Ciência da Computação

Programação Orientada a Objetos

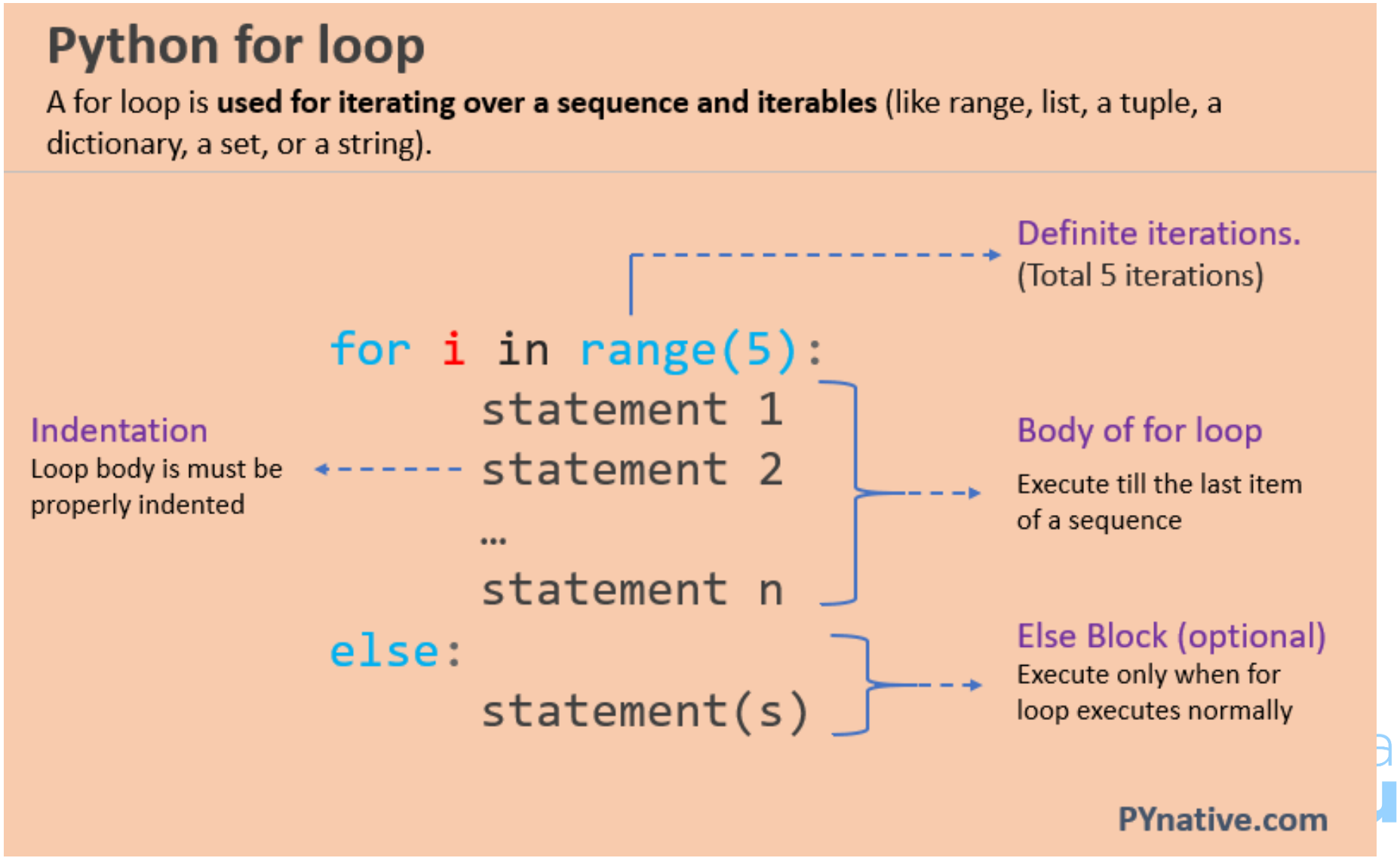
Prof. Luciano Rodrigo Ferretto

Estruturas de Repetição – Instrução **for**

- Parabéns por dominarem a utilização do loop "**for**" em Python! Agora, vamos expandir ainda mais nosso conhecimento e explorar como essa estrutura de repetição é implementada em Java. Embora a ideia central do loop "for" seja similar em ambas as linguagens, existem algumas diferenças importantes que devemos observar.

- Em Python, utilizamos o loop "for" para iterar diretamente sobre elementos em uma sequência, como listas, strings ou até mesmo sequências geradas pela função range(). O Python é conhecido por sua sintaxe clara e legível, que nos permite criar loops de forma muito intuitiva:

Relembrando em Python **for**



Instrução **for each**

- O **for each** em Java é uma estrutura semelhante ao loop for em Python, mas é usada para percorrer coleções de elementos, como arrays (vetores) e listas. Se você já está familiarizado com o loop for em Python, a transição para o **for each** em Java será bastante suave.

```
for (tipo variavel : colecao) {
        sequencia_instruções;
}
```

```
for(tipo variável : coleção)
        instrução;
```

Ciência da Computação

Instrução **for each**

```java
public class ExemploForEach {
    public static void main(String[] args) {
        // Abaixo criamos um Array (vetor) de números inteiros (int)
        int[] numerosInteiros = {1, 3, 5, 7, 2, 4, 6, 8};

        // Esta estrutura interage no vetor, jogando os números
        // para variável "num" em cada loop
        for (int num : numerosInteiros) {
            System.out.println("Número: " + num);
        }
    }
}
```

Ciência da Computação

Instrução **for each**

- O **for each** em Java foi introduzido a partir da versão 5.
- Foi inserido para simplificar a iteração sobre coleções e matrizes, tornando o código mais legível e reduzindo a possibilidade de erros relacionados ao uso de índices.
- O compilador converte esse loop for each em um loop for tradicional que usa um iterador interno da coleção. Isso é feito automaticamente pelo compilador durante o processo de compilação, e o bytecode gerado reflete essa transformação.

```
public static void main(String[] var0) {
    int[] var1 = new int[]{1, 3, 5, 7, 2, 4, 6, 8};
    int[] var2 = var1;
    int var3 = var1.length;

    for(int var4 = 0; var4 < var3; ++var4) {
        int var5 = var2[var4];
        System.out.println("N\u00famero: " + var5);
    }

}
```

Instrução **for i (for tradicional)**

- O **for i** também é chamado de **for tradicional**, pois tem sintaxe muito parecidas nas linguagens mais tradicionais do mercado (C, C++, C#, javascript, PHP)

- A abordagem é um pouco diferente do que já foi visto, mas igualmente poderosa, ou até mais.

- Em Java, o loop **for** tem uma sintaxe mais robusta, permitindo um maior controle sobre as condições de repetição e a manipulação de índices.

Ciência da Computação

Instrução **for i (for tradicional)**

- Geralmente , em uma estrutura **for,** a ***inicialização*** é uma instrução de atribuição que configura o valor inicial da variável de controle de laço, que age como o contador que controla o laço.
- A ***condição*** é uma expressão booleana que determina se o laço será ou não repetido.
- A expressão de ***incremento*** define o valor segundo o qual a variável de controle de laço mudará ao final de cada iteração, ou seja, sempre que o laço for repetido.
- O laço continuará sendo repetido enquanto a *condição* for verdadeira (true)

```
for (inicializacao; condicao; incremento)
{
        //sequencia_instruções;
}
```

```
for (inicialização; condição; incremento)
        //instrução;
```

Ciência da Computação

Aqui temos um exemplo de algoritmo que mostra a raiz quadrada de todos os números de 1 até 100.


```java
public class ExemploForTradicional {
    public static void main(String[] args) {
        /* Nesta estrutura exite a declaração de uma variável "i",
         *     do tipo "int" e é inicializada com o valor 1 (um)
         * A cada loop haverá o incremento de "+ 1" (mais um)
         *     na variável "i"
         * O loop ficará ativo enquanto a variável "i" for menor ou igual
         *     à 100 (100), ou seja, nesse caso o loop será executado
         *     100 vezes*/
        for (int i = 1; i < 100; i++) {
            double raiz = Math.sqrt(i);
            System.out.println("A raiz quadrada de " + i + " é: " + raiz);
        }
    }
}
```

O laço do for pode seguir em sentido **positivo ou negativo**.

Também é possível alterar a variável de controle com **qualquer valor**


```java
/* Abaixo é declarado a variável de controle "i"
 *       e a inicializamos com o valor 100
 * A cada iteração do bloco, ou seja, cada loop, haverá o
 *       decremento de "- 5" (menos 5) na var. de controle
 * O loop ficará ativo enquanto a vairável "i" for maior ou igual a 1
 *
 */
for (int i = 100; i >= 1; i -= 5) {
    double raiz = Math.sqrt(i);
    System.out.println("A raiz quadrada de " + i + " é: " + raiz);
}
```

Utilizando mais de uma variável de controle.

Para isso é utilizado a vírgula (,) para separar as variáveis e expressões.


```
/*Neste exemplo estamos declarando duas variáveis de controle que nomeamos de "i" e "j"
        e as inicializamos com os valores 1 (um) e 10 (dez) respectivamente
  A cada iteração do bloco, ou seja, a cada loop haverá o incremento de "+ 1" (mais um)
        na variável de controle "i" e um decremento de "- 1" (menos um) na variável
        de controle "j"
  O loop ficará ativo enquanto a variável "i" for menor ou igual à "j",
        ou seja, o loop será executado 5 (cinco) vezes
*/
for (int i = 0, j = 10; i <= j; i++, j--) {
    System.out.println("Valores de i e j: " + i + " " + j);
}
```

A condição **não** necessariamente precisa estar relacionada a variável de controle

```
Scanner scan = new Scanner(System.in);
System.out.println("Pressione 'S' para SAIR.");
/*Neste exemplo estamos declarando uma variável de controle que nomeamos de "i"
        e a inicializamos com o valor 1 (um)
  A cada iteração do bloco, ou seja, a cada loop haverá o incremento de "+ 1" (mais um)
        na variável de controle "i"
  Na condição do For definimos que será solicitado uma entrada via Teclado (Scanner + System.in)
        essa entrada será um retorno do tipo String (retorno do método "next()"),
        e desse retorno estrairemos o caracter contido no índice 0 dessa String (charAt(0))
      O loop ficará ativo enquanto esse caracter extraído for diferente de 'S',
        observe que o Java é Case-Sensitive
*/

for (int i = 1; scan.next().charAt(0) != 'S'; i++) {
    System.out.println("Já passou "+ i + " vezes");
}
scan.close();
```

Partes Faltantes

É possível omitir qualquer uma das partes do for se não forem necessárias.

Você pode deixar faltando e fazer o controle fora dos parênteses.

```
/*Neste exemplo NÃO temos nenhuma vairável de controle específica da estrutura do for,
        temos na verdade uma variável "i" que foi declarada e iniciada
        dentro do escopo do método main()
  Também não temos nenhum tipo de iteração (incremento) dentro da estrutura do for.
        a variável que estamos utilizando na condição está sendo alterada no corpo de instruções
        do for
  O loop ficará ativo enquanto a variável "i" for menor que 3 (três),
        ou seja, o loop será executado 3 (três) vezes já que essa variável foi
        iniciada em 0 (zero)
*/
int i = 0;
for (; i < 3;) {
    System.out.println("i: "+ i);
    i++;
}
```

Laço Infinito: Você pode criar um laço infinito usando o for se deixar a expressão condicional vazia. Para sair de um laço infinito você pode utilizar a instrução **break;**

Também temos a instrução **continue** para pular para a próxima iteração.

```
Scanner scan = new Scanner(System.in);
System.out.println("""
        Pressione 'S' para SAIR.
        Pressione 'C' para NÃO contar os loops.
        """);
/*Neste exemplo NÃO estamos declarando nenhuma de controle, nem condição e nem incremento
  Possuímos apenas uma variável no escopo do método main() para servir de contador.
  No corpo de instruções do For definimos que será solicitado uma entrada
            via Teclado (Scanner + System.in), essa entrada será um retorno
            do tipo String (retorno do método "next()"),
            e desse retorno estrairemos o caracter contido no índice 0 dessa String (charAt(0))
        Quando esse caracter extraído for igual à 'S',
            o loop será "quebrado" (a execução sairá do loop) através da instrução "break"
        Quando esse caracter extraído for igual à 'C',
            o loop "pulará" para próxima iteração, ou seja, o que está abaixo desta linha
            não será executado na iteração corrente.
*/
int i = 1;
for (;;) {
    char caracter = scan.next().charAt(0);
    if (caracter == 'S')
        break;
    if (caracter == 'C')
        continue;
    System.out.println("Já passou "+ i++ + " vezes");
}
scan.close();
```

Laço Sem Corpo.

Em Java, o bloco de instruções de um laço for (ou qualquer outro laço) pode estar vazio. A *Instrução nula* é sintaticamente válida.

```
/*Neste exemplo o for possui uma variável de controle declarada com o nome "i"
 *      e iniciada com o valor 1 (um)
 * O loop será executado enquanto i for menor e igual à 5 (cinco)
 * A cada iteração, i será incrementado em "+ 1" (mais um)
 *      e a variável do método main() "sum" terá o seu valor corrente
 *      acrescido pelo valor corrente da variável "i"
 *
 * O loop não possui instruções em seu corpo
 */
int sum = 5;
for (int i = 1; i <= 5; sum += ++i) ;
System.out.println("O valor da variável sum é: " + sum);
```

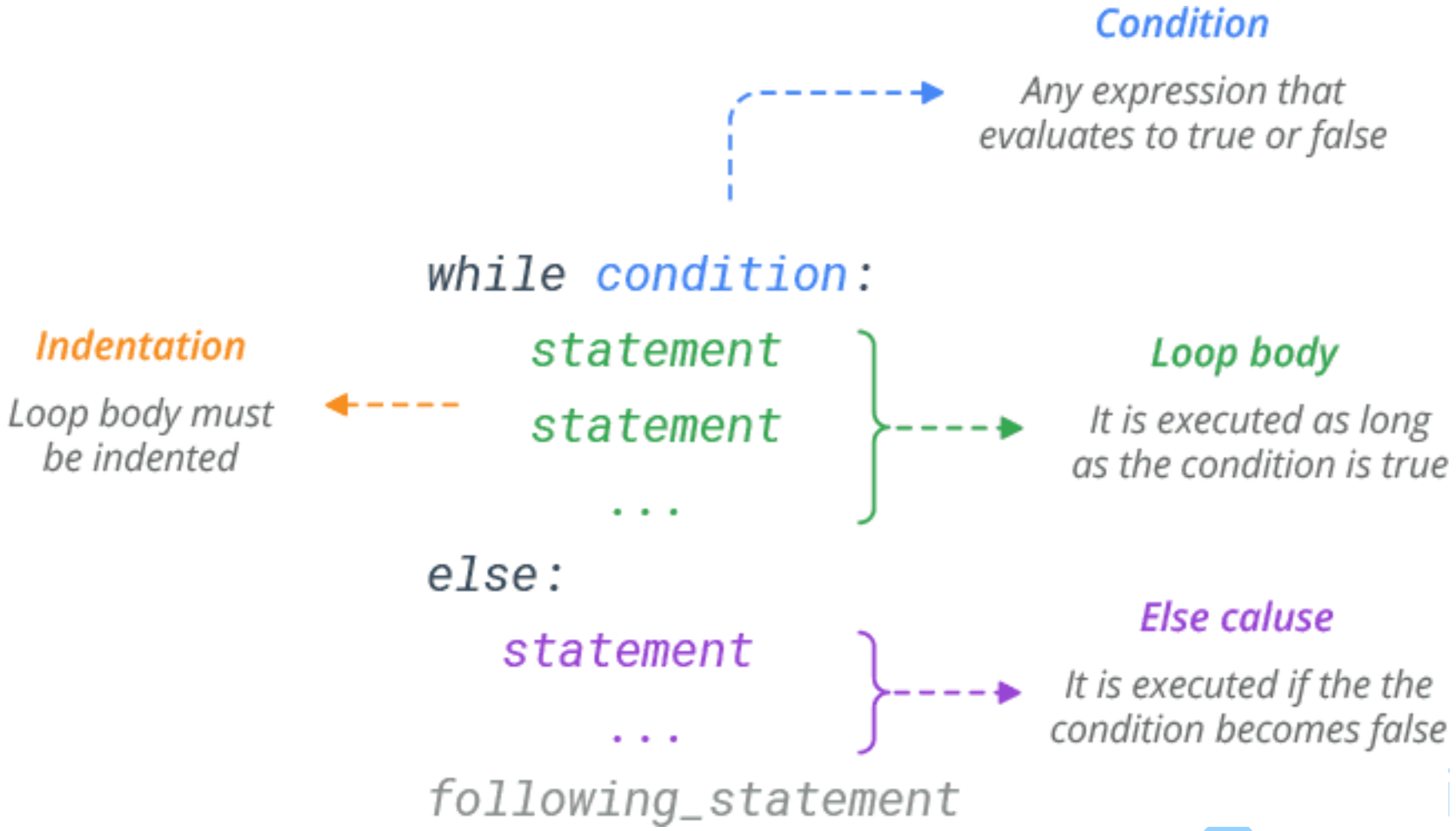
Estruturas de Repetição – Instrução **While**

- A estrutura de repetição **while** é utilizada para executar um bloco de código repetidamente enquanto uma condição específica for verdadeira.

- Essa estrutura é ideal quando o número de repetições não é conhecido antecipadamente ou quando se deseja que o bloco de código seja executado apenas se a condição for verdadeira desde o início.

Ciência da Computação

Relembrando em Python **While**



ia da
Computação

Instrução **While** em Java


```
While (condicao) {
        sequencia_instruções;
}
```

```
while (condicao)
        instrução;
```

Ciência da Computação

Instrução **While** em Java


```java
Scanner scan = new Scanner(System.in);
/*Neste exemplo a execução permanece em loop enquanto
      a variável opcao contiver o valor 'S' */
char opcao = 'S';
while (opcao == 'S') {
    System.out.println("Digite 'S' se deseja permanecer aqui?");
    opcao = scan.next().charAt(0);
}
scan.close();
```

Ciência da Computação

Estruturas de Repetição – Instrução **Do-While**

- A estrutura de repetição **do-while** é semelhante ao **while**, mas garante que o bloco de código seja executado pelo menos uma vez, mesmo que a condição seja falsa desde o início.

```
do {
        sequencia_instruções;
} while (condicao);

do
        instrução;
while (condicao) ;
```

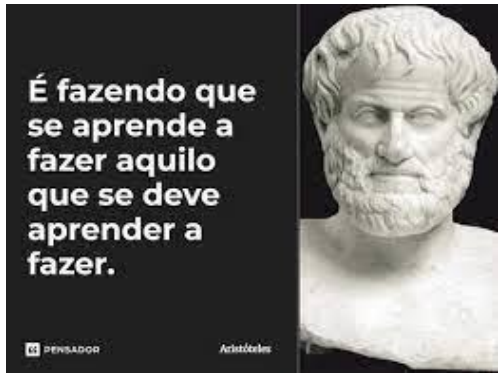
Ciência da Computação

Instrução **do-while** em Java


```java
Scanner scan = new Scanner(System.in);
/*Neste exemplo a execução permanece em loop enquanto a variável opcao contiver o valor 'S'
* a grande diferença para a estrutura while tradicional é que a variável opcao não precisa
* possuir o valor 'S' no início;
*/
char opcao;
do {
    System.out.println("Digite 'S' se deseja permanecer aqui?");
    opcao = scan.next().charAt(0);
} while (opcao == 'S');
scan.close();
```

Ciência da Computação

Vamos para a prática



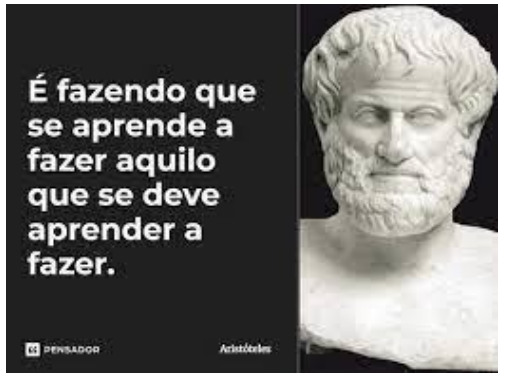
Jogo de Adivinhação

Você deve desenvolver um jogo de adivinhação em Java. O jogo deve funcionar da seguinte forma:

1) O computador irá escolher aleatoriamente um número entre 1 e 100 (inclusive).
2) O jogador terá um número limitado de tentativas para adivinhar o número escolhido pelo computador.
3) Após cada tentativa do jogador, o programa deverá informar se o número escolhido pelo jogador é maior, menor ou igual ao número escolhido pelo computador.
4) O jogo deve continuar até que o jogador adivinhe corretamente o número ou até que ele esgote todas as suas tentativas.
5) Ao final do jogo, o programa deve informar ao jogador se ele adivinhou corretamente o número ou não, e quantas tentativas foram necessárias.

Ciência da Computação

Vamos para a prática



Jogo de Adivinhação

O programa deve ser implementado em Java e deve fazer uso das seguintes estruturas:

1) Estruturas condicionais (if-else, switch-case ou operador ternário) para comparar os números e informar ao jogador se ele está “quente” ou “frio”.

2) Estruturas de repetição (for, while ou do-while) para permitir que o jogador realize múltiplas tentativas até adivinhar corretamente o número ou esgotar suas tentativas.

Ciência da Computação