Conversão para cadeias de caracteres

Juliana Pirolla - Revisão

Motivação

Quando definimos uma classe, os objetos dela recebem uma representação padronizada para ser mostrada ao usuário. Muitas vezes, a representação padrão não é o que buscamos (por exemplo, quando é o endereço de memória).

Assim sendo, é interessante que haja formas de redefinir as formas que serão expressas ao usuário.

Como redefinir a forma que os objetos serão expressos ao usuário?

POdemos redefinir a forma que será expressa ao usuário definindo o método especial `__repr__` (representation) que **retorna a string do objeto**.

Por padrão, se o método **repr**() não estiver definido em uma classe, ele retornará a mesma representação em string fornecida pelo método **str**().

```
class A:
    def __init__(self,val):
        self._val = val

    def get_val(self):
        return self._val

    def __repr__(self):
        return f'A({str(self._val)})'
```

```
a = A(10)
a.get_val()
```

```
10
```

```
a
```

```
A(10)
```

Quando utilizamos repr?

Esse método será usado nas situações em que o objeto precisa ser mostrado para o usuário, ou precisa ser convertido para cadeia de caracteres.

Conversão para cadeias

Em alguns casos queremos que essa conversão de objeto pra cadeia de caracteres seja diferente da forma convencional de mostrar o objeto. Assim, podemos além de definir o `__repr__`, definir também o `__str__`.

```
class B:
    def __init__(self, val) -> None:
        self._val = val

    def get_val(self):
        return self._val

    def __repr__(self):
        return f'B({str(self._val)})'

    def __str__(self):
        return str(self._val)
```

```
b = B(12)
b
```

```
B(12)
```

```
'O valor é' + str(b)
```

```
'O valor é12'
```

```
class B2:
    def __init__(self, val) -> None:
        self._val = val

    def get_val(self):
        return self._val

    def __repr__(self):
        return f'B({str(self._val)})'

    def __str__(self):
        return f'oi, eu sou a chamada do método str e retorno o valor: {str(self._val)}'
```

```
b2 = B2(12)
b2
```

```
B(12)
```

```
str(b2)
print(b2)
```

```
oi, eu sou a chamada do método str e retorno o valor: 12
```

Note que quando fazemos o print, estamos chamando o str.

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __repr__(self):
        return f'Point(x={self.x}, y={self.y})'

    def __str__(self):
        return f'({self.x}, {self.y})'
```

```
p = Point(2, 3)
p
```

```
Point(x=2, y=3)
```

```
print(f'A point at {p}')
```

```
A point at (2, 3)
```