Inheritance in Python

Step 1 Create and design a class

```
# Create a class called Patient.
# 'object' is the class from which all other classes inherit.
class Patient(object):

    # In a docstring, describe what the purpose of the class is.
    ''' Register a patient in a medical centre '''

    # Define a shared variable across the whole class that is the same
for each instance of the class.
    # This is what we call a class variable.
    status = 'Patient'

    # Create a constructor which controls the varibles that each
object can have.
    # The 'self' keyword serves as a placeholder for each instance of
the Patient class.
    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.note = []

    # Functions inside classes are called methods.
    # Methods help the user interact with the instance of a certain
class.
    # In Python, methods are accessible through the dot-notation.
    # Below, we define a method that would allow the user to add a
note to each patient.
    # Every time the user creates a new note, it will be appended to
the old one, so that a record is kept.
    def add_note(self, note):
        ''' Add a note to keep track of the health of a patient. '''

        self.note.append(note)

    # Below, we define a method that returns the information for a
patient.
    def get_details(self):
        ''' Retrieve the current information available for the
patient. '''

        print(f'Patient record \nName: {self.name} \nAge: {self.age} \
nNote: {self.note}')
```

Step 2 Create a new class which inherits everything from the Patient class

```
# Define a class that will be used to create a subset of all patients,
namely, infants.
# The Infant class would inherit all class variables and methods
defined in the Patient class.
class Infant(Patient):
    ''' Patients under 2 years old. '''

    def __init__(self, name, age):

        # Create a variable unique to the Infant class.
        # Once an object of the Infant class is instantiated, an empty
'vaccinations' list is automatically createsd.
        self.vaccinations = []

        # The super() function searches the Patient class for the
__init__ method.
        # It therefore initializes the variables from the Patient
class once an instance of the Infant class
        # is created.
        # Therefore, upon instantiation, an infant would have a name,
an age, a list of notes,
        # a list of vaccinations, and a status.
        super().__init__(name, age)

    # Define a new method unique to the Infant class.
    def add_vac(self, vaccine):
        ''' Keep a record of the vaccines the infant has gotten. '''

        self.vaccinations.append(vaccine)

    # Override the get_details() method that was defined in the
Patient class.
    def get_details(self):
        ''' Retrieve the current information available for the
patient. '''

        print(f'Patient record \nName: {self.name} \nAge: {self.age} '
\
                  f'\nNote: {self.note} \nVaccines:
{self.vaccinations}')
```

Step 2 Create an object - an instance of the Patient class

```
# Create an instance of the Patient class and an instance of the
Infant class.
```

```
alice = Patient('Alice', 42)
bob = Infant('Bob', 1)
```

Step 3 Get the details for each patient

```
# Access the get_details() method through the dot-notation.
alice.get_details()

# Access the get_details() method through the dot-notation.
# Note the additional list of vaccines.
bob.get_details()
```

Step 4 Add a note for each patient

```
alice.add_note('The patient had a slight stomach discomfort.')
alice.add_note('The discomfort had disappeared in two days time.')

bob.add_note('The patient is healthy and well.')
```

Step 5 Add a vaccine

```
# Access the add_vac() method through the dot-notation.
bob.add_vac('MMR')
```

Step 6 Get the updated details for both patients

```
alice.get_details()

bob.get_details()
```