## Groups

Create a class called **Person**. Upon initialization it will receive a **name** (str) and a **surname** (str). Implement the needed **magic methods**, so that:

* Each person could be represented by his/ her **names, separated by a single space**.
* When you concatenate two people you should return a **new instance** of a person who will take **the first name from the first person and the surname from the second person**.

Create another class called **Group**. Upon initialization it should receive a **name** (str) and **people** (list of Person instances). Implement the needed **magic methods**, so that:

* When you access the **length of a group instance**, you should receive the **total number of people** in the group.
* When you **concatenate two groups**, you should return a **new instance** of a group which will have a name -string in the format **"{first\_name} {second\_name}"** and **all the people** in the two groups will participate in the new one too.
* Each group should be represented in the format **"Group {name} with members {members' names separated by comma and space}"**
* You could **iterate over a group** and **each person** (element of the group) should be represented in the format **"Person {index}: {person's name}"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| p0 = Person('Aliko', 'Dangote')  p1 = Person('Bill', 'Gates')  p2 = Person('Warren', 'Buffet')  p3 = Person('Elon', 'Musk')  p4 = p2 + p3  first\_group = Group('\_\_VIP\_\_', [p0, p1, p2])  second\_group = Group('Special', [p3, p4])  third\_group = first\_group + second\_group  print(len(first\_group))  print(second\_group)  print(third\_group[0])  for person in third\_group:  print(person) | 3  Group Special with members Elon Musk, Warren Musk  Person 0: Aliko Dangote  Person 0: Aliko Dangote  Person 1: Bill Gates  Person 2: Warren Buffet  Person 3: Elon Musk  Person 4: Warren Musk |