Q1. Take a look at these two classes.

```
class ComputerLecturer(object):
    salary = 100000
    monthly_bonus = 500
    def __init__(self, name, age, address, phone,
programming_modules):
        self.name = name
        self.age = age
        self.address = address
        self.phone = phone
        self.programming_modules = programming_modules


class LinguisticsLecturer(object):
    salary = 100000
    monthly_bonus = 500
    def __init__(self, name, age, address, phone, bilingual):
        self.name = name
        self.age = age
        self.address = address
        self.phone = phone
        self.bilingual = bilingual
```

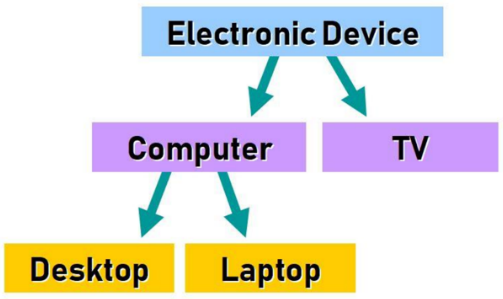
Don't you notice something very peculiar? **There is a lot of code repetition!** Each one of these two **classes has the** attributes name, age, address, phone, salary, and monthly_bonus. salary and monthly_bonus even have the same value. Use the inheritance concepts that you learnt so far to avoid code the code repetition and optimise your code.

Q2. Mammal and Panda: Attribute Inheritance

- Create a Panda class that inherits from the Mammal class (see below).
- Make Panda inherit all the attributes defined in Mammal
- Add a class attribute to Panda: is_endangered = True
- Add an instance attribute: code
- Create an instance of Panda and store it in the variable my_panda. You can choose any values as the arguments used to create the instance.

```
class Mammal:
    def __init__(self, name, age, health, num_offspring,
years_in_captivity):
        self.name = name
        self.age = age
        self.health = health
        self.num_offspring = num_offspring
        self.years_in_captivity = years_in_captivity
# Define the Panda class below this line
```

Q3. Take the following diagram and convert the hierarchy into Python code.



You are free to implement the classes as you wish. Please include attributes and methods in each class and follow the style guidelines that you have learned during the oop weeks. Create some instances and test your code with some creative scenarios to set and display values.

Tips:

- Make sure that the most general attributes for example (voltage, weight, height, and color etc) should be included in the superclass ElectronicDevice because they are shared by all the subclasses.
- Then, in the subclasses, you include attributes that only apply to them and to their potential subclasses. For example, the Computer class could have an attribute memory and hard_drive which applies to all instances of Computer. Its subclasses, Desktop and Laptop will inherit these attributes and this way, we avoid code repetition.

Q4. Create two classes that inherit from the Pizza class (See the code below)

- PizzaMargherita class. Instance attribute: has_extra_cheese
- PizzaMarinara class. Instance attribute: has_extra_basil
- These two classes must inherit all the instance attributes of the Pizza class.
- Create two instances, one of each subclass and assign them to their corresponding variable

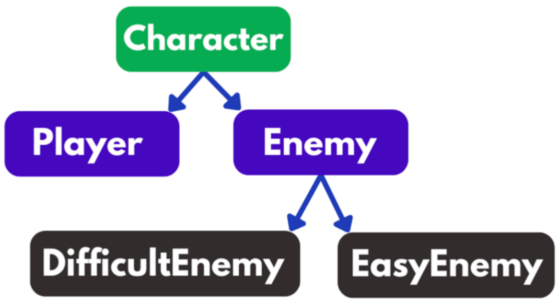
```
class Pizza:
    def __init__(self, size, toppings, price, rating):
        self.size = size  # "Small", "Medium", or "Large"
        self.toppings = toppings  # A list of toppings
        self.price = price
        self.rating = rating  # Scale from 1 to 5
# Add the subclasses below this line
```

Q5. You just signed up for a video game development competition and your team decided to use inheritance to represent the characters. Some team members have made mistakes in the code and the inheritance is not working correctly. The due date to submit the game is tomorrow and you are the only one who can save your team from being disqualified.

Your task is to:
Fix the errors in the code developed by your team.
Implement the correct hierarchy.

Requirements:
Enemy must be a subclass of **Character**.
Player must be a subclass of **Character**.
Enemy must be a superclass of **DifficultEnemy** and **EasyEnemy**.

Hierarchy:
The hierarchy can be illustrated like this:



Code:
This is the code that your team wrote. It throws many errors and the inheritance is not defined correctly.

```
class Sprite:

    def __init__(self, x, y, img_file, speed, life_counter):
        self.x = x
        self.y = y
        self.img_file = img_file
        self.speed = speed
        self.life_counter = life_counter


class Enemy(Sprite):

    def __init__(self, x, y, img_file, speed):
        __init__(self, x, y, img_file, speed, 5)
        self.message = "I'm here to protect my master"


class Player(Enemy):

    def __init__(self, x, y, img_file, speed):
        Sprite.(self, y, img_file, speed, 6)
        self.speed = 56


class DifficultEnemy(Enemy):
      def __init__(self, x, y, img_file):
```

```
                Enemy.__init__(self, img_file, 80)


class EasyEnemy(Player):
    Enemy.__init__(self, x, y, img_file, 40)
    def __init__(self, x, y, img_file):
          self.life_counter = 1
```

Q6. Please use the following overriding example:

```
class Teacher:

    def __init__(self, full_name, teacher_id):
        self.full_name = full_name
        self.teacher_id = teacher_id

    def welcome_students(self):
        print(f"Welcome to class!, I'm your teacher. My name
is {self.name}")
class ScienceTeacher(Teacher):

    def welcome_students(self):
        print(f"Science is amazing.")
        print(f"Welcome to class. I'm your teacher:
{self.name}")

my_science_teacher = ScienceTeacher("Emily Smith", "S355A213")
my_science_teacher.welcome_students()
```

Update the code above so the welcome_students method in the superclass can be executed when you run the code of `my_science_teacher.welcome_students()`