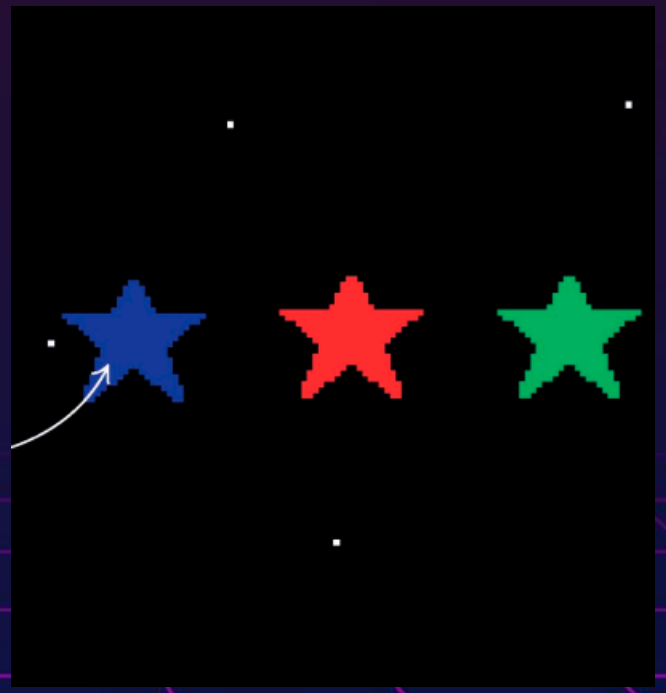
5.2
Red Alert Project

Part 2 of 3

5.2 Red Alert Project

- Last lesson we have set up the basic structure for our project.
- We have learned how to use **tuples** and **dictionaries** in Python.
- Our mission in this lesson is to complete the empty user defined functions that create and layout our star **Actors** on screen.



Flowchart to Make Stars

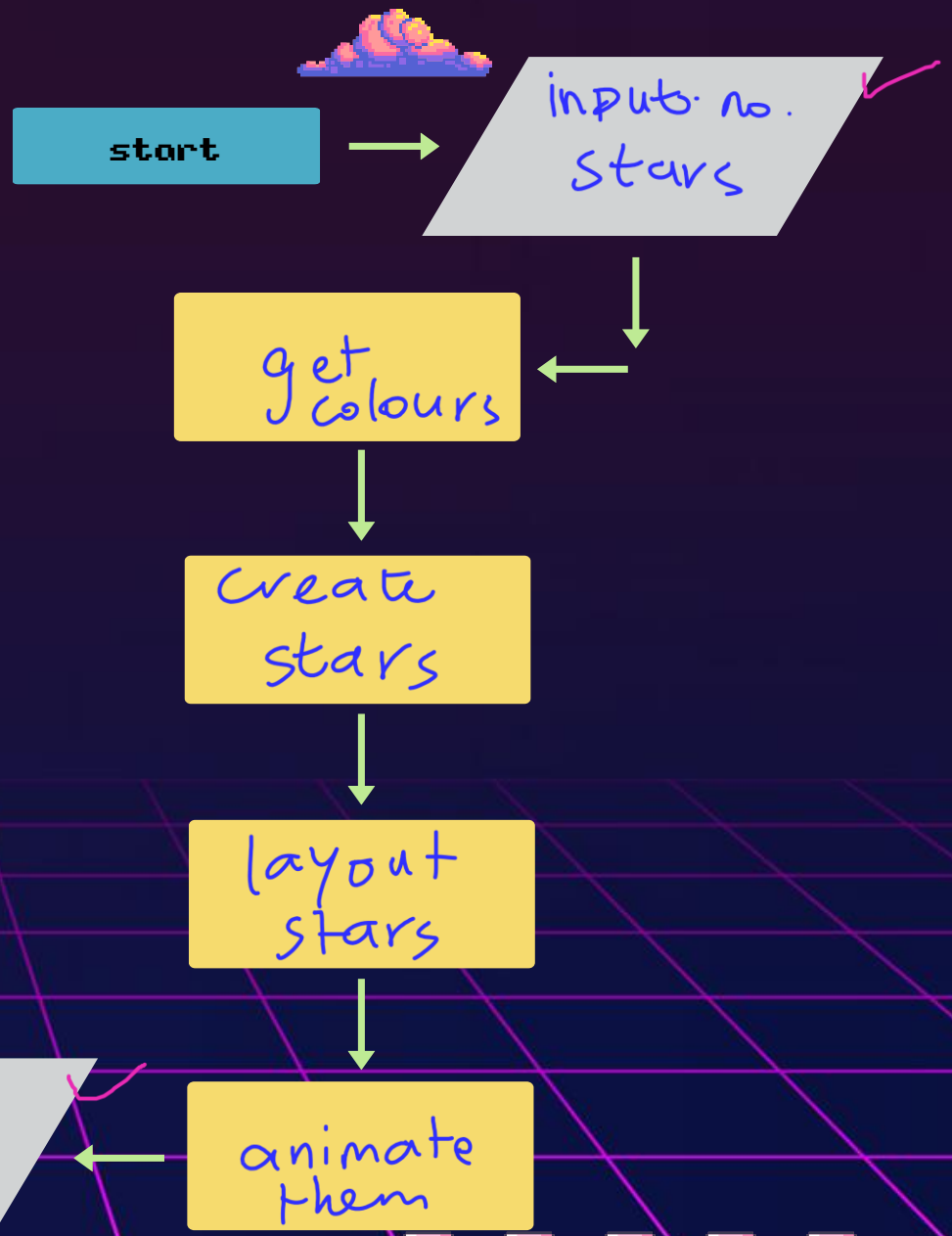
- get colors
- ~~input no. stars~~
- animate stars
- layout stars
- create stars
- ~~output stars~~

```
def make_stars(number_of_extra_stars):
    colors_to_create = get_colors_to_create(number_of_extra_stars)
    new_stars = create_stars(colors_to_create)
    layout_stars(new_stars)
    animate_stars(new_stars)
    return new_stars
```

In

*

output



We will code as we learn

- This is a big project, so we are going to code bits as we learn.
- First, we should understand the tasks that are planned.
- We will implement all the TODOs in the code to revise the simpler concepts we covered and learn new things!

Understand the sample

First to complete is a _ user _-defined function named **get_colors_to_create()**.

As input, it accepts a _ parameter _ that indicates how many stars are green or blue.

As output __, it **returns** a _ list _ of string colors one of which is red.

Get Colors

return value

3. ???

1. ???

```
def get_colors_to_create(number_of_extra_stars):

    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

    for i in range(0, number_of_extra_stars):

        _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

        colors_to_create.append(random_color)

    return colors_to_create
```

2. ???

function def

Oh no, why are the labels out of place? Help!

Looks like some code is missing as well. What shall we do?

parameter

Fill in the Blanks

We will code as we learn

- Time to put our skills to the test.
- Find the relevant TODO exercises and fill in the missing code.
- Taking on a challenge can be fun and feel less difficult in this way.

Understand the sample

Next up, we need to create stars **for every** _ color in **colors_to_create**.

We will use a _ **loop** _ with the _ **in** _ operator.

We must be careful to _ **indent** _ our code properly, otherwise it will not work.

Create Stars

The lines of code are jumbled up. Can you pick the right order?

```
1. def create_stars(colors_to_create):
2.     star = Actor(color + "_star")
3.      for color in colors_to_create:
4.         new_stars = []
5.         return new_stars
6.     new_stars.append(star)
```

A. 1, 4, 3, 2, 6, 5

B. 1, 4, 3, 6, 2, 5

C. 6, 5, 4, 3, 2, 1

Multiple Choice

We will code as we learn

- Time to put our skills to the test.
- Find the relevant TODO exercises and fill in the missing code.
- We have written enough features to test our progress at this point.
- Taking on a challenge can be fun and feel less difficult in this way.

Understand the sample

To place the stars evenly on the screen we will write some **arithmetic** expressions.

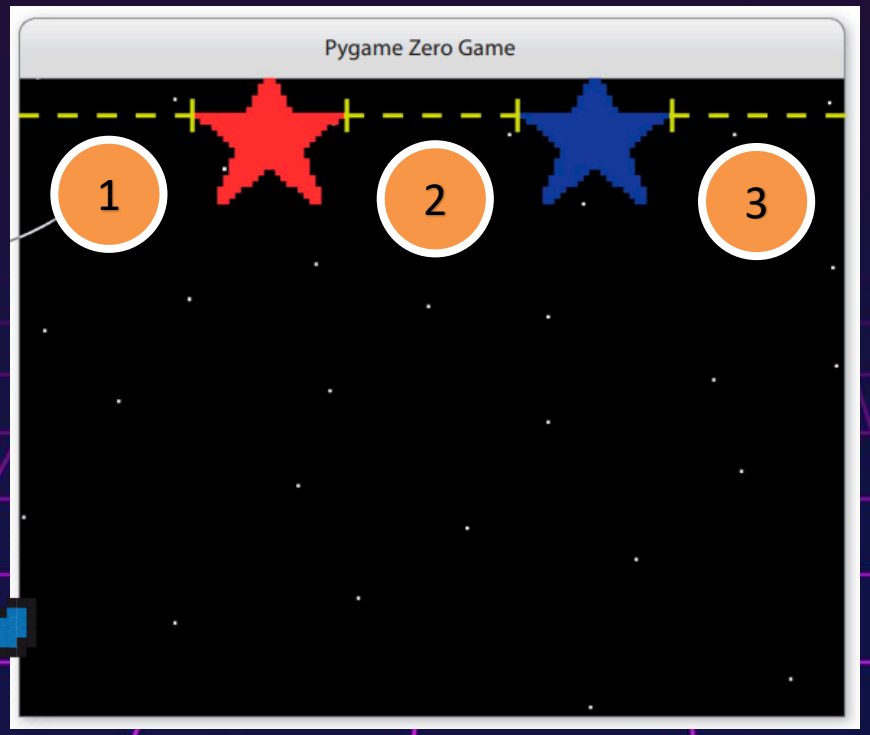
These expressions will combine _**variables**_ with **operators** like **division**_.

After a bit of math, we will get a new **position**_ to update our star.

Layout Stars

Multiple Choice

For two stars, we need three gaps. How can we calculate gaps for stars?



A. Gaps = Stars − 1

B. Gaps = Stars

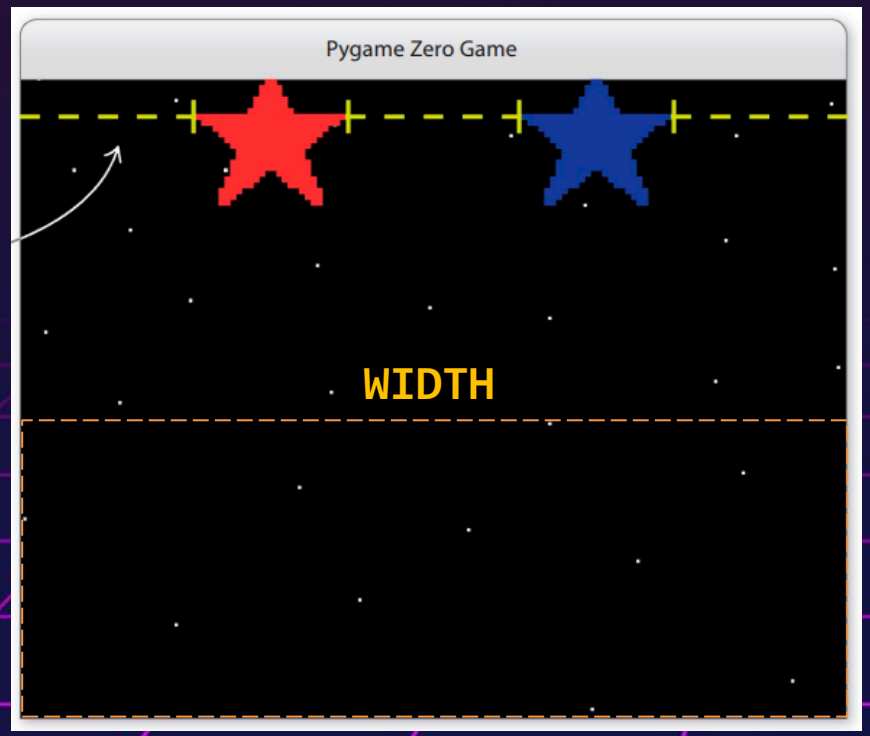
C. Gaps = Stars + 1

Drag some stars and try to lay them out neatly on a line.

Layout Stars

Multiple Choice

Given the number of gaps, how wide would each gap be?



A. WIDTH // No. of Gaps

B. WIDTH / No. of Gaps

C. No. of Gaps / WIDTH

Layout Stars

Make the right function call to randomly shuffle our stars before laying them out on screen.

function call

```
>>> random.shuffle(stars_to_layout)
```

We do this so that the red star changes position at every level.

Fill in the Blanks

Function Docs

- item random.choice(list)

  Gives a random item from a list.

- random.shuffle(list)

  Changes the order of items in the list itself.

- list random.sample(list, n)

  Gives random n items from the list.

We will code as we learn

- Time to put our skills to the test.
- Find the relevant TODO exercises and fill in the missing code.
- We have written enough features to test our progress at this point.
- Taking on a challenge can be fun and feel less difficult in this way.

Understand the sample

The stars at the top need to be **animated** to slowly reach the bottom.

Like our stars, their animations are stored in a **list**.

Animation is complex so we'll use a **function** from the **pgzrun** module.

Animate Stars

- The animate() function is provided by Pygame Zero and makes it easy to move objects on the screen.
- We simply need to understand the parameters the function takes, there are a few.
- You could probably guess what each parameter is for in the activity on the next slide.

```
animation = animate(star,
                    duration=duration,
                    on_finished = on_game_over
                    y=HEIGHT)
```

Animate Stars

Match the parameters with the correct description.

1	`star`	The element being displayed on screen that needs to be animated.
2	`duration`	The number of seconds the animation should last.
3	`on_finished`	What the program should do next when the animation finished.
4	`y`	Setting the size of the animation for a particular direction.

We will code as we learn

- Time to put our skills to the test.
- Find the relevant TODO exercises and fill in the missing code.
- We have written enough features to test our progress at this point.
- Taking on a challenge can be fun and feel less difficult in this way.

Are you ready to take over?

- There is one more TODO exercise for you to complete.
- Do you think you are ready to complete the game with less guidance?
- Find the last exercise and complete it with courage.
- Do not forget to **test your code** for **syntax errors**.

You have leveled up!

Way to go for getting this far!

Stars for everyone.

What comes next?

We will complete the code that handles the player’s interactions with the screen and control the game variables.