LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Blocos de Código

- Blocos de código são delimitados pelo uso de indentação
- A indentação deve ser constante no bloco de código
- É uma boa prática não misturar tabulação com espaços
- Usar quatro espaços para indentação é uma convenção amplamente aceita, além de ser uma recomendação oficial (http://python.org/dev/peps/pep-0008)

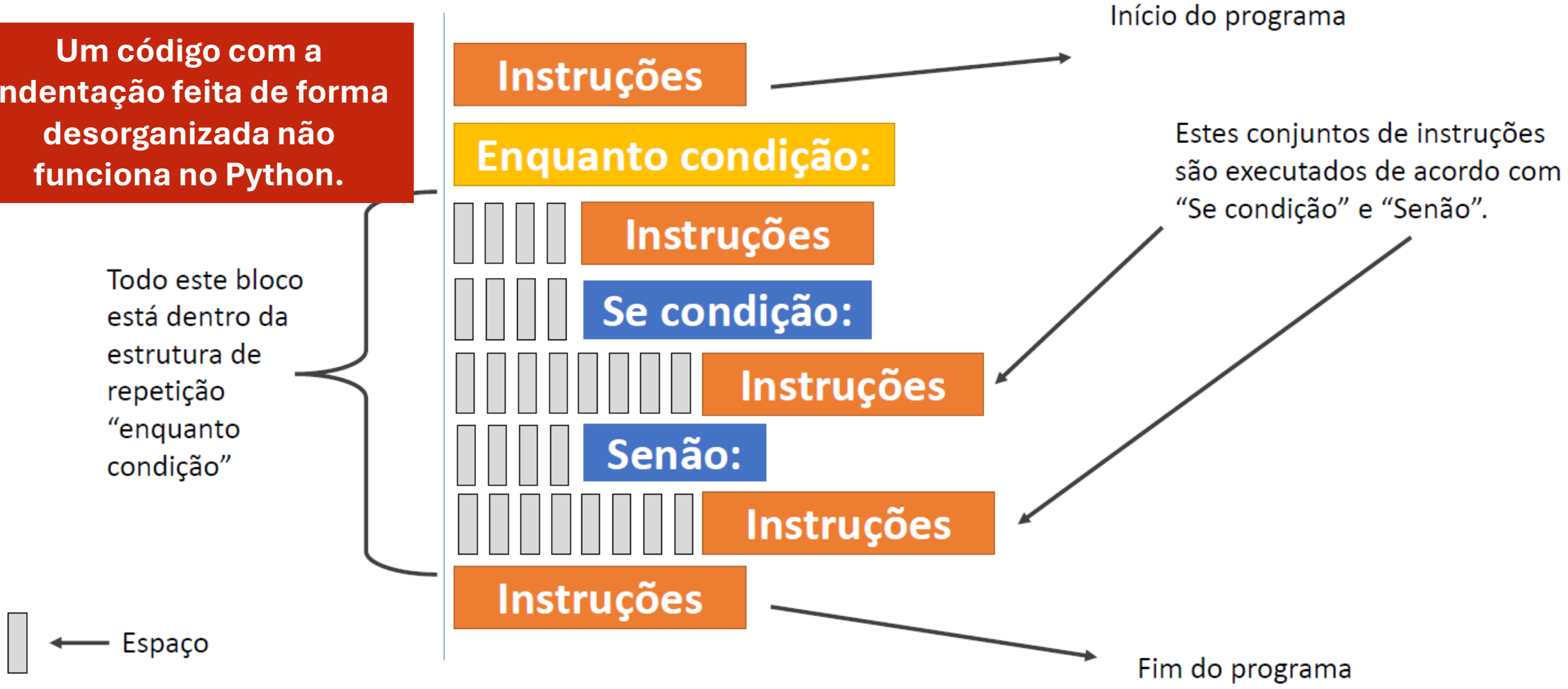
LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Um código com a indentação feita de forma desorganizada não funciona no Python.



LÓGICA DE PROGRAMAÇÃO

ESTRUTURA DE DECISÃO

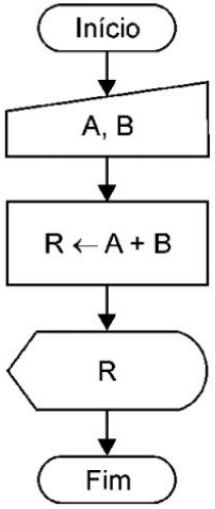


Diagrama de blocos

Diagrama de quadros

Estrutura de operação computacional de ***sequência***.

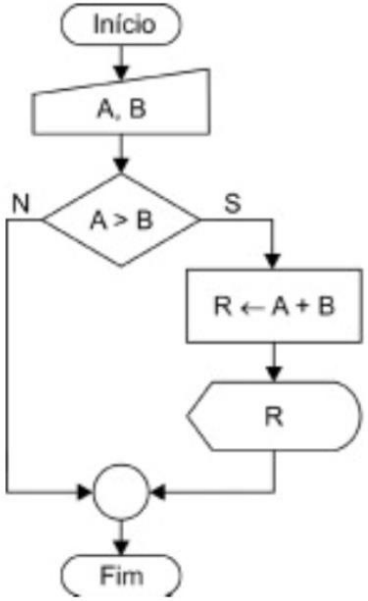


Diagrama de blocos

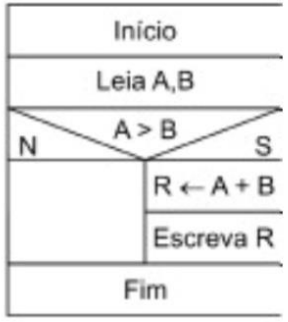


Diagrama de quadros

Estrutura de operação computacional de ***decisão simples***

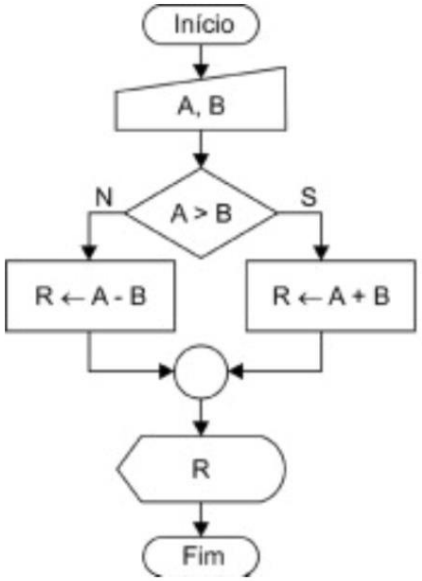


Diagrama de blocos

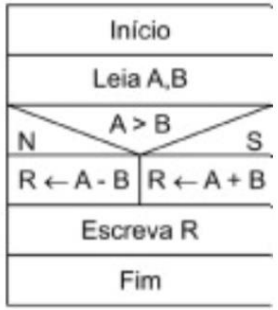


Diagrama de quadros

Estrutura de operação computacional de ***decisão composta***

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Estrutura de decisão ou condições.

Executar ou não executar? Eis a questão...

Nem sempre todas as linhas dos programas serão executadas. Muitas vezes, será mais interessante ***decidir*** que ***partes do programa devem ser executadas*** com base no resultado de uma ***condição***. A base dessas decisões consistirá em expressões lógicas que permitam representar escolhas em programas.

As condições servem para selecionar quando ***uma parte do programa deve ser ativada*** e quando ***deve ser simplesmente ignorada***.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Estrutura de decisão ou condições.

Em Python a estrutura de decisão é o ***if*** *(se)*.

```
if condição:
    bloco verdadeiro
```

Python é uma das poucas linguagens de programação que utiliza o ***deslocamento do texto à direita (recuo)*** para marcar o ***início e o fim*** de um ***bloco de instrução***. Os programadores ***Pythonicos***, gostam e utilizar a palavra ***Suíte*** em vez de ***Bloco***. Outras linguagens contam com palavras especiais para isso, como ***BEGIN*** e ***END***, em Pascal; ou as famosas chaves **{** e **}** em C e Java.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Estrutura de decisão ou condições.

Para que serve os *(dois pontos)* ***em inglês(colon)*** **:**

```
if condição:
    bloco verdadeiro
```

Os dois-pontos (**:**) são importantes, no sentido de que introduzem um novo **bloco/*suíte de código*** que deve ser ***recuado à direita***. Se você ***esquecer de recuar o código depois dos dois-pontos, o interpretador gerará um erro***.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Estrutura de decisão ou condições.

O ***if*** nada mais é que nosso ***"se"***. Poderemos então entendê-lo em português da seguinte forma: ***se a condição for verdadeira, faça alguma coisa.***

```
#estrutura de decisão em Python
idade = int(input("Entre com sua idade"))
if idade >= 18:
   print("Posso tirar habilitação para dirigir")
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Estrutura de decisão ou condições.

No Python temos a cláusula ***else*** *(se não)* para especificar o que fazer caso o resultado da avaliação do condição ***if*** seja ***falso***.

```
#estrutura de decisão em Python
idade = int(input("Entre com sua idade"))
if idade >= 18:
   print("Posso tirar habilitação para dirigir")
else:
   print("Não posso dirigir")
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Estruturas aninhadas de decisão ou condições com *if, elif*

Nem sempre nossos programas serão tão simples. Muitas vezes precisaremos aninhar vários ***if*** para obter o comportamento desejado do programa.

```
#estruturas aninhadas de decisão em Python
#if, else
idade = int(input("Qual a sua idade?"))
if idade >= 18:
    habilitado = input("Tem habilitação? Sim ou Não")
    if habilitado == "Não":
        print("Posso tirar habilitação para dirigir.")
    else:
        if habilitado == "Sim":
            print("Você já pode dirigir.")
else:
    print("Sou menor de idade, não posso dirigir.")
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

elif

Python apresenta uma solução muito interessante ao problema de ***múltiplos ifs aninhados***, a clausula ***elif*** substitui um par de ***else if***.

```
#estruturas aninhadas de decisão em Python
#if, else, elif

idade = int(input("Qual a sua idade?"))
if idade >= 18:
   habilitado = input("Tem habilitação? Sim ou Não")
   if habilitado == "Não":
      print("Posso tirar habilitação para dirigir.")
   elif habilitado == "Sim":
      print("Você já pode dirigir.")
else:
   print("Sou menor de idade, não posso dirigir.")
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Estrutura de decisão ou condições.

Os operadores ***relacionais*** e operadores ***lógicos*** podem e devem ser utilizados com a ***estrutura de decisão if***.

```
if 10 >= 20 and 20 < 10 or 20 > 10:
    bloco verdadeiro
```

Os ***operadores lógicos*** podem ser combinados em expressões lógicas mais complexas.
Quando tiver mais de um operador lógico, avalia-se o ***not*** primeiro, ***and*** em segundo e ***or*** em terceiro, na dúvida utilize parênteses ***( )*** para modificar a precedência.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Match Case.

O bloco **match case** é uma estrutura de controle de fluxo que permite comparar uma variável com diferentes valores ou padrões de maneira **mais organizada e legível** do que as tradicionais estruturas *if/elif/else*.

Introduzido no Python 3.10 (atual 3.12), ele é particularmente útil quando há muitos valores específicos a serem verificados.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Match Case.

Imagine que você tem uma variável e quer executar diferentes ações dependendo do valor dessa variável. Antes do **match case**, você provavelmente usaria uma série de **if, elif e else.**

```
#matchCase
nome = "Lucas"

match nome:
    case "Lucas":
        print("Olá, Lucas!")
    case "Maria":
        print("Olá, Maria!")
    case _:
        print("Não sei quem é você!")
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Qual utilizar? If, else, elif, match case.

Use **if, elif, else** para condições booleanas, operações lógicas e onde as verificações não são simples igualdades.

Use **match case** para múltiplas alternativas de comparação de um único valor.

Em termos de desempenho, a diferença será geralmente **insignificante,** o **match case** tende a ser um pouco mais rápido para muitas comparações simples, mas o ganho é pequeno.

Priorize a clareza do código ao escolher entre os dois.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

If else elif

```
valor = 20

if valor > 10 and valor < 30:
    print("Valor está entre 10 e 30.")
elif valor == 30:
    print("Valor é exatamente 30.")
else:
    print("Valor está fora do intervalo.")
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Match Case.

```
comando = "parar"

match comando:
    case "iniciar":
        print("Iniciando o sistema...")
    case "parar":
        print("Sistema parando.")
    case "reiniciar":
        print("Reiniciando o sistema.")
    case _:
        print("Comando desconhecido.")
```

VAMOS PRATICAR.

EXEMPLOS E ATIVIDADES DE ESTRUTURAS DE CONDIÇÃO