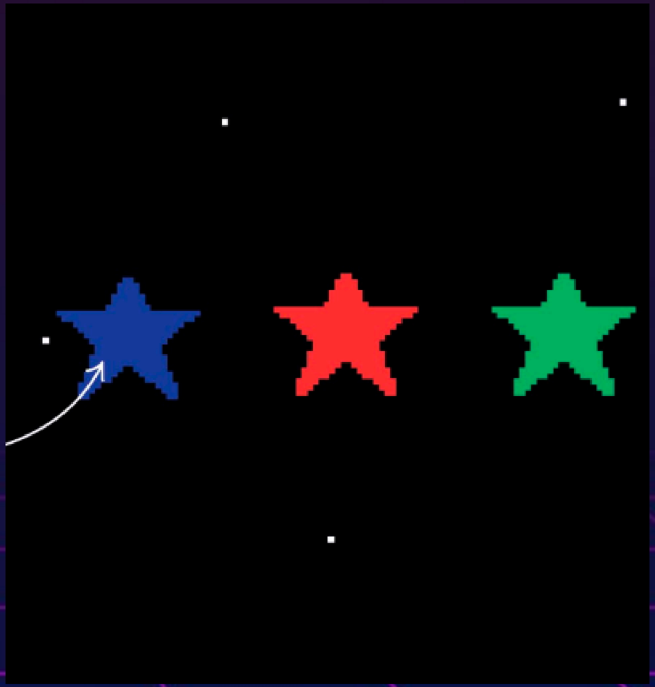
5.1
Red Alert Project

Part 1 of 3

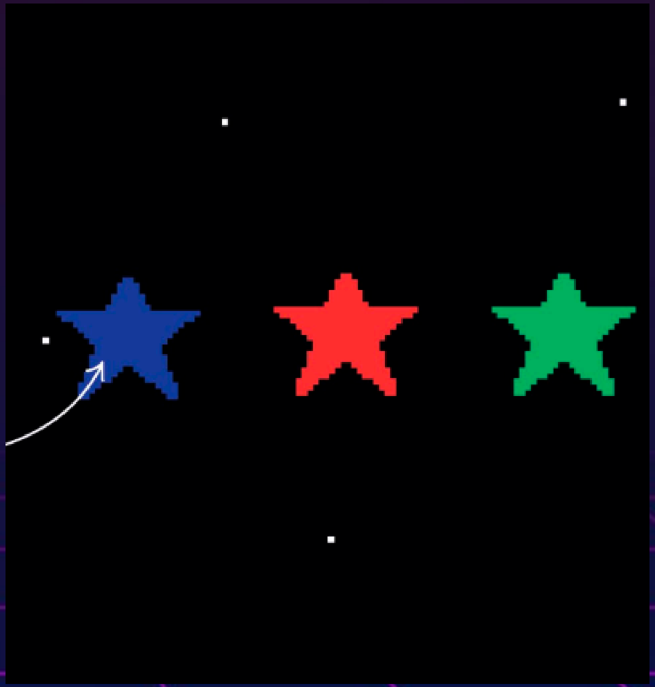
5.1 Red Alert Project

- As multicolored stars fall down the screen, players must click on the **red star** to earn points and keep the game moving. When the player hits a star that is **not red**, the game is over.
- You will build a program with a **graphical interface** using the PyGame Zero **module**.
- You will level up by using **assignment operators** to update the score.
- You will also use a **dictionary** to store fun messages that get displayed at various times during game play.



What happens

- At the start, two stars appear and start moving down the screen.
- If the player clicks on the red star before any of the stars reach the bottom of the screen, then points are earned, and the game moves on to the next level.
- With each level, more green and blue stars are added, and they move faster than before!
- If the player clicks on any star other than the red one, or if the stars reach the bottom of the screen, the game ends.



Flowchart

~~draw/align stars~~

display message

~~add stars and speed~~

~~red star?~~ *

~~is star at bottom?~~ *

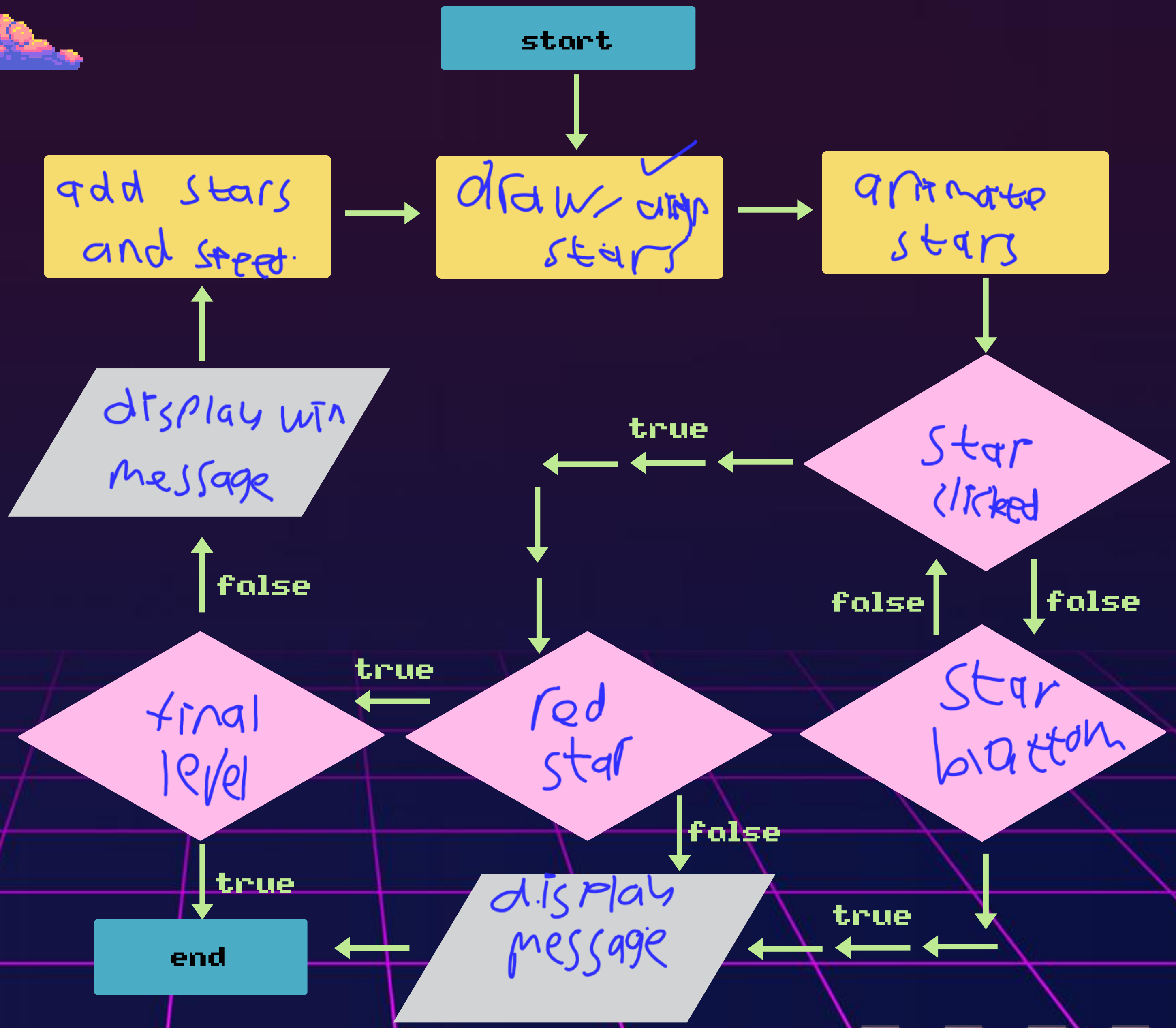
~~display message~~

~~star clicked?~~ *

~~animate stars~~

~~display win message~~ *

~~final level?~~ *



We will code as we learn

- This is a big project, so we are going to code bits as we learn.
- First, we should set up and create our project files.
- Have a look at the sample code, give it a whirl.
- We will implement all the TODOs in the code to revise the simpler concepts we covered and learn new things!

Understand the sample

We need to import modules like _random_ and _pgzrun_ for this game.

These give us useful _functions_ to randomize star colors and animations.

We will be learning about new structures: _tuples_ and **dictionaries**.

Centre of the screen

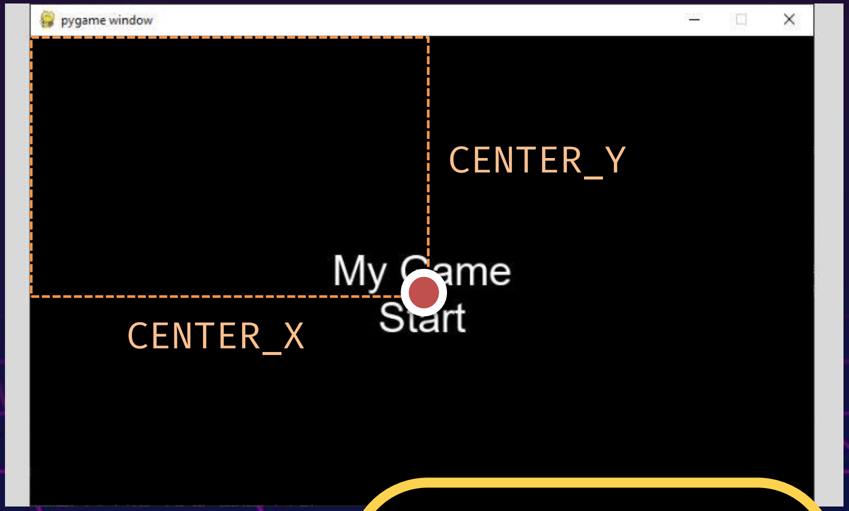
tuple

```
CENTER = (CENTER_X, CENTER_Y)
```

can store multiple items in a single variable

How is this different to a list then?

In most ways it is similar except that **items cannot change**!



screen

Let us practice

Complete the snippet

Complete the code snippet that creates a variable that stores a student record consisting of a name, date of birth, and ID number. Which way do you prefer?

```
# the old way with 3 variables
student_name = “John”
student_dob = ____________
____________ = “123410M”
# new way using a tuple
student = _____ “John”, ____________ “123410M”)
```

student_id

“12/08/2010”,

“12/08/2010”

(

Fill in the Blanks

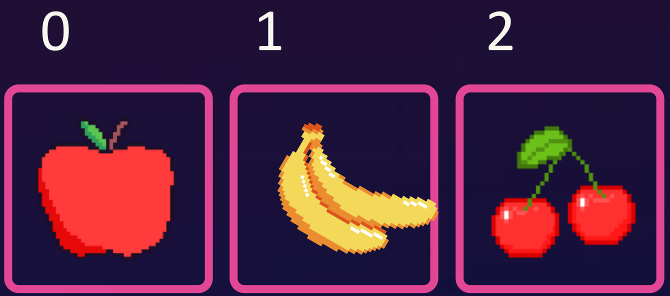
Tuples and lists have this in common

```
fruits_list = ["apple", "banana", "cherry"]
fruits_tuple = ("apple", "banana", "cherry")
first_fruit_list = fruits_list[0]
first_fruit_tuple = fruits_tuple[0]
```

index

Short Answer

How can you get the cherry from the tuple?



Like lists, each item in a tuple has a unique position starting from zero.

We will code as we learn

- Time to put what we learned into practice.
- Find the relevant TODO exercises and fill in the code.
- Taking on a challenge can be fun and feel less difficult in this way.

Our game messages

dictionary

```
messages = {
     "lose": "Try again.",
     "win": "Great job!"
}
```

store items in the form of key, value pairs.

Dictionaries are powerful tools.

Use **curly brackets** and **keys** and **values** separated by a **comma**.

We would like to have a variety of game messages displayed at many stages of the game to make it feel more interactive.

Having all our game messages in one place is convenient for us to make changes to the messages.

Easier than lists!

key

```
if game_over == True:
   lose_message = messages["lose"]
   display_message("GAME OVER!", lose_message)
elif game_complete == True:
    # display win message → Win
else:
    # player continues → Normal Level
```

The key acts as a unique index to the value we need. Much easier to remember than a number.

Let us practice

Complete the snippet

Convert our magic-key - chest matrix into a dictionary that stores key-value pairs.

brand	Ford
model	Mustang
year	2024

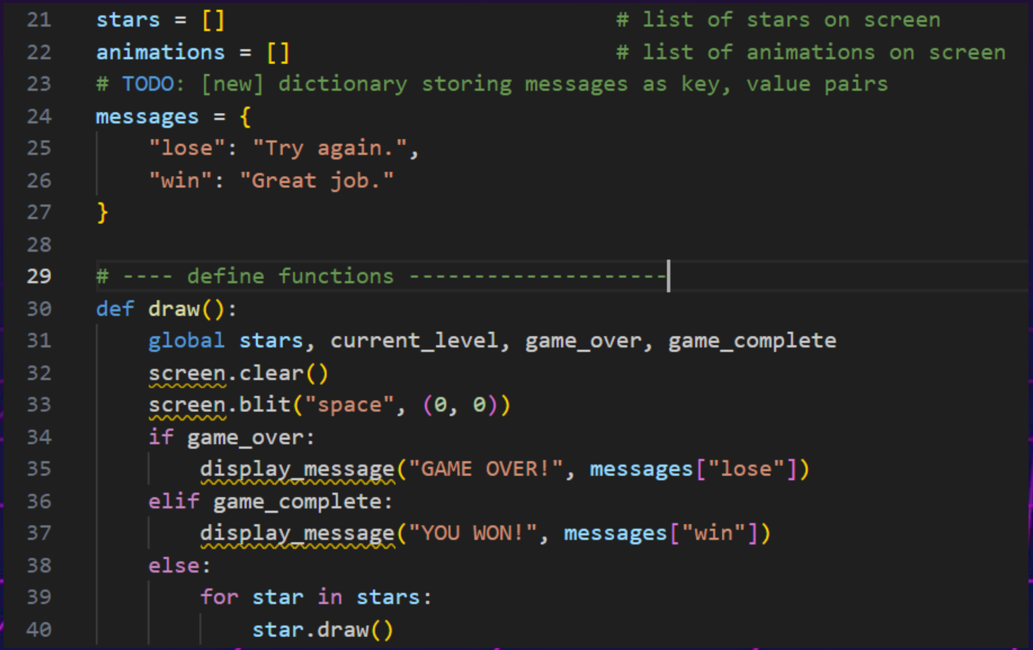
Fill in the Blanks

```
my_dream_car = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 2024
}
```

Do not forget the details like **curly brackets** and **quotes**.

Understand the code sample

Where in the code sample is the dictionary being created and used?



```
21  stars = []                                # list of stars on screen
22  animations = []                           # list of animations on screen
23  # TODO: [new] dictionary storing messages as key, value pairs
24  messages = {
25      "lose": "Try again.",
26      "win": "Great job."
27  }
28
29  # ---- define functions --------------------
30  def draw():
31      global stars, current_level, game_over, game_complete
32      screen.clear()
33      screen.blit("space", (0, 0))
34      if game_over:
35          display_message("GAME OVER!", messages["lose"])
36      elif game_complete:
37          display_message("YOU WON!", messages["win"])
38      else:
39          for star in stars:
40              star.draw()
```

A. Lines 21, 38, 40

B. Lines 24, 32, 38

C. Lines 24, 35, 37

Multiple Choice

More practice

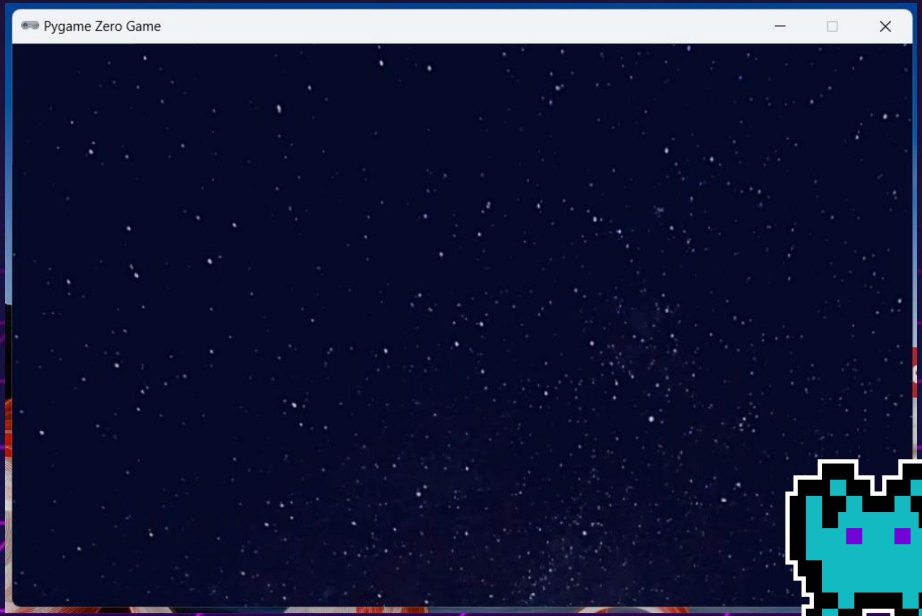
Short Answer

```
if game_over == True:
   lose_message = messages[“lose”]
   display_message(“GAME OVER!”, lose_message)
elif game_complete == True:
    # get the win message
    display_message(“YOU WIN!”, win_message)
else:
    # player continues
```

Write the line of code that gets the win message from the dictionary are stores it in a variable.
Look at the snippet for a hint.

We will code as we learn

- Complete the remaining TODO exercises and **test your code**.



You have leveled up!

Way to go for getting this far!

Stars for everyone.

What comes next?

Several _______________ are defined, two of which are completed for us.

The completed ones _______________ and draw _______________ e.g., stars.

The `make_stars()` function _______________ the remaining functions which are empty.

functions

calls

make

elements

What comes next?

We will complete the code in the empty functions that all contribute to create the star elements that are displayed on screen.