# Python Classes Worksheet

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Part 1: Starter Example - Meet the Robot

Here's a simple Python class to get started:

class Robot:  
 def \_\_init\_\_(self, name, color):  
 self.name = name  
 self.color = color  
   
 def say\_hello(self):  
 print(f"Hello! My name is {self.name} and I am {self.color}.")

Activity 1: Try It Out

1. Copy the code above into your Python editor.  
2. Create two robots and make them introduce themselves. Here's an example to help you:

robot1 = Robot("Robo1", "Red")  
robot2 = Robot("Robo2", "Blue")  
  
robot1.say\_hello()  
robot2.say\_hello()

Questions:

1. What do you notice when you run the code?

2. Can you change the name and color of the robots to something else?

## Part 2: Adding More Attributes

Expand the robot class to include a power level:  
  
class Robot:  
 def \_\_init\_\_(self, name, color, power):  
 self.name = name  
 self.color = color  
 self.power = power  
   
 def say\_hello(self):  
 print(f"Hello! My name is {self.name}, I am {self.color}, and my power level is {self.power}.")

Activity 2: Modify the Code

1. Add the new power attribute to your robots:

robot1 = Robot("Robo1", "Red", 50)  
robot2 = Robot("Robo2", "Blue", 80)

2. Update the say\_hello method to include the power level.

Challenge: Create a third robot with a different name, color, and power level. Make all three robots introduce themselves.

## Part 3: Adding Actions

Let’s give the robot the ability to charge its power level:  
  
class Robot:  
 def \_\_init\_\_(self, name, color, power):  
 self.name = name  
 self.color = color  
 self.power = power  
   
 def say\_hello(self):  
 print(f"Hello! My name is {self.name}, I am {self.color}, and my power level is {self.power}.")  
   
 def charge(self):  
 self.power += 10  
 print(f"{self.name} is charging! Power level is now {self.power}.")

Activity 3: Charging the Robots

1. Add the charge method to your code.

2. Use the charge method on one of your robots:

robot1.charge()

Questions:

1. What happens when you charge the robot multiple times?

2. Can you add a message to say when the robot’s power level reaches 100?

## Part 4: Customizing Your Robots

Activity 4: Design Your Own Robot

1. Create a new robot class with:  
 - Attributes: name, type (e.g., "cleaner", "fighter"), and speed.  
 - Methods:  
 - introduce: Prints the robot’s details.  
 - boost\_speed: Increases the robot’s speed by 5.  
2. Example to get you started:  
  
class CustomRobot:  
 def \_\_init\_\_(self, name, type, speed):  
 self.name = name  
 self.type = type  
 self.speed = speed  
   
 def introduce(self):  
 print(f"I am {self.name}, a {self.type} robot, with a speed of {self.speed}.")  
   
 def boost\_speed(self):  
 self.speed += 5  
 print(f"{self.name}'s speed increased to {self.speed}!")

Challenge:

1. Create two custom robots with different types and speeds.

2. Make them introduce themselves.

3. Boost their speed and print the new values.

## Extension: Battle the Robots

Activity 5: Robot Battle!  
1. Add a fight method to the Robot class. Example:  
  
def fight(self, other\_robot):  
 if self.power > other\_robot.power:  
 print(f"{self.name} wins the battle!")  
 elif self.power < other\_robot.power:  
 print(f"{other\_robot.name} wins the battle!")  
 else:  
 print("It's a tie!")  
  
2. Use the fight method to pit two robots against each other:  
  
robot1.fight(robot2)

Challenge: Add a health attribute to the robot and make the fight method reduce the loser’s health by 10.

## Reflection Questions

1. What was the most fun part of creating and modifying robots?

2. What is something you still find tricky about classes?

3. Can you think of other real-world things that could be modeled using classes?