Educação Profissional Paulista

Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Lógica de programação e algoritmos

Recursividade

Aula 1

Código da aula: [DADOS]ANO1C3B3S21A1

Mapa da unidade 1 Componente 4

Lógica de programação e algoritmos

semana 10

Prática de estruturas de controle: condições

semana 16

Variáveis e tipos de dados

semana 21

Você está aqui!

Recursividade

semana 23

Busca e ordenação

Lógica de programação e algoritmos

Mapa da unidade 1
Componente 4

Você está aqui!

21

Recursividade

Aula 1

Código da aula: [DADOS]ANO1C3B3S21A1

Objetivos da aula

- Introduzir o conceito de recursão;
- Abordar como esse conceito funciona computacionalmente;
- Diferenciá-lo de alguns outros conceitos, como o de iteração.

Recursos didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Acesso ao laboratório de informática e/ou internet.

Duração da aula

50 minutos.

Competências técnicas

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados.

Competências socioemocionais

- Demonstrar resiliência para lidar com pressões e enfrentar novos desafios, bem como saber lidar com as frustrações, por exemplo, quando um projeto de ciência de dados falhar.

Elaborado especialmente para o curso com apoio da ferramenta Microsoft Copilot.

Primeiras **ideias**

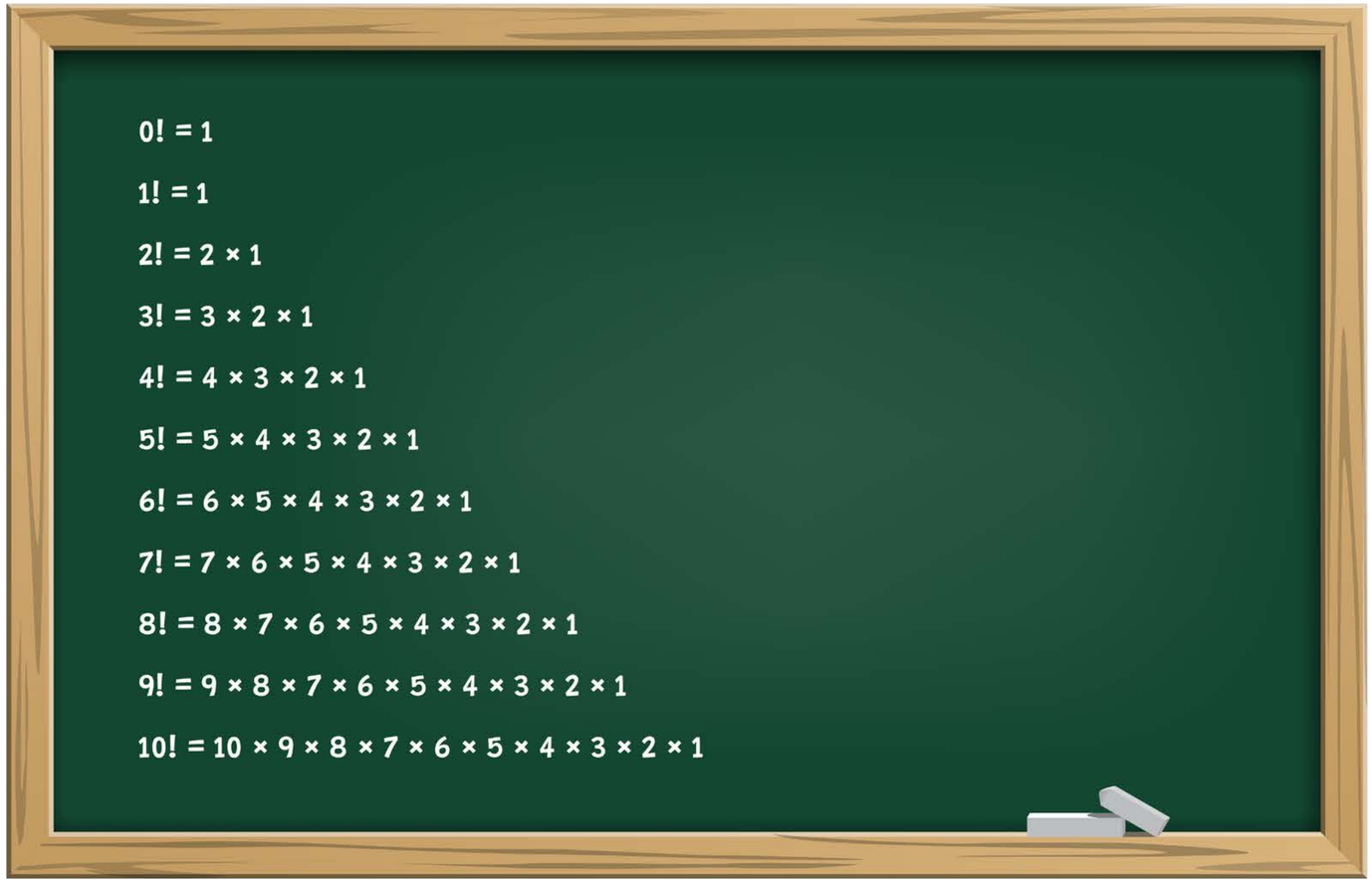
Você conhece as bonecas da imagem anterior?

Quanto trabalho é necessário para tirar, de dentro de umas das outras, todas as bonecas da imagem?

O que aconteceria se a boneca fosse grande o suficiente para comportar 100 bonecas dentro dela?

Ponto de
partida

Identificando situações repetitivas



Construindo o **conceito**

O que é recursão?

- Uma **recursão** ocorre quando uma função “chama” a si mesma para resolver um problema.
- Há uma preocupação, quando falamos de eventos recursivos, sobre a possibilidade de se iniciar um **looping infinito**. Por conta disso, é importante estabelecer uma condição base para terminar a recursão.
- **Recursões são importantes**, porque representam uma forma de resolver problemas de programação complexos de forma mais direta.

Construindo o **conceito**

Estrutura de pilha

- Uma **pilha (stack)** é uma estrutura de dados do tipo LIFO (*Last In, First Out*), na qual o último elemento adicionado é o primeiro a ser resolvido.
- Imagine uma pilha de pratos: o ultimo prato a ser colocado será o primeiro a ser retirado no momento de utilizá-los a mesa.

Pilhas são utilizadas para gerenciar as chamadas de funções e retornos em programas. Quando coloca-se, em uma pilha, mais dados do que ela é capaz de comportar, temos um **estouro de pilha (*stack overflow*)**.

Teremos uma aula própria para esse conteúdo em momento posterior, então não se preocupe em entender agora os mínimos detalhes.

Construindo
o **conceito**

Exemplo de recursão

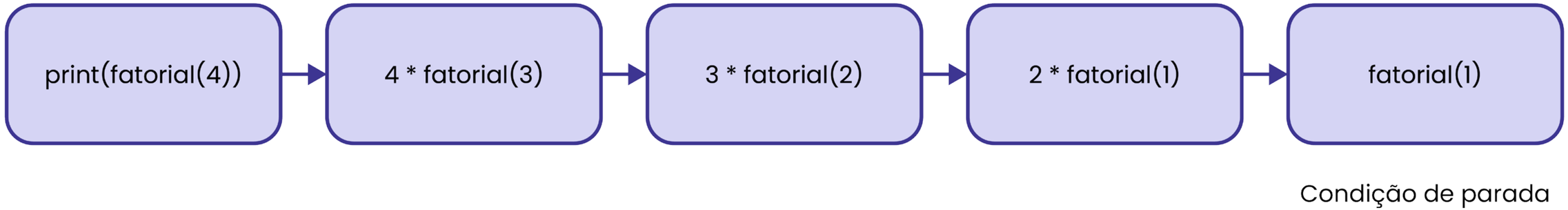
```python
def fatorial(n):
    if n == 1:
        fat = 1
    else:
        fat = n * fatorial(n-1)
    print("calculo do fatorial:", fat)
    return fat

print("resultado:", fatorial(5))
```

```
calculo do fatorial: 1
calculo do fatorial: 2
calculo do fatorial: 6
calculo do fatorial: 24
calculo do fatorial: 120
resultado: 120
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

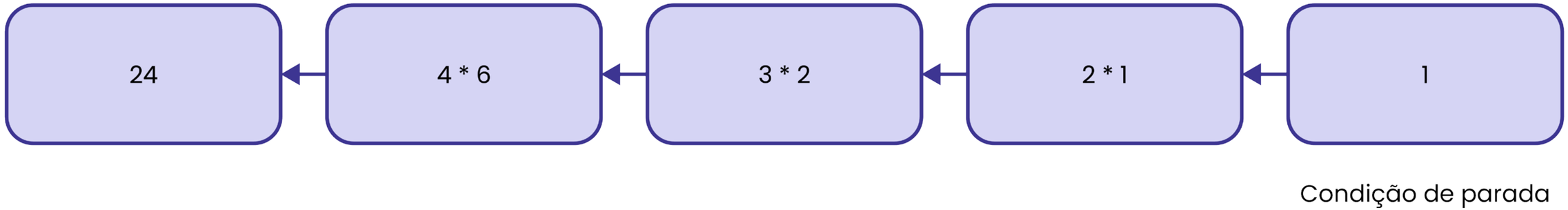
Construindo
o **conceito**

O que aconteceu no programa?



Construindo
o **conceito**

O que aconteceu no programa?



Construindo o **conceito**

Cuidados ao usar recursão

Atenção!

© Getty Images

- **Condição base:** é muito importante definir, corretamente, a condição base para evitar uma recursão infinita.
- **Profundidade de recursão:** existem limitações de memória, que podem ocasionar um *stack overflow*; linguagens têm limites para chamadas recursivas (ex.: o Python tem um limite padrão de 1000 chamadas).
- **Eficiência:** em alguns casos, a recursão pode ser menos eficiente em termos de uso de memória e tempo de execução em comparação com a **iteração**.

Construindo o **conceito**

Recursão *versus* iteração

Recursão

Elegante para problemas naturalmente recursivos (veremos nas próximas aulas).
Pode ser mais intuitiva para certos problemas.

Iteração

Geralmente, mais eficiente em termos de memória e desempenho.
Pode ser mais simples de entender em problemas diretos.

Qual escolher?

A decisão entre recursão ou iteração depende do problema específico, de limitações de linguagem e das restrições de desempenho e do tempo de desenvolvimento do sistema projetado.

Registro

Vamos fazer um **quiz**

O que é essencial para evitar uma recursão infinita?

- Uma condição de parada
- Uma função auxiliar
- Um *loop for*
- Uma chamada de sistema

Vamos fazer um **quiz**

Qual das seguintes afirmações sobre recursão é verdadeira?

- Recursão sempre usa menos memória que iteração
- Todas as funções recursivas podem ser escritas como iterações, de forma simples
- Recursão não pode ser usada em linguagens modernas
- Recursão pode resolver problemas com subproblemas menores

Vamos fazer um **quiz**

Quando a iteração é preferível à recursão?

- Quando o problema não tem uma condição base clara
- Quando estamos trabalhando com problemas de divisão e conquista
- Quando a eficiência de memória e tempo é uma prioridade
- Quando a linguagem de programação não suporta *loops*

O que nós **aprendemos hoje?**

Nuvem de palavras

© Getty Images

O que nós **aprendemos hoje?**

Então ficamos assim...

1. Funções recursivas são funções que “chamam” a si mesmas para solucionar problemas.

2. Ao trabalhar com recursão, é importante ter clareza sobre quando interromper o crescimento da pilha, usando uma condição de parada (condição base).

3. Verificamos que há um paralelo entre recursão e iteração. Em algumas situações, a recursão oferece uma solução mais intuitiva, apesar das potenciais desvantagens em eficiência de memória e tempo.

Saiba mais

Você acredita que o cálculo do MDC (máximo divisor comum) é outro uso na recursão?

Acesse o link a seguir e veja como realizar essa implementação:

ALURA. *Como implementar um algoritmo de cálculo de MDC recursivo em Python?*, 19 abr. 2018. Disponível em: https://cursos.alura.com.br/forum/topico-como-implementar-um-algoritmo-de-calculo-de-mdc-recursivo-em-python-59824. Acesso em: 13 jun. 2024.

Referências da aula

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. *Lógica de programação*: a construção de algoritmos e estruturas de dados com aplicações em Python. Porto Alegre: Bookman, 2022.

Educação Profissional Paulista

Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO