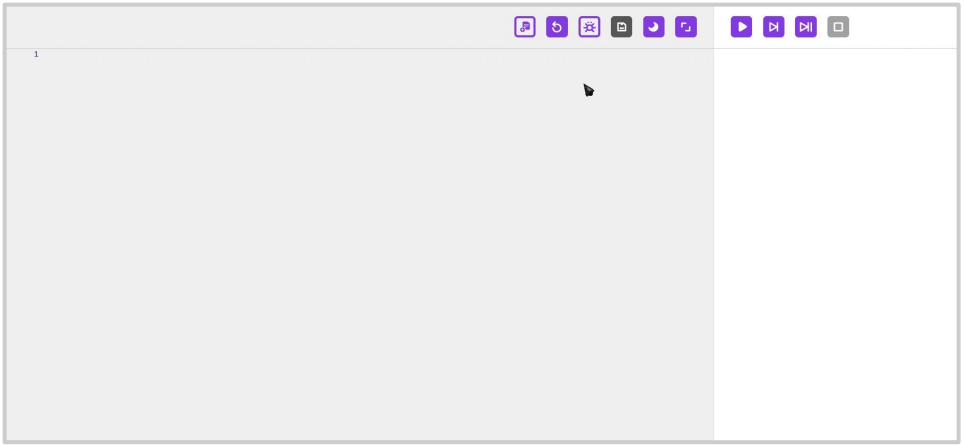


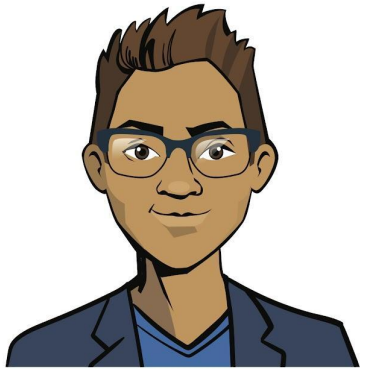
How do we get an image from a file to work with?

Working with graphic files

In Pygame, sprites can be defined not only by text and geometric objects, but also by pictures.

Command	*Purpose*
`image = pygame.image.load(filename)`	Uploading an image for work from a file named filename. The file must be stored in the same folder as the project.





Brain storm

Picture class

A rectangular area is already described in Area. Let's copy this class from our previous project.

Let's create a derived Picture class that displays an image.

```
class Picture(Area):
    def __init__(self, filename, x=0, y=0, width=10, height=10):
     super().__init__(x=x, y=y, width=width, height=height, color=None)
     self.image = pygame.image.load(filename)

   def draw(self):
     mw.blit(self.image, (self.rect.x, self.rect.y))
```

The images can already be found with the project on the platform.

Program flowchart:

The result is a game scene filled with color, with ball and platform sprites.

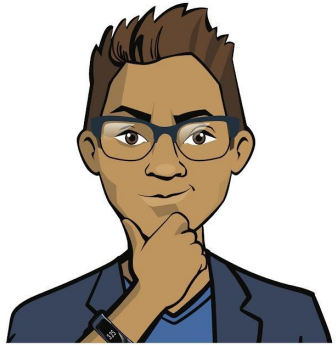
- Connect Pygame modules
- Set up a background and a timer
- *Create the **Area** class*
- *Create the **Picture** class*
- Set up the ball and platform
- Game loop:
  - Display the ball and the platform
  - Update the position of the ball and the platform
  - Update the scene

Brain storm

Your tasks:

1. Program the game template with a colored background.
1. Implement the Picture class and create two sprites: the ball and the platform.

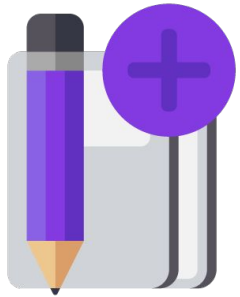
If there is time left over, think about how the monster sprites will be created and positioned.



Brain storm

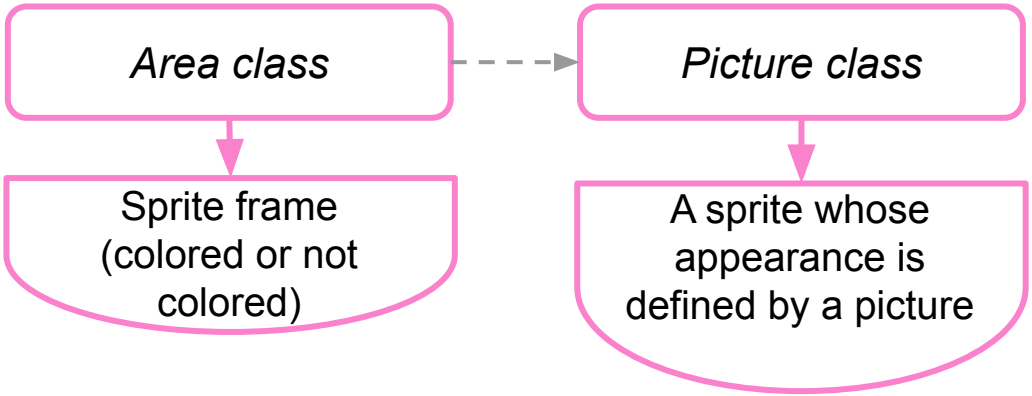
Brainstorm:

Monster objects

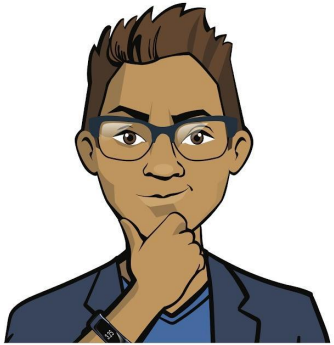


Create a set of cards

You already know how to create a sprite whose appearance will be determined by the picture.



But is everything really this simple when creating a large number of sprites? How do we create 24 monsters?



Brain storm

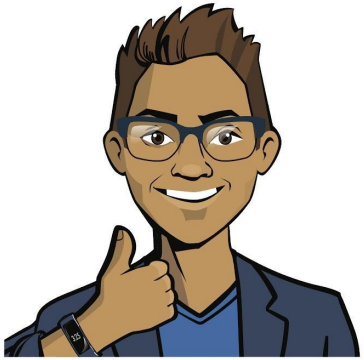
How do we create one monsters?

One monster is created in a similar way to the ball or the platform.

```
Enemy = Picture('enemy.png', x, y, 50, 50)
```

Coordinates of the point where the sprite appears.

The width and height of the frame which the sprite is written into.



The monster is also displayed similarly to other sprites.

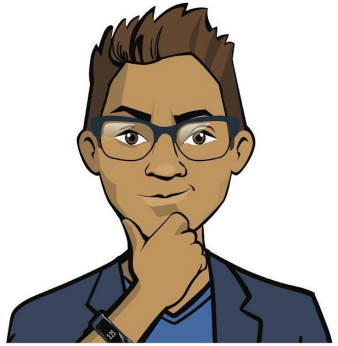
Brain storm

Working with a large number of sprites

But there are also some problems related to working with a large number of sprites at the same time:

- When creating sprites, a separate command is needed for each of the 24 objects that are the same type.
- When drawing them, they will need to apply the draw() method to each sprite one after another.
- An extremely long condition will be required to check for a collision with the ball.

Let's analyze one of the problems.



Brain storm

Working with a large number of sprites

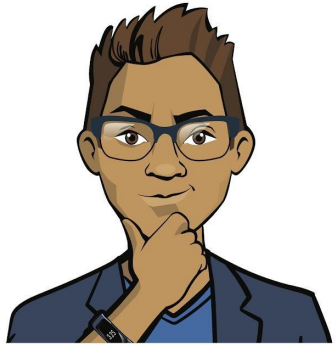
The problem with working with a large number of sprites:

➔ When drawing them, you need to apply draw() to each sprite one after another.

The game loop will be overloaded with the same type of commands. How can we optimize it?

Has the game not finished?

> Display sprite_1
> Display sprite_2
> Display sprite_3
> Display sprite_4
> ...
> Display sprite_24



Brain storm

Working with a large number of sprites

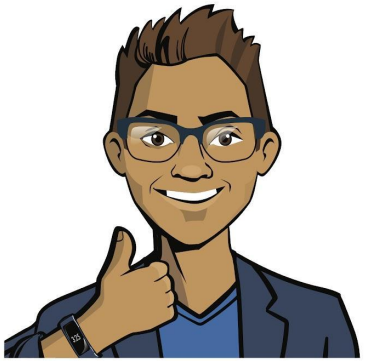
The problem with working with a large number of sprites:

➔ When drawing them, you need to apply draw() to each sprite one after another.

***By introducing lists**, we can greatly simplify the code:*

Has the game not finished?

> **For each sprite from the list:**
>
> > **Display sprite:**



Brain storm

Working with a large number of sprites

Other problems can also be solved with the help of lists:

Problem	*Solution*
When creating sprites, a separate command is needed for each of the 24 objects that are the same type.	Create and display a list of monsters in the loop.
When drawing them, they will need to apply the draw() method to each sprite one after another.	Draw all the sprites by iterating through the list elements in the "for" loop.
An extremely long condition will be required to check for a collision with the ball.	Perform a check inside the "for" loop that iterates through all the list elements.

Brain storm

Working with a large number of sprites

Other problems can also be solved with the help of lists:

Problem	*Solution*
When creating sprites, a separate command is needed for each of the 24 objects that are the same type.	Create and display a list of monsters in the loop.
When drawing them, they will need to apply the draw() method to each sprite one after another.	Draw all the sprites by iterating through the list elements in the "for" loop.
An extremely long condition will be required to check for a collision with the ball.	Perform a check inside the "for" loop that iterates through all the list elements.

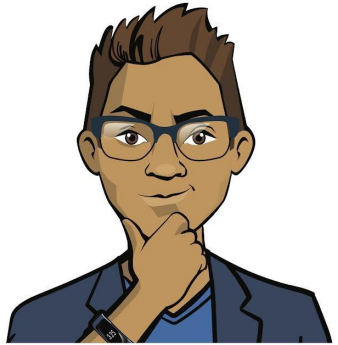
Brain storm

Let's take a closer look at creating sprites.

Creating and filling in a list of monsters

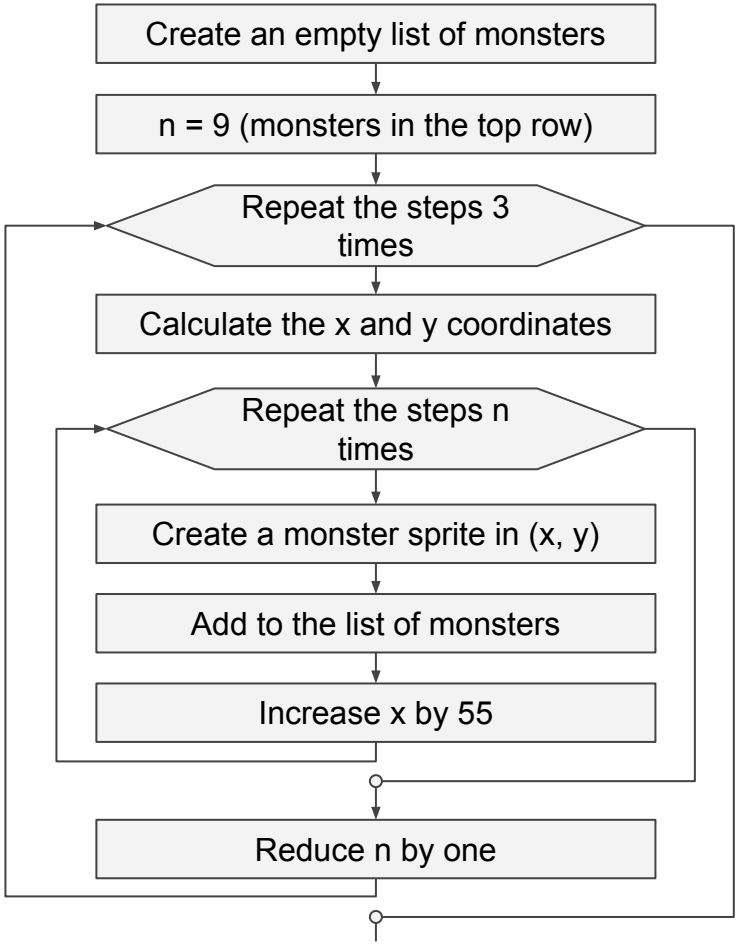
There should be a total of 24 monsters in the scene.

The sprites are ordered and placed at an equal distance from each other.

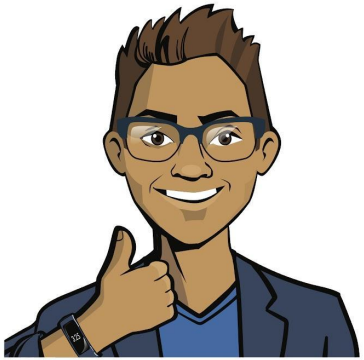


Brain storm

Creating and filling in a list of monsters

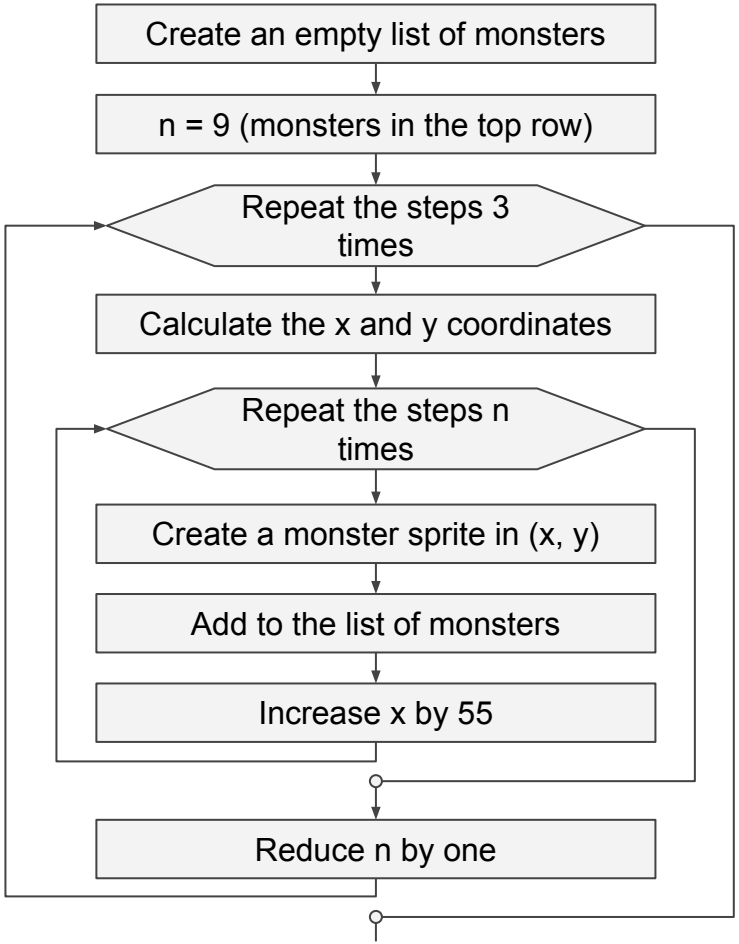


The algorithm can be like this!



Brain storm

Creating and filling in a list of monsters



By the number of rows of monsters.

By the number of monsters in one row.

If you know the initial x and y, you can calculate the point for the current monster. For example:

```
x = start_x + (27 * j)
```

Column number

Brain storm

Overall scheme for the program

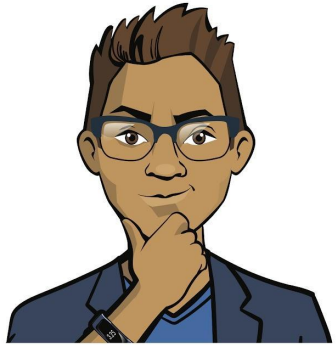
The main part of the program with the creation and display of the monsters:

- Creating an empty list of monsters
- Filling in the list of monsters
- Until the game is over...
  - Has the "ShutDown" button been pressed?
    - The game is over
  - Redraw the ball and the platform
  - Draw a set of monsters

Brain storm

Your tasks:

1. Create a list of monsters and fill it with monster sprites.
2. Display a set of sprites in the scene.



Brain storm