Aula 07

Programação Orientada a Objetos

```python
class Carro:
    """Classe Carro"""

    def __init__(self, modelo: str, cor: str, ano: int) -> None:
        self.modelo: str = modelo
        self.cor: str = cor
        self.ano: int = ano
        self.pisca: bool = False

    def buzina(self) -> None:
        """Buzina e verifica o estado do pisca do carro."""
        print(f'O carro {self.modelo} está buzinando.')
        print(f'e está com o pisca {"ligado" if self.pisca else "desligado"}.')

    def toggle_pisca(self) -> None:
        """Liga ou desliga o pisca do carro."""
        self.pisca = not self.pisca
```

```python
if __name__ == '__main__':
    volvo = classes.Carro('Volvo', 'Verde', 2000)
    volvo.buzina()
    volvo.toggle_pisca()
    volvo.buzina()
```

```
PS C:\Users\augusto.hertzog\GitHub\master-python> python main.py
O carro Volvo está buzinando.
e está com o pisca desligado.

O carro Volvo está buzinando.
e está com o pisca ligado.

PS C:\Users\augusto.hertzog\GitHub\master-python> _
```

Atributos – part Deux

Atributos de Classe

Os atributos de classe são variáveis definidas dentro de uma classe em Python, que são compartilhadas por todas as instâncias dessa classe. Enquanto os atributos de instância são específicos de cada objeto criado a partir da classe, os atributos de classe são os mesmos para todas as instâncias.

Quando um atributo de classe é definido, ele se torna acessível a todas as instâncias da classe, bem como à própria classe. Esses atributos são úteis para armazenar informações que são comuns a todas as instâncias, como constantes, configurações, contadores, entre outros.

A definição de um atributo de classe ocorre fora dos métodos da classe, e sim no escopo da própria classe.

```python
class Carro:
    """Classe Carro"""
    qtd_rodas:int = 4

    def __init__(self, modelo: str, cor: str, ano: int) -> None:
        self.modelo: str = modelo
        self.cor: str = cor
        self.ano: int = ano
        self.pisca: bool = False

    def buzina(self) -> None:
        """Buzina e verifica o estado do pisca do carro."""
        print(f'O carro {self.modelo} está buzinando.')
        print(
            f'e está com o pisca {"ligado" if self.pisca else "desligado"}.\n')

    def toggle_pisca(self) -> None:
        """Liga ou desliga o pisca do carro."""
        self.pisca = not self.pisca
```

No exemplo acima, definimos uma variável no escopo de classe chamada qtd_rodas, que recebe o valor 4. Esse valor será compartilhado entre todas as instâncias da classe.

```python
import classes

if __name__ == '__main__':
    volvo = classes.Carro('Volvo', 'Verde', 2000)
    fusca = classes.Carro('Fusca', 'Azul', 1987)
    print(f'Quantidade de rodas do {volvo.modelo} é {volvo.qtd_rodas}.')
    print(f'Quantidade de rodas do {fusca.modelo} é {fusca.qtd_rodas}.')
```

```
PS C:\Users\augusto.hertzog\GitHub\master-python> python main.py
Quantidade de rodas do Volvo é 4.
Quantidade de rodas do Fusca é 4.
PS C:\Users\augusto.hertzog\GitHub\master-python>
```

Essa propriedade nos permite guardar um valor que queremos manter entre todas as instâncias da classe, como um contador, por exemplo.

```python
class Carro:
    """Classe Carro"""
    total_carros: int = 0

    def __init__(self, modelo: str, cor: str, ano: int) -> None:
        self.modelo: str = modelo
        self.cor: str = cor
        self.ano: int = ano
        self.pisca: bool = False
        Carro.total_carros += 1
```

Neste exemplo, a classe Carro possui um atributo de classe chamado total_carros inicializado como 0. Esse atributo será responsável por contar o número total de carros criados.

Dentro do método __init__(), sempre que um novo objeto Carro é criado, a variável de instância de classe total_carros é incrementada em 1.

```python
from classes import Carro

if __name__ == '__main__':
    volvo: Carro = Carro('Volvo', 'Verde', 2000)
    fusca: Carro = Carro('Fusca', 'Azul', 1987)
    print(f'Total de carros criados {Carro.total_carros}.')
    bmw: Carro = Carro('Brasília Muito Welha', 'Prata', 1950)
    print(f'Total de carros criados {Carro.total_carros}.')
```

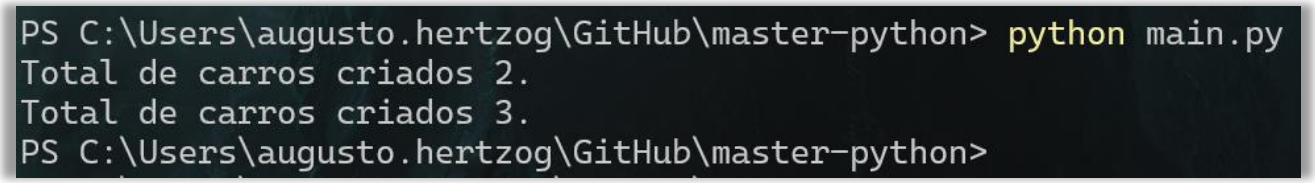
Ao criar os objetos carro1 e carro2, o contador é atualizado para 2. Em seguida, ao criar o objeto carro3, o contador é atualizado para 3.

Os atributos de classe podem ser acessados de duas maneiras:

- Através da classe: Podemos acessar os atributos de classe diretamente usando o nome da classe, seguido do nome do atributo.
- Através de uma instância: Também podemos acessar os atributos de classe por meio de uma instância da classe. Nesse caso, a instância herda os atributos da classe.

```python
from classes import Carro

if __name__ == '__main__':
    volvo: Carro = Carro('Volvo', 'Verde', 2000)
    fusca: Carro = Carro('Fusca', 'Azul', 1987)
    print(f'Total de carros criados {Carro.total_carros}.')
    bmw: Carro = Carro('Brasília Muito Welha', 'Prata', 1950)
    print(f'Total de carros criados {bmw.total_carros}.')
```



Os atributos de instância de classe são úteis para armazenar informações comuns a todas as instâncias, como constantes ou configurações.

Além disso, é importante mencionar que os atributos de classe podem ser alterados por qualquer instância ou pela própria classe. No entanto, quando um atributo de classe é alterado por uma instância, essa alteração é válida apenas para essa instância específica, enquanto as outras instâncias continuarão com o valor original do atributo de classe.

Método Mágico __str__

O método mágico __str__ é um dos métodos especiais em Python que permite definir a representação em formato de string de um objeto de uma classe. Ele é chamado automaticamente quando a função str() é aplicada a um objeto ou quando o objeto é usado como argumento para a função print().

Ao implementar o método __str__ em uma classe, você pode controlar a saída de texto que representa o objeto, personalizando-a de acordo com suas necessidades. Isso permite que você forneça uma descrição mais significativa e legível do objeto quando ele é convertido em uma string.

A assinatura básica do método __str__ é a seguinte:

```python
class Carro:
    """Classe Carro"""

    def __init__(self, modelo: str = '') -> None:
        self.modelo: str = modelo

    def __str__(self) -> str:
        return f'O meu modelo é {self.modelo}.'
```

Dentro do método __str__, você pode construir e retornar uma string que represente o objeto de uma forma desejada. Geralmente, isso envolve a concatenação de valores dos atributos do objeto para criar a representação em formato de string.

```python
from classes import Carro

if __name__ == '__main__':
    fusca: Carro = Carro('Fusca')
    print(fusca)
```

No exemplo acima, o método __str__ é definido na classe Carro. Ele retorna uma string formatada que contém o modelo do carro.

```
PS C:\Users\augusto.hertzog\GitHub\master-python> python main.py
O meu modelo é Fusca.
PS C:\Users\augusto.hertzog\GitHub\master-python>
```

Ao chamar print(fusca), o método __str__ é invocado automaticamente e retorna a representação em formato de string do objeto carro. A saída é "O meu modelo é Fusca.", que foi definida pela implementação do método __str__.

Usar o método __str__ é útil para fornecer uma representação mais legível de um objeto em formato de string, facilitando a depuração, exibição de informações relevantes e comunicação com outros programadores.

É importante notar que o método __str__ é diferente do método __repr__. O método __repr__ também é um método mágico em Python, mas é responsável por retornar uma representação mais formal e detalhada do objeto, geralmente usada para reproduzir o objeto. O __str__ é mais voltado para uma representação amigável ao usuário. Veremos mais sobre o método __repr__ em uma aula futura.

Exercícios para Praticar

1. Crie uma classe Pessoa com um atributo de classe total_pessoas. Incremente esse valor toda vez que uma nova pessoa for criada.
2. Implemente uma classe Circulo com um atributo de classe pi que armazena o valor de π (pi). Use esse valor para calcular a área de diferentes círculos.
3. Crie uma classe Funcionario com um atributo de classe salario_minimo. Defina um método para verificar se o salário de um funcionário é maior ou igual ao salário mínimo.
4. Implemente uma classe ContaBancaria com um atributo de classe taxa_juros. Use esse valor para calcular os juros de diferentes contas bancárias.
5. Crie uma classe Retangulo com um atributo de classe num_lados igual a 4. Use esse valor para verificar se um objeto é de fato um retângulo.
6. Implemente uma classe Estudante com um atributo de classe nota_minima que representa a nota mínima para passar em uma disciplina. Verifique se um estudante passou ou não com base em sua nota.
7. Crie uma classe Produto com um atributo de classe quantidade_total que representa a quantidade total desse produto em estoque. Atualize esse valor toda vez que um novo produto for adicionado ou vendido.
8. Implemente uma classe Veiculo com um atributo de classe velocidade_maxima. Verifique se a velocidade de um veículo está dentro do limite permitido.
9. Crie uma classe Animal com um atributo de classe num_patas que representa o número de patas desse animal. Use esse valor para verificar se um animal é quadrúpede.
10. Implemente uma classe TimeFutebol com um atributo de classe total_jogadores que armazena o número total de jogadores no time. Verifique se um time de futebol tem o número correto de jogadores em campo.
11. Crie uma classe Telefone com um atributo de classe codigo_pais que armazena o código do país. Use esse valor para formatar o número de telefone corretamente.
12. Implemente uma classe Turma com um atributo de classe num_alunos que representa o número de alunos na turma. Verifique se a turma está lotada com base no número de alunos.
13. Crie uma classe Cachorro com um atributo de classe num_patas igual a 4. Use esse valor para verificar se um objeto é de fato um cachorro.
14. Implemente uma classe Pessoa com um atributo de classe maioridade que representa a idade mínima para ser considerado maior de idade. Verifique se uma pessoa é maior de idade com base em sua idade.
15. Crie uma classe Triangulo com um atributo de classe num_lados igual a 3. Use esse valor para verificar se um objeto é de fato um triângulo.
16. Implemente uma classe Empresa com um atributo de classe num_funcionarios que representa o número de funcionários na empresa. Atualize esse valor toda vez que um novo funcionário for contratado ou demitido.
17. Crie uma classe Banco com um atributo de classe taxa_servico que representa a taxa de serviço cobrada pelo banco. Use esse valor para calcular a taxa de serviço de diferentes transações bancárias.
18. Implemente uma classe Cidade com um atributo de classe populacao_total que representa a população total da cidade. Atualize esse valor toda vez que uma nova pessoa se mudar ou sair da cidade.
19. Crie uma classe CestaFrutas com um atributo de classe num_frutas que representa o número de frutas na cesta. Verifique se a cesta está vazia com base no número de frutas.
20. Implemente uma classe Curso com um atributo de classe nota_aprovacao que representa a nota mínima para ser aprovado no curso. Verifique se um aluno foi aprovado ou não com base em sua nota.

21. Crie uma classe Livro com os atributos titulo, autor e ano. Implemente o método __str__ para exibir as informações do livro em formato de string.
22. Implemente uma classe Pessoa com os atributos nome, idade e profissao. Crie o método __str__ para exibir as informações da pessoa de forma legível.
23. Crie uma classe Animal com os atributos nome e especie. Implemente o método __str__ para exibir o nome e a espécie do animal.
24. Implemente uma classe Produto com os atributos nome, preco e quantidade. Crie o método __str__ para exibir as informações do produto, incluindo o valor total em estoque.
25. Crie uma classe Carro com os atributos marca, modelo e ano. Implemente o método __str__ para exibir as informações do carro em formato de string.
26. Implemente uma classe Aluno com os atributos nome, idade e curso. Crie o método __str__ para exibir as informações do aluno de forma formatada.

27. Crie uma classe Retangulo com os atributos largura e altura. Implemente o método __str__ para exibir as dimensões do retângulo em formato de string.
28. Implemente uma classe Ponto com os atributos x e y. Crie o método __str__ para exibir as coordenadas do ponto em formato de string.
29. Crie uma classe Pedido com os atributos id, data e itens. Implemente o método __str__ para exibir as informações do pedido, incluindo os itens.
30. Implemente uma classe Funcionario com os atributos nome, cargo e salario. Crie o método __str__ para exibir as informações do funcionário em formato de string.
31. Crie uma classe Cachorro com os atributos nome, raca e idade. Implemente o método __str__ para exibir as informações do cachorro em formato de string.
32. Implemente uma classe Data com os atributos dia, mes e ano. Crie o método __str__ para exibir a data no formato "dia/mes/ano".
33. Crie uma classe ContaCorrente com os atributos numero, saldo e titular. Implemente o método __str__ para exibir as informações da conta corrente em formato de string.
34. Implemente uma classe Produto com os atributos nome, preco e descricao. Crie o método __str__ para exibir as informações do produto em formato de string.
35. Crie uma classe Cidade com os atributos nome e estado. Implemente o método __str__ para exibir o nome da cidade e o estado em formato de string.
36. Implemente uma classe Computador com os atributos marca, modelo e memoria. Crie o método __str__ para exibir as informações do computador em formato de string.
37. Crie uma classe Triangulo com os atributos lado1, lado2 e lado3. Implemente o método __str__ para exibir as medidas dos lados do triângulo em formato de string.
38. Implemente uma classe Cliente com os atributos nome, endereco e telefone. Crie o método __str__ para exibir as informações do cliente em formato de string.
39. Crie uma classe Professor com os atributos nome, disciplina e sala. Implemente o método __str__ para exibir as informações do professor em formato de string.
40. Implemente uma classe Veiculo com os atributos marca, modelo e ano. Crie o método __str__ para exibir as informações do veículo em formato de string.

41. Crie uma classe Banco com o atributo de classe taxa_juros e o método __str__ para exibir a taxa de juros do banco.
42. Implemente uma classe Loja com o atributo de classe total_vendas que representa o valor total das vendas realizadas pela loja. Crie o método __str__ para exibir o valor total das vendas.
43. Crie uma classe CarrinhoCompras com o atributo de classe total_itens que representa o número total de itens no carrinho. Implemente o método __str__ para exibir o número total de itens.
44. Implemente uma classe Matematica com o atributo de classe pi e o método __str__ para exibir o valor de pi.
45. Crie uma classe Escola com o atributo de classe num_alunos que representa o número total de alunos na escola. Crie o método __str__ para exibir o número total de alunos.
46. Implemente uma classe LojaVirtual com o atributo de classe num_pedidos que representa o número total de pedidos realizados na loja. Crie o método __str__ para exibir o número total de pedidos.
47. Crie uma classe Biblioteca com o atributo de classe livros_disponiveis que representa o número de livros disponíveis na biblioteca. Implemente o método __str__ para exibir o número de livros disponíveis.
48. Implemente uma classe Restaurante com o atributo de classe num_mesas que representa o número total de mesas do restaurante. Crie o método __str__ para exibir o número total de mesas.
49. Crie uma classe Clube com o atributo de classe num_socios que representa o número total de sócios do clube. Implemente o método __str__ para exibir o número total de sócios.
50. Implemente uma classe Universidade com o atributo de classe num_alunos que representa o número total de alunos matriculados na universidade. Crie o método __str__ para exibir o número total de alunos.
51. Crie uma classe Empresa com o atributo de classe num_funcionarios que representa o número total de funcionários na empresa. Implemente o método __str__ para exibir o número total de funcionários.
52. Implemente uma classe Agenda com o atributo de classe num_contatos que representa o número total de contatos na agenda. Crie o método __str__ para exibir o número total de contatos.
53. Crie uma classe Estacionamento com o atributo de classe num_vagas que representa o número total de vagas do estacionamento. Implemente o método __str__ para exibir o número total de vagas.
54. Implemente uma classe Hotel com o atributo de classe num_quartos que representa o número total de quartos do hotel. Crie o método __str__ para exibir o número total de quartos.

55. Crie uma classe Concessionaria com o atributo de classenum_carrosque representa o número total de carros disponíveis na concessionária. Implemente o métodostr` para exibir o número total de carros.
56. Implemente uma classe ClasseVoo com o atributo de classe num_passageiros que representa o número total de passageiros em um voo. Crie o método __str__ para exibir o número total de passageiros.
57. Crie uma classe Curso com o atributo de classe num_alunos que representa o número total de alunos matriculados no curso. Implemente o método __str__ para exibir o número total de alunos.
58. Implemente uma classe Academia com o atributo de classe num_alunos que representa o número total de alunos na academia. Crie o método __str__ para exibir o número total de alunos.
59. Crie uma classe Jogo com o atributo de classe num_jogadores que representa o número total de jogadores no jogo. Implemente o método __str__ para exibir o número total de jogadores.
60. Implemente uma classe TimeFutebol com o atributo de classe num_titulos que representa o número total de títulos conquistados pelo time de futebol. Crie o método __str__ para exibir o número total de títulos.

Exercícios Complexos

61. Crie uma classe LojaVirtual com os atributos de classe total_pedidos e total_vendas, representando o número total de pedidos realizados e o valor total das vendas. Implemente os métodos __init__ para inicializar os atributos e __str__ para exibir as informações da loja.
62. Implemente uma classe Banco com os atributos de classe taxa_juros e saldo_total, representando a taxa de juros aplicada e o saldo total de todas as contas do banco. Crie métodos para registrar transações bancárias e atualizar o saldo total. Implemente o método __str__ para exibir as informações do banco.
63. Crie uma classe Turma com os atributos de classe total_alunos e media_notas, representando o número total de alunos na turma e a média das notas dos alunos. Implemente métodos para adicionar notas dos alunos e atualizar a média. Implemente o método __str__ para exibir as informações da turma.
64. Implemente uma classe Estoque com os atributos de classe total_produtos e valor_total, representando o número total de produtos em estoque e o valor total desses produtos. Crie métodos para adicionar produtos ao estoque e atualizar o valor total. Implemente o método __str__ para exibir as informações do estoque.
65. Crie uma classe Escola com os atributos de classe total_alunos e total_professores, representando o número total de alunos e professores da escola. Implemente métodos para adicionar alunos e professores e atualizar os totais. Implemente o método __str__ para exibir as informações da escola.
66. Implemente uma classe Cinema com os atributos de classe total_salas e total_ingressos, representando o número total de salas de cinema e o número total de ingressos vendidos. Crie métodos para vender ingressos e atualizar o total de ingressos vendidos. Implemente o método __str__ para exibir as informações do cinema.
67. Crie uma classe AgendaContatos com os atributos de classe total_contatos e contatos, representando o número total de contatos na agenda e uma lista de contatos. Implemente métodos para adicionar contatos e atualizar o total. Implemente o método __str__ para exibir as informações da agenda.
68. Implemente uma classe Restaurante com os atributos de classe total_mesas e mesas_ocupadas, representando o número total de mesas do restaurante e o número de mesas atualmente ocupadas. Crie métodos para reservar mesas e atualizar o número de mesas ocupadas. Implemente o método __str__ para exibir as informações do restaurante.
69. Crie uma classe GaleriaArte com os atributos de classe total_obras e obras_vendidas, representando o número total de obras de arte na galeria e o número de obras vendidas. Implemente métodos para adicionar obras e registrar vendas. Implemente o método __str__ para exibir as informações da galeria.
70. Implemente uma classe Academia com os atributos de classe total_alunos e total_instrutores, representando o número total de alunos matriculados na academia e o número total de instrutores. Crie métodos para matricular alunos e contratar instrutores. Implemente o método __str__ para exibir as informações da academia.