In [1]:

```
#Herança de classes em Python
```

In [2]:

```
class Pessoa:   #classe já definida anteriormente
    def __init__(self, nome):
        self._nome=nome
    def getNome(self):
        return self._nome
    def __str__(self):
        return 'Chamo-me ' + self.getNome()
```

In [3]:

```
class Aluno(Pessoa):    #criação da subclasse Aluno, por derivação da classe base Pesso
a
    def __init__(self, nome, numero):
        super().__init__(nome) # or Pessoa.__init__(self,nome)
        self._numero=numero
    def __str__(self):
        return super().__str__() + ' e tenho o nº ' + str(self._numero)
```

In [4]:

```
class Professor(Pessoa):    #criação da subclasse Professor, por derivação da classe ba
se Pessoa
    def __init__(self, nome, categoria):
        super().__init__(nome) # or Pessoa.__init__(self,nome)
        self._categoria=categoria
    def __str__(self):
        return super().__str__() + ' e sou ' + self._categoria
```

In [5]:

```
a=Aluno('Ana',10)
```

In [6]:

```
p=Professor('Rui','Catedrático')
```

In [7]:

```
pessoas=[a, p, Aluno('Pedro', 25), Pessoa('Rita')]  # coleção híbrida
```

In [8]:

```
for x in pessoas:
    print(x)    #polimorfismo: esta mesma instrução significa coisas diferentes em temp
o de execução (invoca o método __str__() do objeto referenciado por x, ou seja, um dos
 3 métodos __str__())
```

```
Chamo-me Ana e tenho o nº 10
Chamo-me Rui e sou Catedrático
Chamo-me Pedro e tenho o nº 25
Chamo-me Rita
```