

```
In [ ]: #passo1
```

```
In [ ]: class Pessoa:
        def __init__(self, nome):
            self._nome=nome
```

```
In [ ]: p1=Pessoa("Ana")
```

```
In [ ]: print(p1)
        print(p1._nome)  #não recomendável

<__main__.Pessoa object at 0x000001A93494F0A0>
Ana
```

```
In [ ]:
```

```
In [ ]: #passo2
```

```
In [ ]: class Pessoa:
        def __init__(self, nome):
            self._nome=nome
        def getNome(self):
            return self._nome
        def __str__(self):
            return 'Eu chamo-me ' + self.getNome()
```

```
In [ ]: p1=Pessoa("Ana")
```

```
In [ ]: print(p1)
        print(p1.__str__())
        print(str(p1))
        print(p1.getNome())
```

```
Eu chamo-me Ana
Eu chamo-me Ana
Eu chamo-me Ana
Ana
```

```
In [ ]:
```

```
In [ ]: #passo3
```

```
In [ ]: p2=Pessoa("Ana")
```

```
In [ ]: p2==p1
```

```
Out[ ]: False
```

```
In [ ]: p2 is p1
```

```
Out[ ]: False
```

```
In [ ]: class Pessoa:
        def __init__(self, nome):
            self._nome=nome
        def getNome(self):
            return self._nome
        def __str__(self):
            return 'Eu chamo-me ' + self.getNome()
        def __eq__(self, outro):
            if not isinstance(outro, Pessoa): return False
            return self._nome==outro._nome
```

```
In [ ]: p1=Pessoa("Ana")
        p2=Pessoa("Ana")
        p3=Pessoa("Rui")
```

```
In [ ]: p1==p2
```

```
Out[ ]: True
```

```
In [ ]: p1.__eq__(p2)
```

```
Out[ ]: True
```

```
In [ ]: p1==p3
```

```
Out[ ]: False
```

```
In [ ]: p1 is p2
```

```
Out[ ]: False
```

```
In [ ]: p4=p1
```

```
In [ ]: p4 is p1
```

```
Out[ ]: True
```

```
In [ ]: p4==p1
```

```
Out[ ]: True
```

```
In [ ]:
```

```
In [ ]: #passo4 - definamos uma função geradora e, com ela, um iterador
```

```
In [ ]: def fg(): #função geradora  
        yield 'estes são os vários resultados'  
        yield 'devolvidos'  
        yield 'pelo método __next__()' #este yield não é necessário  
        yield 'dum objeto gerado por esta função'
```

```
In [ ]: g=fg() #o objeto g tem o método __next__(), Logo pode ser usado como iterador (é um tipo de itera  
dor)
```

```
In [ ]: next(g)
```

```
Out[ ]: 'estes são os vários resultados'
```

```
In [ ]: next(g)
```

```
Out[ ]: 'devolvidos'
```

```
In [ ]: next(g)
```

```
Out[ ]: 'pelo método __next__()'
```

```
In [ ]: next(g)
```

```
Out[ ]: 'dum objeto gerado por esta função'
```

```
In [ ]: next(g)
```

```
-----  
StopIteration                                Traceback (most recent call last)  
<ipython-input-29-e734f8aca5ac> in <module>  
----> 1 next(g)
```

```
StopIteration:
```

```
In [ ]:
```