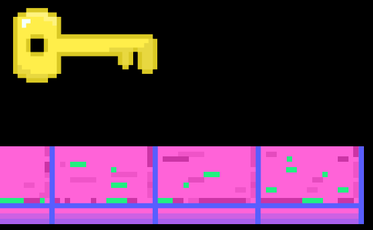

GAME START
PLAY
MENU
EXIT

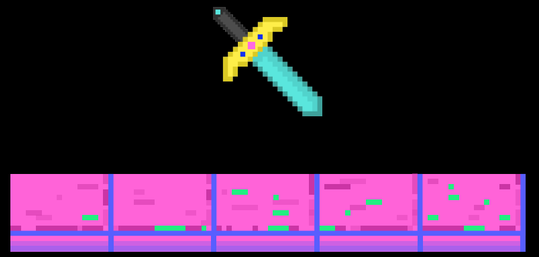
MENU 01 07 12

AGENDA

TOPICS COVERED



INTRO TO OBJECT-
ORIENTED PROGRAMMING



MAKE SPACE INVADERS
WITH GAMEMAKER STUDIO 2

SIGN IN

BACK TO AGENDA PAGE

OBJECT ORIENTED PROGRAMMING

MENU

WHAT IS IT?

- Programming Paradigm, a way to code
- We have classes (things) with attributes (characteristics) attached to it
- A Person class has attributes Name and DOB

BACK TO AGENDA PAGE

MENU

FEATURES

- Classes and Objects
- Inheritance
- Polymorphism
- and some more...

BACK TO AGENDA PAGE

MENU

APPLICATIONS & COMMON LANGUAGES

- Software Dev
- Game Dev

- Python
- Java
- C++
- Ruby

BACK TO AGENDA PAGE

MENU

“ChatGPT, tell me in one sentence why programmers should use OOP.”

“Programmers should use Object-Oriented Programming (OOP) to better organize and structure code by encapsulating data and behavior into reusable and modular objects, leading to more maintainable and scalable software systems.”

BACK TO AGENDA PAGE

MENU

WHAT?

- Create structures
- Write reusable code
- Promote inheritance, polymorphism, abstraction

BACK TO AGENDA PAGE

SOUNDS TOO FANCY...

But how does it actually work?

SIGN IN

BACK TO AGENDA PAGE

CLASSES AND OBJECTS

MENU

CLASS

A class is like a template for creating things.

For example, if you had a game:

- 1 player has a name and a superpower (2 attributes)
- Then 5 players have 10 attributes altogether!

BACK TO AGENDA PAGE

MENU

TAKE A LOOK AT THE DEFAULT STRATEGY

BACK TO AGENDA PAGE

```
# Define their names
player1_name = "Eden"
player2_name = "Rama"
player3_name = "Kasie"
player4_name = "Kevin"
player5_name = "Prat"

# And their powers...
player1_power = "Gravity Manipulation"
player2_power = "Force Fields"
player3_power = "Super Speed"
player4_power = "Elemental control"
player5_power = "Flight"

print(player1_name + " has superpower: " + player1_power)
print(player2_name + " has superpower: " + player2_power)
print(player3_name + " has superpower: " + player3_power)
print(player4_name + " has superpower: " + player4_power)
print(player5_name + " has superpower: " + player5_power)
```

MENU

ADVANTAGES

- Easy to write
- Simple logic

DISADVANTAGES

- Lack of abstraction, hard for larger databases.
- Code duplication
- Limited reusability (can’t modify or extend)
- How do you know which player has which superpowers?

BACK TO AGENDA PAGE

MENU

SOLUTION? DEFINE A CLASS!

```
class Player:
  def __init__(self, name, power):
    self.name = name
    self.power = power
```

BACK TO AGENDA PAGE

MENU

CREAT AN OBJECT

An object is an instance of a class

```
player1 = Player("Eden", "Gravity Manipulation")
player2 = Player("Rama", "Force Fields")
player3 = Player("Kasie", "Super Speed")
```

BACK TO AGENDA PAGE

MENU

1 STEP BACK

Class name

```
class Player:
    def __init__(self, name, power):
        self.name = name
        self.power = power
```

Define a method, a function that the object can use

Referencing attributes, the variables inside the class

BACK TO AGENDA PAGE

MENU

WHAT IS A METHOD?

A method is a function that can be called by an object to perform a task.

Example:

- The __init__ function (automatic upon creation)
- findDistance, output

How is it different to a normal function?

BACK TO AGENDA PAGE

MENU

HOW IS THAT DIFFERENT FROM A FUNCTION???

A method is linked with a class, and the class is `implicitly` passed to the method

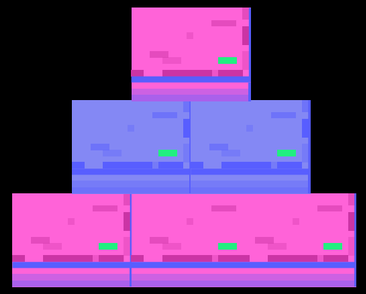
A method can operate on data within the class

BACK TO AGENDA PAGE

MENU

NOW...

Say I want to print a sentence describing a class

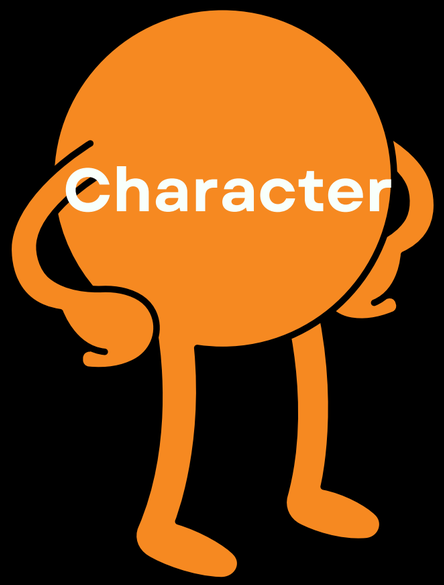


BACK TO AGENDA PAGE

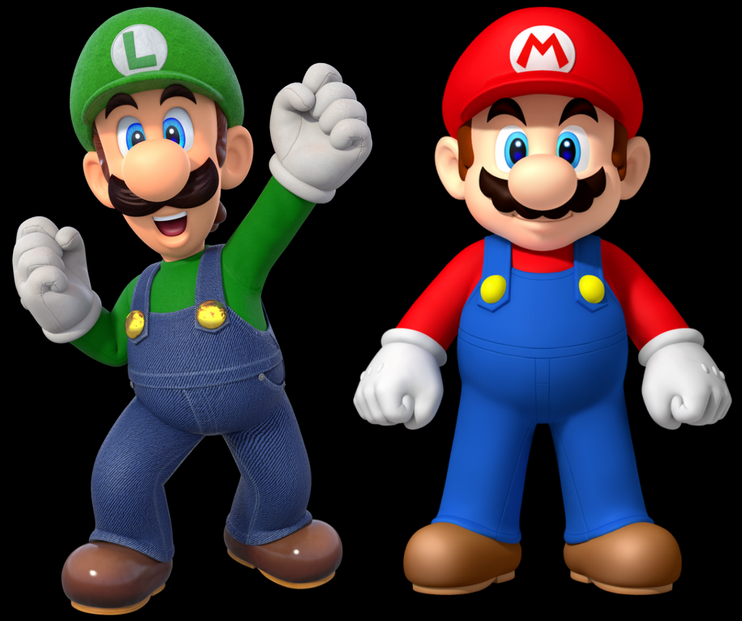
```
class Player:
  def __init__(self, name, power):
    self.name = name
    self.power = power

  def describe(self):
    print(self.name + " has superpower: " + self.power)


# Create your objects, instances of your class
player1 = Player("Eden", "Gravity Manipulation")
player2 = Player("Rama", "Force Fields")
player3 = Player("Kasie", "Super Speed")


player1.describe()
player2.describe()
player3.describe()
```

MENU

CLASS



OBJECT



ATTRIBUTE

Name

Speed

Special ability

Vehicle preference

BACK TO AGENDA PAGE

MENU

What should be a class?

YES

- Person
- Furniture
- ChessPiece
- Device

NO

- Name
- Height
- Knight
- Laptop

BACK TO AGENDA PAGE

SIGN IN

BACK TO AGENDA PAGE

INHERITANCE

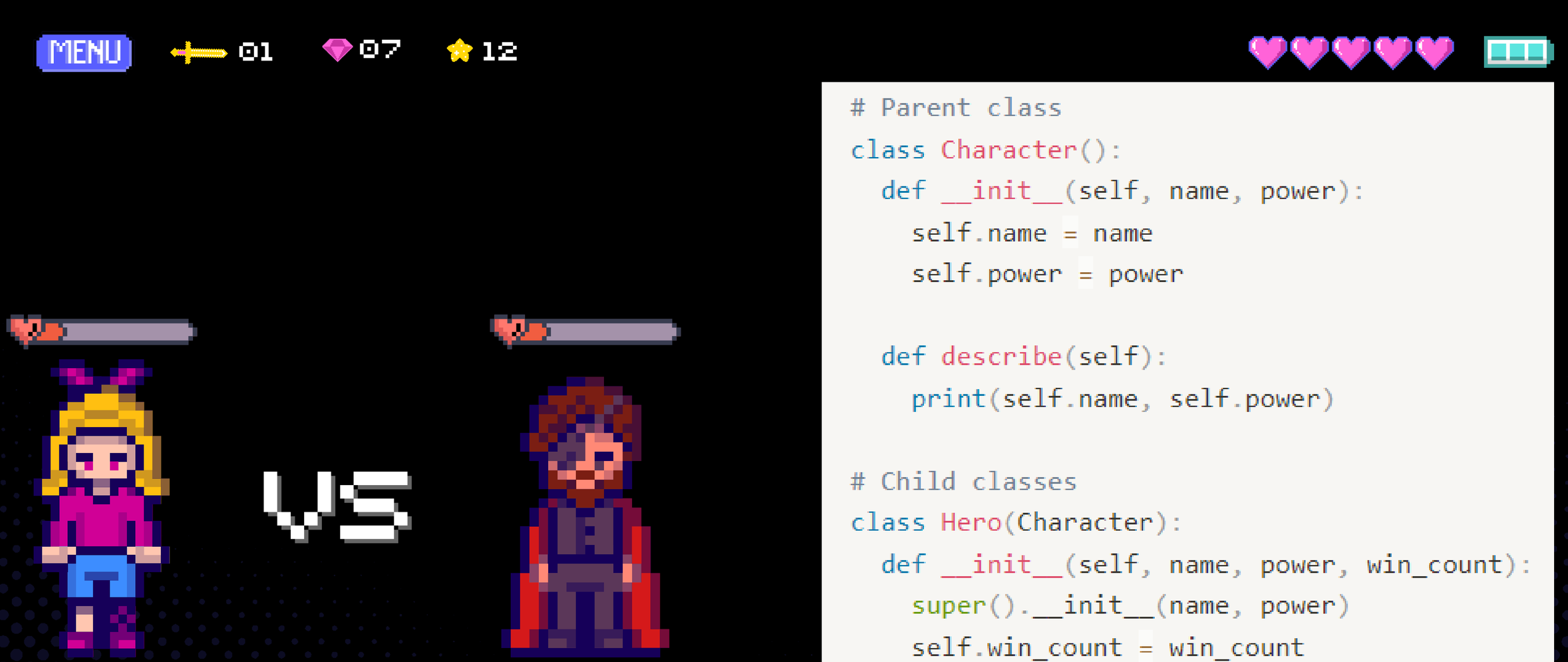
WHAT IS INHERITANCE?

- Inheritance is when you have one class that inherits some or all of its properties from another class. We say the child class inherits from the parent class.

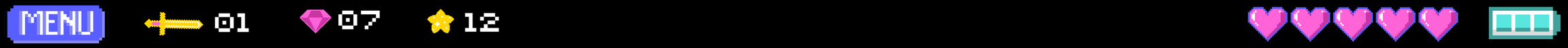
- For example, there are multiple characters in a game, some are heroes, and others are villains. So we define a parent class as follows:


```python
class Character():
  def __init__(self, name, power):
    self.name = name
    self.power = power

  def describe(self):
    print(self.name, self.power)
```

And then....




```python
# Parent class
class Character():
  def __init__(self, name, power):
    self.name = name
    self.power = power

  def describe(self):
    print(self.name, self.power)

# Child classes
class Hero(Character):
  def __init__(self, name, power, win_count):
    super().__init__(name, power)
    self.win_count = win_count

class Villain(Character):
  def __init__(self, name, power, story):
    super().__init__(name, power)
    self.story = story
```



Now we can create objects just like we did earlier:

```
spiderman = Hero("Spiderman", "Spider Powers", 10)
DocOctopus = Villain("Doctor Octopus", "Electronic Arms", "Originally a brilliant sc
ientist, his greatest invention, a set of metallic limbs, became fused to his body b
y an accident which caused his insanity. He has telepathic control of these arms, wh
ich are strong enough to physically hurt Spider-Man.[12] While Doctor Octopus is reg
arded as one of Spider-Man's archenemies, he also been portrayed as an antihero, and
even starred in his own comic book storyline that saw him becoming a superhero calle
d the Superior Spider-Man after the original Spider-Man's death.")
```

SIGN IN

BACK TO AGENDA PAGE

POLYMORPHISM

An overview

MENU

"MANY FORMS"

It is like using the same function for different purposes

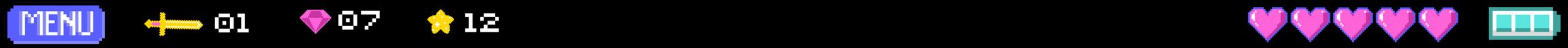
```
# len() function
print(len("Hello World!"))
print(len([1, 2, 3, 4, 5]))
print(len("Apples", "Bananas"))
print(len({"num1": 1, "num2": 2, "num3": 3}))
```

BACK TO AGENDA PAGE

SIGN IN

BACK TO AGENDA PAGE

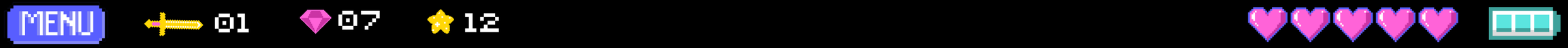
LAST SECTION I PROMISE



EVENTS AND LISTENERS

- events in game occur when something happens and changes the state (usually connections in behaviour between game objects)

- Examples include:
  - clicking
  - key presses
  - in game events like health damage



MULTIPLE EVENTS?

A class can have multiple events, just like a character can kick, move, jump, defend, lose/gain health and talk.

We use what's called an "event listener", these are:

- Methods in classes
- Exist for the sole purpose of waiting for events to happen,
  - When an event occurs, they are triggered.
  - Example, pressing the UP key (event) moves the player (action by listener)

MENU 01 07 12

WHO DOESN'T LIKE A DIAGRAM