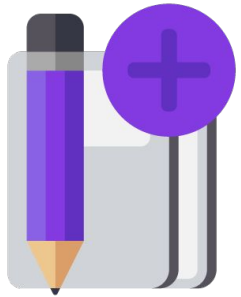
Brainstorming:

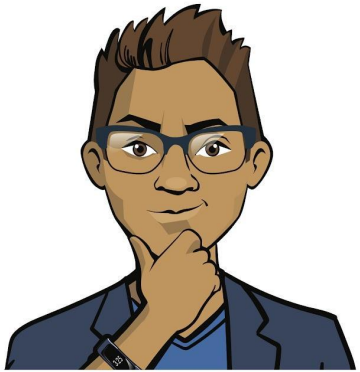
Creating classes



How do we create our own object?

We need to know the following:

- ➔ What kind of object is it? What data describes it?
- ➔ What can this object do? What actions does it perform?



Let's look at an example using a familiar object

Brain storming

To create your own object, you need to describe its data and actions.

- it has wheels
- it has headlights
- it has doors
- it can drive
- it starts
- people can sit in it
- it has mirrors

— it's a car

Brain storming

A class

is a common name for lots of objects;

in programming is a general description of how these objects should be structured.

IT'S a car

An object

Object class

Knowledge about all these objects

Brain storming

An instance of a class

is an object created according to the description programmed in the class.

A class

An instance of a class

Brain storming

An instance of a class

is an object created according to the description programmed in the class.

```
instance = Class()
```

The object receives everything that the class knows and knows how to do.

Properties

Methods

Creating an object

Describing the class:

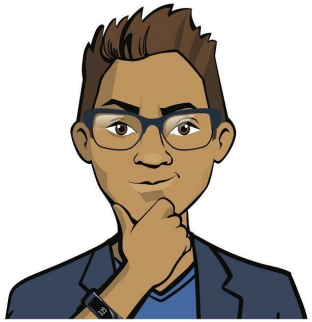
Properties

Methods

Brain storming

Which class do you already know?

You've been working with instances of a well-known class for a long time!



Brain storming

It's the Turtle class

Let's look at creating an *instance* of the Turtle class:

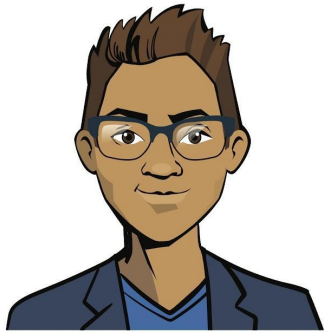
```
t1 = Turtle()
```

An instance of a class

Class **constructor**

The class name in parentheses is the **command** that creates a new object of that class.

The result is a link leading to the object (stored in a variable).

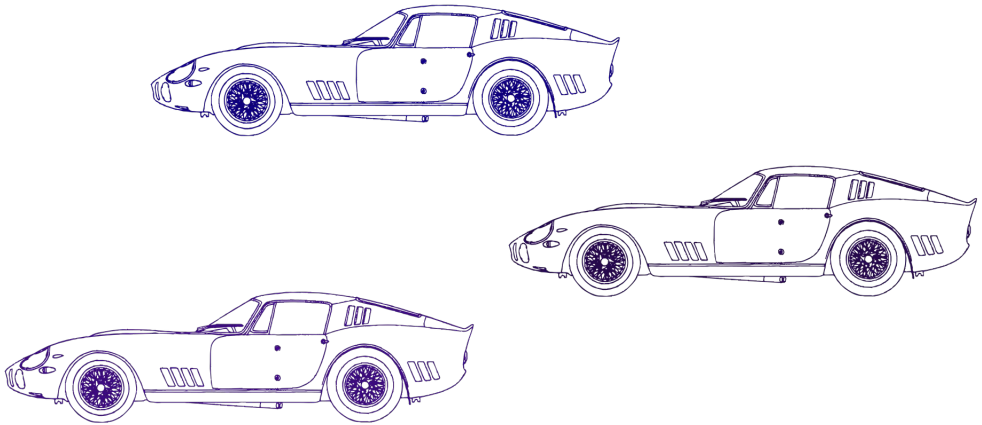


Brain storming

A constructor

is a method that is automatically called when an object is created. It creates an instance of the class.

The constructor often sets the properties of the object being created.



Brain storming

Creating a class

`class` is the command that creates a class.

`self` is the current object of the class.

```
class Class name ():
    def __init__(self, Data ):
        self. Property = Data

    def print_info(self):
        print('Information about the object:', self. Property )

Instance = Class name ( Property )
```

Brain storming

Creating a class

`class` is the command that creates a class.

`self` is the current object of the class.

```
class Class name ():
    def __init__(self, Data ):
        self. Property = Data

    def print_info(self):
        print('Information about the object:', self. Property )

Instance = Class name ( Property )
```

The command the description of the class begins with.

Brain storming

Creating a class

`class` is the command that creates a class.

`self` is the current object of the class.

```
class Class name ():
    def __init__(self, Data ):
        self. Property = Data

    def print_info(self):
        print('Information about the object:', self. Property )

Instance = Class name ( Property )
```

A constructor with the process of creating an instance of the class.

Brain storming

Creating a class

`class` is the command that creates a class.

`self` is the current object of the class.

```
class Class name ():
    def __init__(self, Data ):
        self. Property = Data

    def print_info(self):
        print('Information about the object:', self. Property )

Instance = Class name ( Property )
```

Class method (it can be any!)

Brain storming

Creating a class

`class` is the command that creates a class.

`self` is the current object of the class.

```
class Class name ():
    def __init__(self, Data ):
        self. Property = Data

    def print_info(self):
        print('Information about the object:', self. Property )

Instance = Class name ( Property )
```

Creating an instance of the class with the specified property.

Brain storming

Text Quest "The Knight and the Dragon"

The essence of the quest:

- ➔ The knight goes to the dragon's lair to fight it and collect its treasures.
- ➔ On the way to the lair, rascals appear. The knight can either go past them or engage in a fight with them.
- ➔ The knight gains experience for defeating a rascal. His armor gets tougher, and his strike gets stronger. But you could die in the fight and not make it to the dragon.

Brain storming

Text Quest "The Knight and the Dragon"

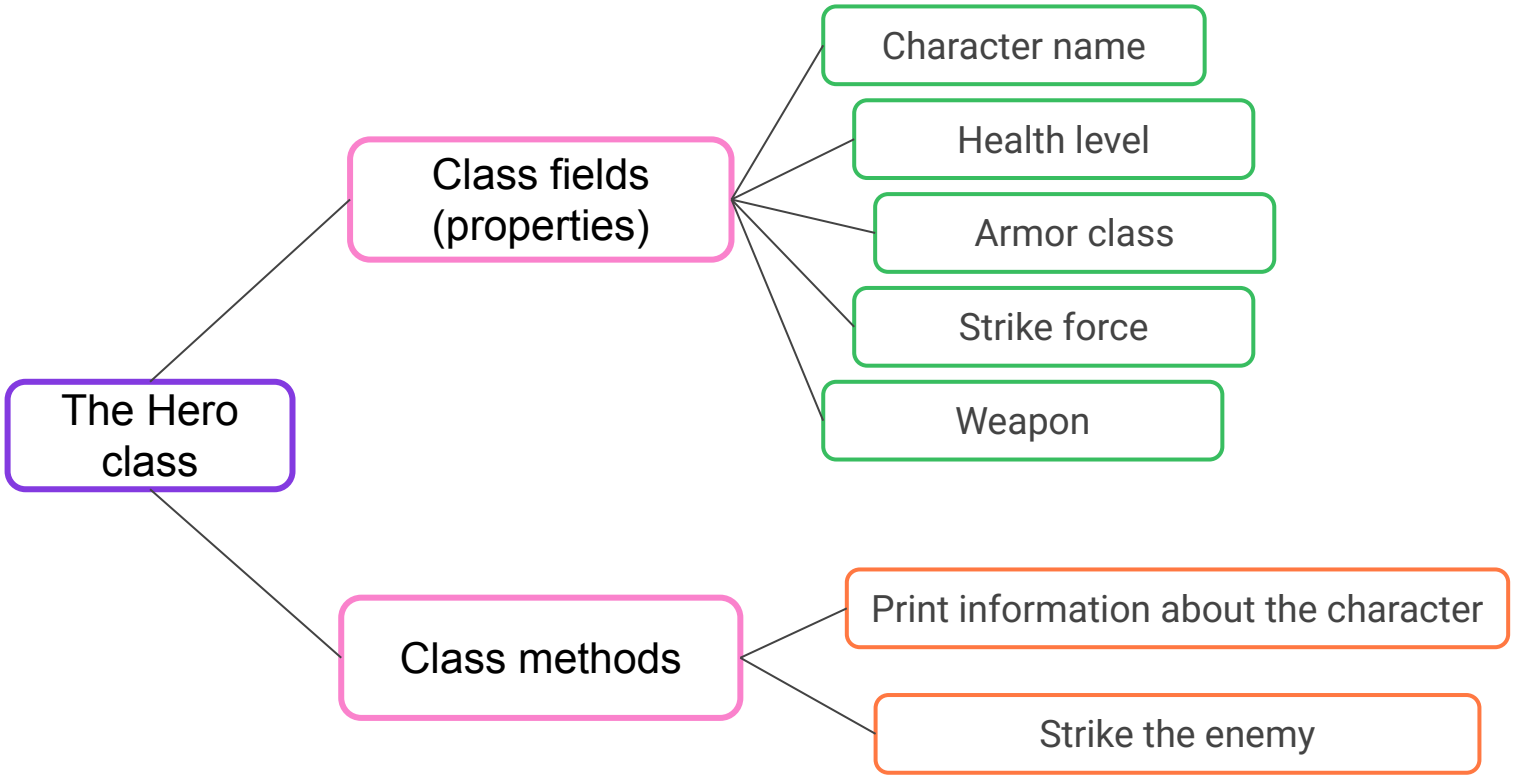
What are we going to do?

- ➔ **We'll create** the Hero class.
- ➔ **We'll describe the** class **fields in the constructor.**
- ➔ **We'll define the methods of the** class.
- ➔ **We'll create instances** of the Hero class: the knight, rascals, and dragon.
- ➔ **We'll program duels** between the instances of the class.

Brain storming

The Hero class

Let's program the class with the fields and methods set out in the mind map.



Brain storming

Hero class: constructor and printing data

When creating an instance of a class, a character with the specified properties must be created.

Health: 50

Richard

Armor: 25

Weapon: sword

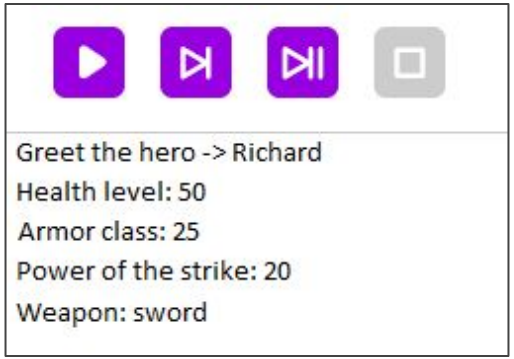
Strike force: 20

Brain storming

```
class Hero():
    def __init__(self, name, health, armor, power, weapon):
        self.name = name
        self.health = health #number
        self.armor = armor #number
        self.power = power #number
        self.weapon = weapon #string

    def print_info(self):
        print('Greet the hero ->', self.name)
        print('Health level:', self.health)
        #continue on your own

knight = Hero('Richard', 50, 25, 20, 'sword')
knight.print_info()
```



Brain storming

Hero class: strike() method

We'll create two instances of the class and program one hero to strike at the other.

Richard

Health: 50

Weapon: sword

Armor: 25

Strike force: 20

Helen

Health: 20

Weapon: bow and arrow

Armor: 5

Strike force: 5

Brain storming

Hero class: strike() method

We'll create two instances of the class and program one hero to strike at the other.

Richard

Health: 50

Weapon: sword

Armor: 25

Strike force: 20

Strike!

Helen

Health: 5

Weapon: bow and arrow

Armor: 0

Strike force: 5

Brain storming

We'll add a method to the class for attacking the enemy.

First, damage is dealt to the armor, and when there's none left, to the health.

```
def strike(self, enemy):
        print(
             '-> STRIKE! ' + self.name + ' is attacking ' + enemy.name +
             ' with a strike force of ' + str(self.power) + ', using a ' + self.weapon + '\n')

        enemy.armor -= self.power
        if enemy.armor < 0:
            enemy.health += enemy.armor
            enemy.armor = 0

        print(
             enemy.name + ' is reeling.\nArmor class down to '
             + str(enemy.armor) + ', and health level is up to '
             + str(enemy.health) + '\n')
```

Specify as a parameter the Hero object that is being struck.

Damage is dealt to the armor. If there is no armor left, subtract the rest from the health.

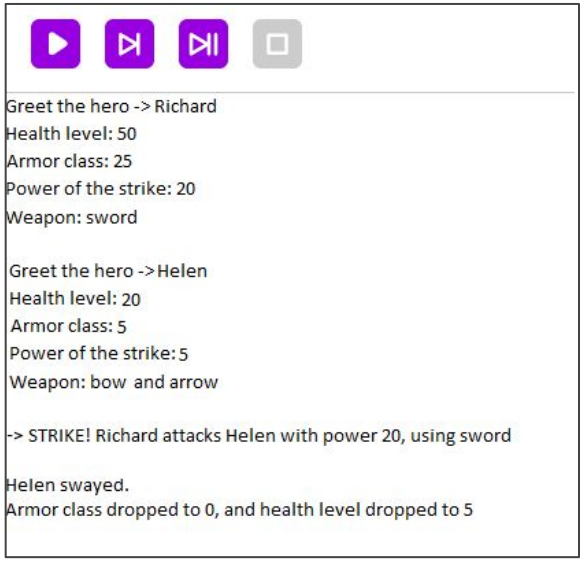
Brain storming

```
class Hero():
    def __init__(self, name, health, armor, power, weapon):
        Method body

    def print_info(self):
        Method body

    def strike(self, enemy):
        Method body
```

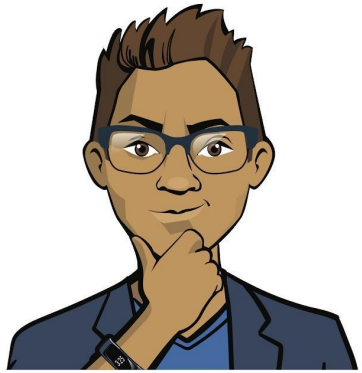
```
knight = Hero('Richard', 50, 25, 20, 'sword')
knight.print_info()
rascal = Hero('Helen', 20, 5, 5, 'bow and arrow')
rascal.print_info()
knight.strike(rascal)
```



Brain storming

Tasks:

- ➔ Create a Hero class with the listed fields and methods.
- ➔ Create two instances of the class: knight and rascal.
- ➔ Practice "striking" using the strike() method:
  - ◆ Rule for duels: strikes are delivered in turn.

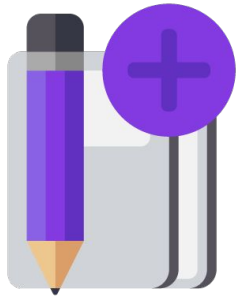


Brain storming

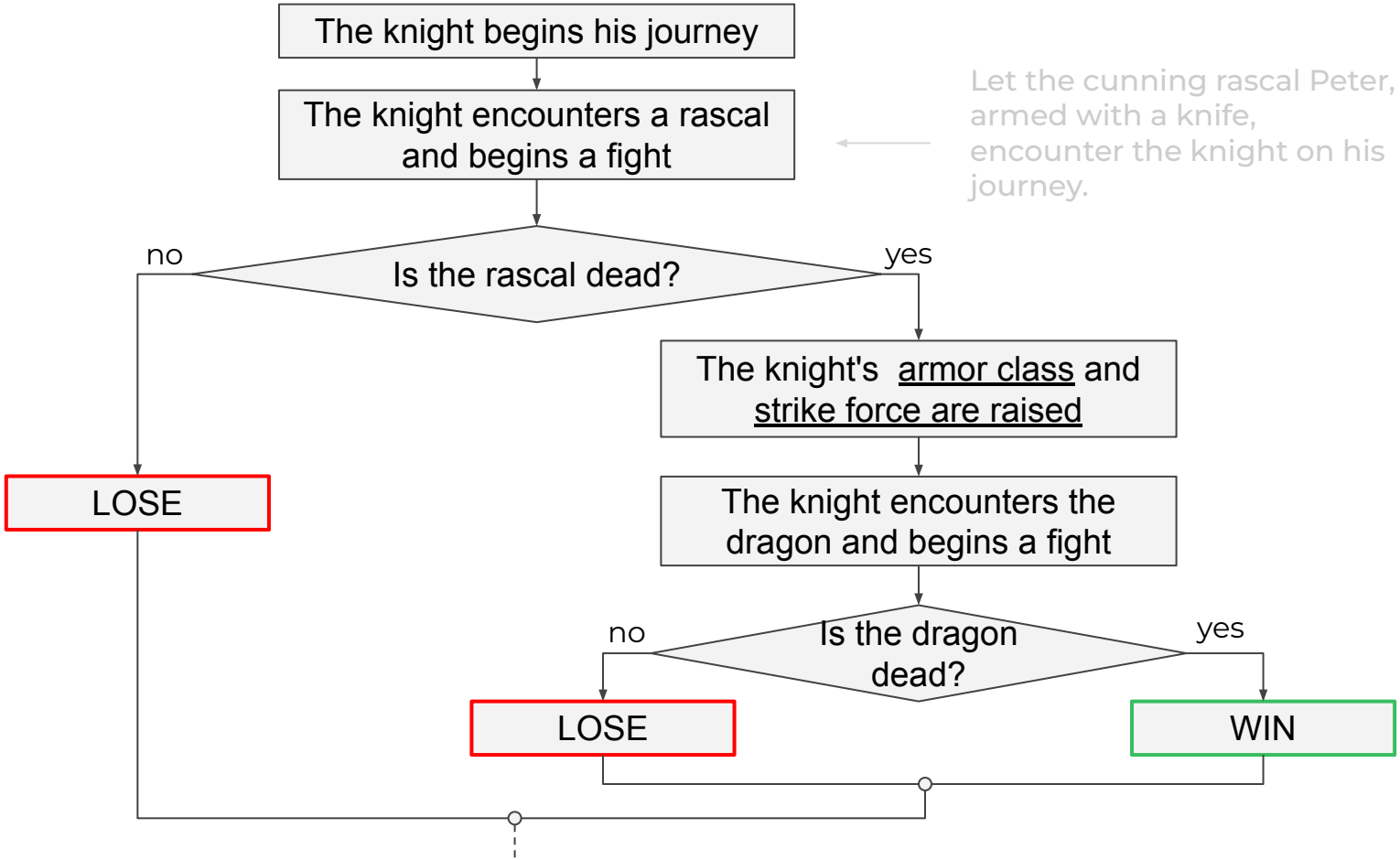
Module 5. Lesson 2. OOP. Events

Brainstorming:

Quest
The Knight and the Dragon



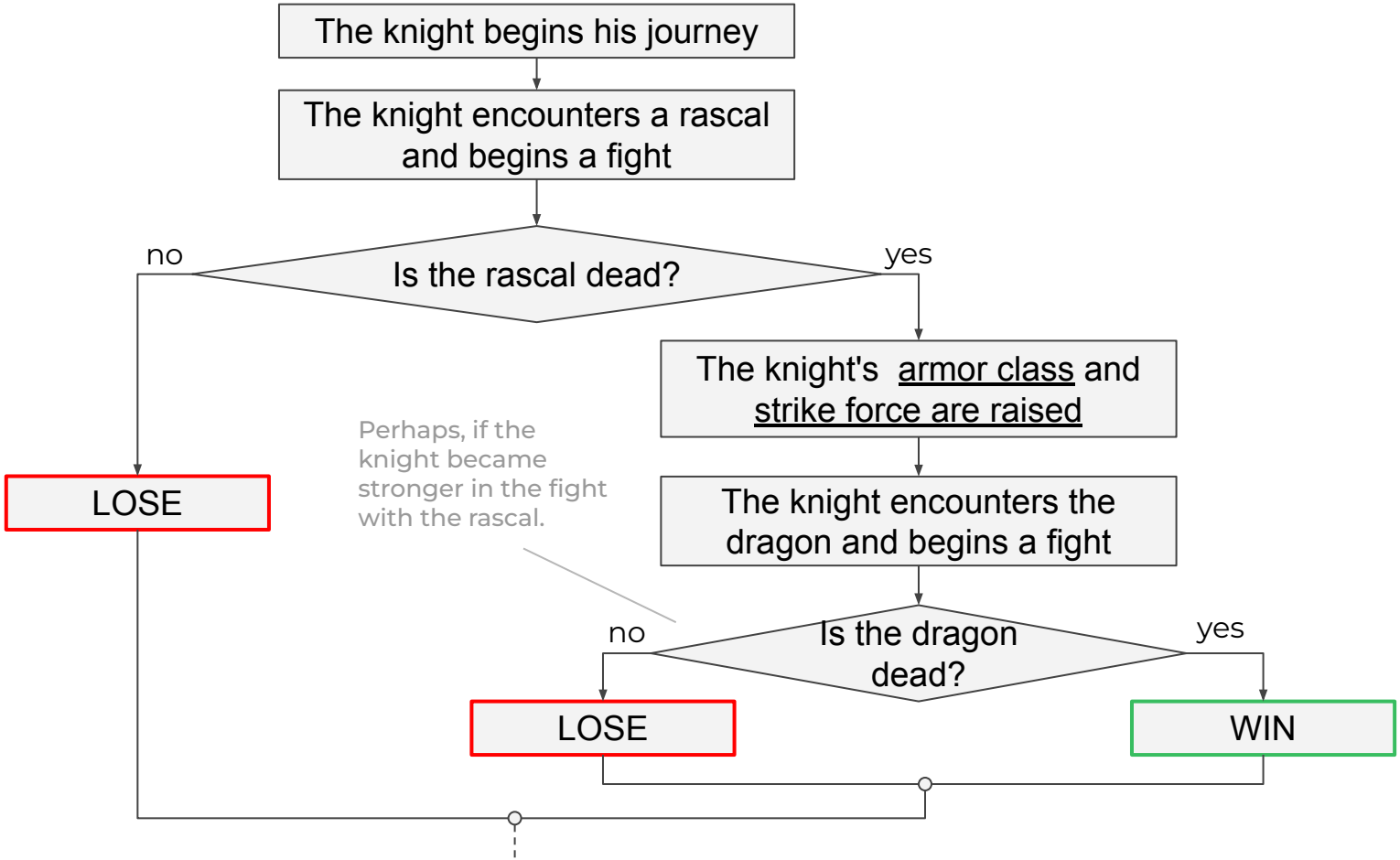
Game Quest "The Knight and the Dragon"



Let the cunning rascal Peter, armed with a knife, encounter the knight on his journey.

Brain storming

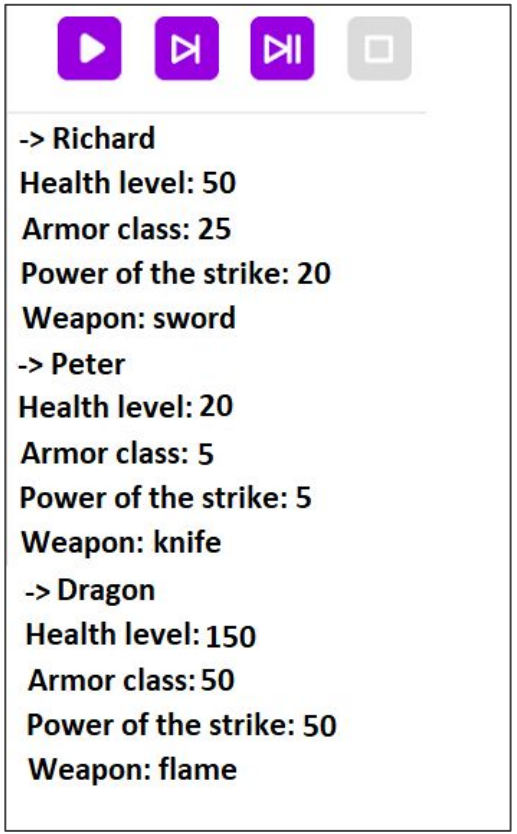
Game Quest "The Knight and the Dragon"



Brain storming

Step 1 - save the Hero as a module

- ➔ Leave only the Hero class in the level. Save it as the Hero module.
- ➔ In the next level, connect the **Hero module**, **create the Knight, Rascal, and Dragon characters**, and **print information** about them.



Brain storming

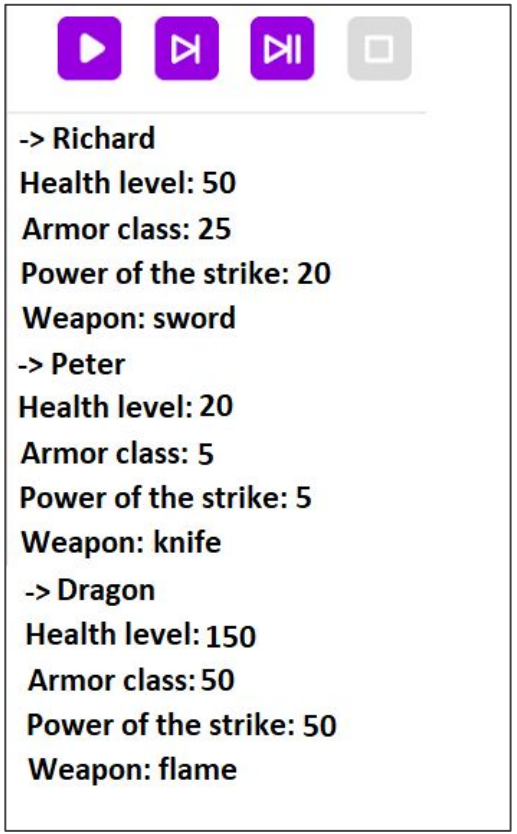
Step 1 - save the Hero as a module

- ➔ Leave only the Hero class in the level. Save it as the Hero module.
- ➔ In the next level, connect the **Hero module**, **create the Knight, Rascal, and Dragon characters**, and **print information** about them.

```
from hero import Hero

knight = Hero('Richard', 50, 25, 20, 'sword')
rascal = Hero('Peter', 20, 5, 5, 'knife')
dragon = Hero('Dragon', 150, 50, 50, 'flame')

#complete the data printing yourself
```



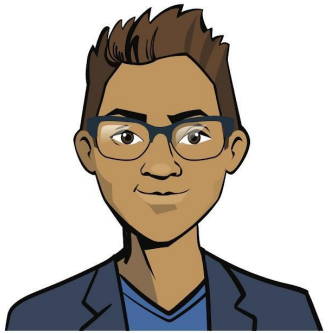
Brain storming

Step 2 - create the fight() method

The essence of the method:

- The participants in the fight strike each other (the strike() method) until only one of them remains alive.

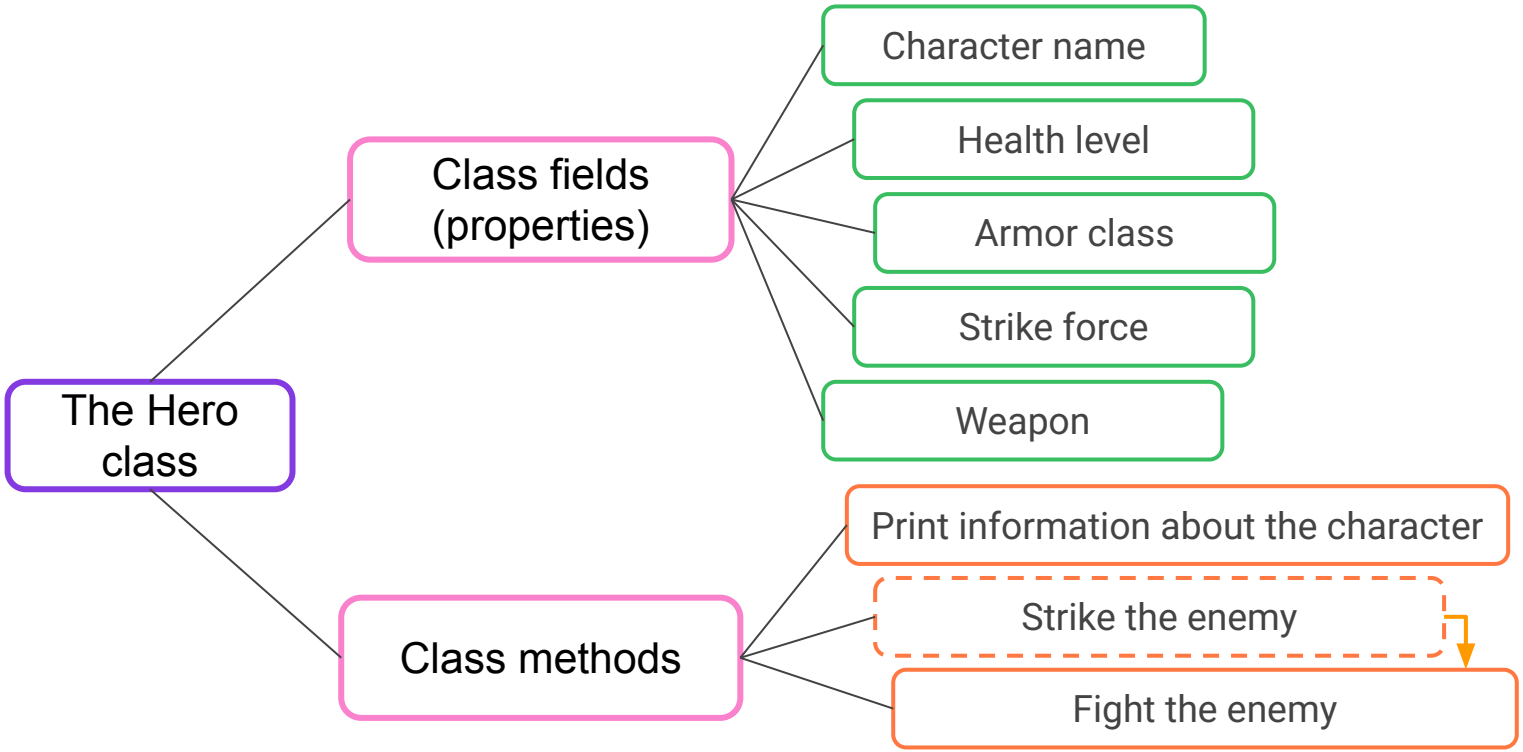
How do we program that method?
Is it possible to call the strike() method in the fight() method?



Brain storming

We'll add the fight() method to the Hero class

We're going to use the strike() method in the body.



Brain storming

We'll add the fight() method to the Hero class

We're going to use the strike() method in the body.

```
  def fight(self, enemy):
       while self.health and enemy.health > 0:
           self.strike(enemy)
           if enemy.health <= 0:
               print(enemy.name, 'has fallen in this difficult
battle!\n')
               break
           sleep(5)

           enemy.strike(self)
           if self.health <= 0:
               print(self.name, 'has fallen in this difficult
battle!\n')
                break
           sleep(5)
```

As long as the health level of the opponents is above 0.

The characters attack each other in turn.

If the health level of one of them is less than 0, the battle is over.

We'll put pauses to generate intrigue.

Brain storming

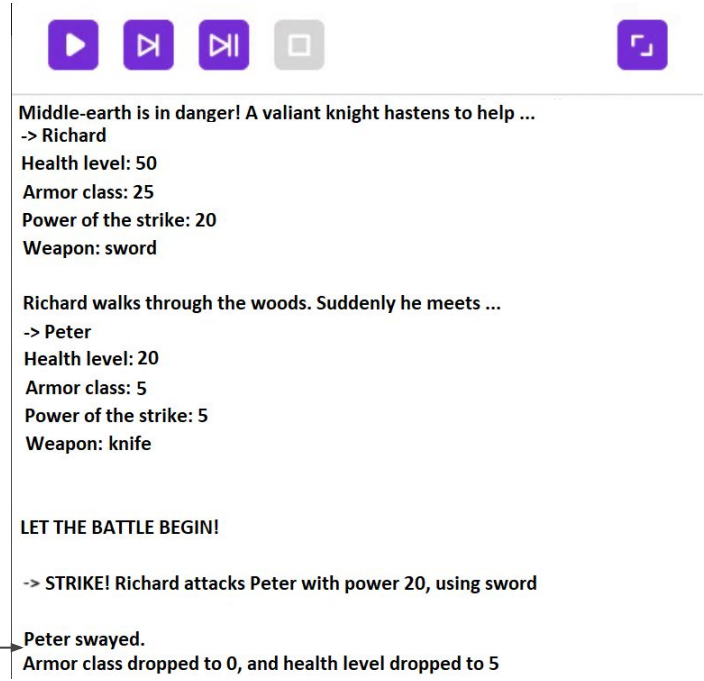
```
class Hero(): #Hero module
    def __init__(self, name, health, armor, power, weapon):
        Method body

    def print_info(self):
        Method body

    def strike(self, enemy):
        Method body

    def fight(self, enemy):
        Method body
```

Brain storming



Let's program the battle between the Knight and the Rascal in the next level.

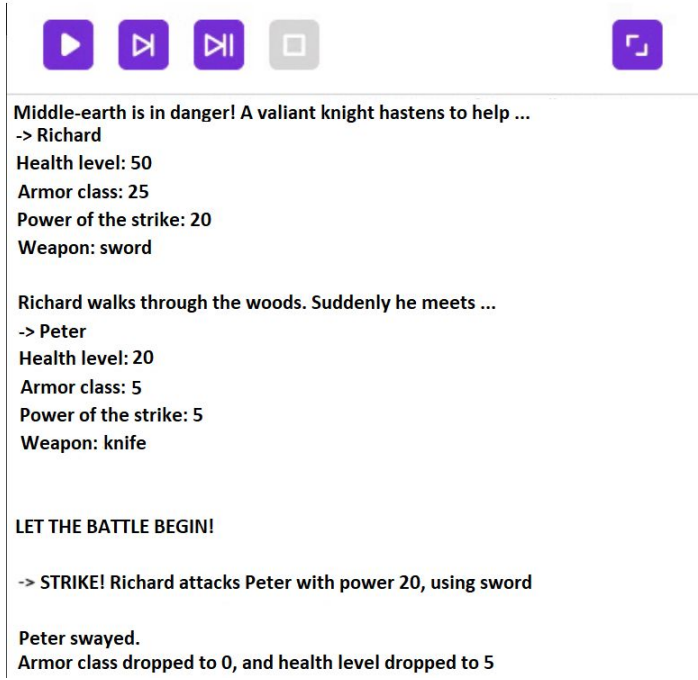
The main part of the program:

```
from hero import Hero

knight = Hero('Richard', 50, 25, 20, 'sword')
knight.print_info()

rascal = Hero('Peter', 20, 5, 5, 'knife')
rascal.print_info()

sleep(5)
knight.fight(rascal)
```



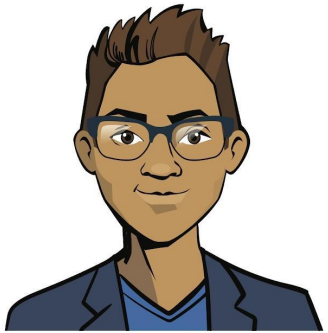
Brain storming

Step 3 - improving the strike and armor

The essence of the method:

- if **the knight survives**:
  - the health is restored to its initial value;
  - the armor class is doubled;
  - the strike force is doubled.

How do we program restoring the health and improving the armor and striking?



Brain storming

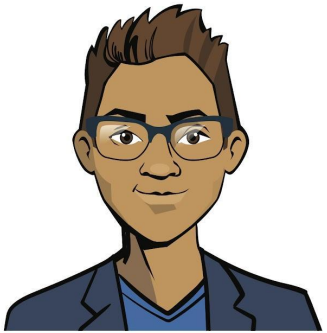
We'll add to the main part of the program

```
if knight.health > 0:
    knight.health = 100
    knight.power *= 2
    knight.armor = 10 * 2
    print(
        '\n' + knight.name +
        'regained their strength and gained more experience.
  Now the strike force is: ' +  str(knight.power) + ', and
  armor class:' +   str(knight.armor) + '\n')
```

If the knight survives.

We restore the health, raise the strength, and double the armor.

We notify the user about those changes.



Brain storming

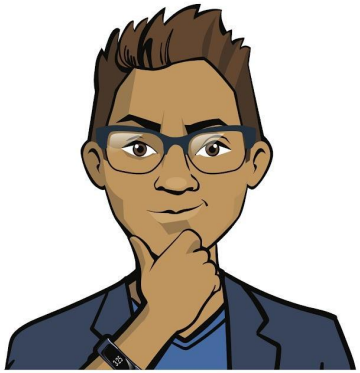
Adding to the code for the fight with the Dragon,
and shaping the narration:

```
Middle-Earth is in danger! A valiant knight is hurrying to the rescue...
->Richard
Health level: 50
Armor class: 25
Power of the strike: 20
Weapon: sword
```

Brain storming

Tasks:

- ➔ Move the Hero class **into a separate module.**
- ➔ We'll add the **fight() method** to the Hero class.
- ➔ Program the quest!
  - ◆ Program **the duel between the knight and the rascal**.
    If the knight wins, raise his armor class and strength.
  - ◆ Program **the duel between the knight and the dragon**.



Brain storming