Operadores e Estruturas de Decisões

Disciplina: Programação para Engenharia

Professores: Cristiane Pavei Martinello Fernandes
Douglas de Medeiros Deolindo
Giovani Martins Cascaes
Joel Barbosa Panchyniak
Marcelo Marcos Amoroso
Marcos Antônio Jeremias Coelho
Ramon de Souza Coan

Praticando com o Python...

1) Crie um programa que imprima o comprimento de uma string fornecida pelo usuário.

2) Escreva um programa que leia uma frase e converta em uma outra variável a cadeia de caracteres de letras maiúsculas em letras minúsculas.

3) Elabore um programa que solicite uma frase ao usuário e escreva a frase toda em maiúscula. No mesmo programa exiba a frase sem espaços em branco.

4) Desenvolva um programa que solicite uma frase ao usuário e escreva a frase invertida.

Praticando com o Python...

5) Crie um programa que leia o nome completo de uma pessoa todo em minúsculo e exiba este nome com as primeiras letras em maiúsculo.

6) Seguindo o exercício acima, exiba da mesma forma, no entanto a entrada será com todos os caracteres maiúsculos.

7) Elabore um programa que leia o nome do usuário e mostre o nome de traz para frente, utilizando somente letras maiúsculas.

8) Desenvolva um programa que leia uma frase e um caractere. Em seguida, exiba ambos e o número de ocorrências do caractere na frase.

Praticando com o Python...

9) Elabore um programa que leia uma frase, uma palavra antiga e uma palavra nova. O programa deve exibir uma string contendo a frase original e outra com a ocorrência da palavra antiga substituída pela palavra nova.

Exemplo:

Frase: “Quem parte e reparte fica com a maior parte”
Palavra antiga: “parte”
Palavra nova: “parcela”
Resultado a ser impresso no programa : “Quem parcela e reparcela fica com a maior parcela”

O que são operadores?

Operadores são responsáveis por diferentes operações, a depender do tipo de dado utilizado. Os operadores em Python:

- **Aritméticos** (usados em expressões aritméticas - cálculos)
- **Relacionais** (usados em comparações numéricas)
- **Lógicos** (usados em comparações lógicas)
- **de Atribuição** (armazenamento de valores em variáveis)

Operadores de atribuição

Os operadores de atribuição são responsáveis por alocar valores em variáveis.

- Variável do lado esquerdo, valor ou expressão do lado direito

```
>>> x = 0
```

- Pode-se atribuir valor a várias varáveis ao mesmo tempo

```
>>> x = y = 2
```

Operadores de atribuição

Pode-se também atribuir valores diferentes para variáveis distintas em uma mesma instrução

```
>>> x,y = 3, 5
>>> x
3
>>> y
5
```

Operadores de atribuição

No Python, pode-se também atribuir (trocar) valores em variáveis, sem utilizar uma variável auxiliar.

```
a = 5
b = 25
print("Antes da troca")
print("a:", a, "b:", b)

a, b = b, a          #Troca

print("Após a troca")
print("a:", a, "b:", b)
```

Operadores de Atribuição

Operador	Descrição	Exemplo
=	Atribuição	p = 4
+=	Adição e Atribuição	p += 2
-=	Subtração e Atribuição	p -= 5
*=	Multiplicação e Atribuição	p *= 18
/=	Divisão e Atribuição	p /= 2
//=	Divisão Inteira e Atribuição	p //= 4
%=	Resto da divisão e Atribuição	p %= 3
**=	Potenciação e Atribuição	p **= 20

```
var op= constante ou expressão      x += 10      #x = x + 10
```

Operadores Aritméticos

Os operadores aritméticos são responsáveis por realizar as operações matemáticas básicas.

Operador	Descrição	Exemplo
*	Multiplicação	P * 20
/	Divisão	x / y
//	Divisão Inteira	a // b
%	Resto da divisão	5 / 2
+	Adição	f + u
-	Subtração	h - 3

Prioridade dos operadores Aritméticos

Assim como na matemática, operadores seguem uma ordem de precedência. Naturalmente, é possível realizar operações na ordem desejada com o uso de parênteses.

Operador	Descrição	Prioridade
(x)	Expressão entre ()	1
**	exponenciação	2
- x	Operador unário	3
*, /, //, %	Multiplicação e divisão	4
+, -	Adição e Subtração	5

Operadores Relacionais

Os operadores Relacionais avaliam expressões e retornam verdadeiro ou falso, assim como os operadores Lógicos. No entanto, os relacionais são responsáveis por operações de comparação de magnitude.

Operador	Descrição	Exemplo
<	Menor que	P < 20
<=	Menor ou igual a	x <= y
>	Maior que	a > b
>=	Maior ou igual a	5 >= 2
==	Igualdade	f == u
!=	Diferente de	h != 3

Operadores Relacionais

```
>>> a = 1
>>> b = 1
>>> a == b
True
>>> a <= b
True
```

```
>>> a != b
False
```

```
>>> 2 >= -1
True
```

Operadores Lógicos

Os operadores **Lógicos** são muito utilizados em estruturas condicionais e estruturas de repetições. Esses operadores avaliam expressões lógicas e retornam valores verdadeiro ou falso. Em Python: **True** e **False**

Operador	Descrição	Exemplo
and	E lógico	x >3 and x < 10
or	OU lógico	x <3 or x > 10
not	Negação	not a

Operadores Lógicos – Tabela verdade

O Python oferece três operadores lógicos (booleanos): **or** (ou), **and** (e) e **not** (não). A tabela verdade abaixo apresenta os valores retornados pelos operadores para diferentes valores dos operandos.

a	b	not a	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Operadores Lógicos - Exemplos

```
>>> nome1 = "Maria"
>>> nome2 = "Joao"
>>> (nome1 == "Maria") and (nome1 != nome2)
True
>>> a = 1
>>> b = 1
>>> c = 3
>>> a == b
True
>>> b == c
False
>>> (a==b) and (b==c)
False
>>> (a==b) or (b==c)
True
```

Biblioteca **math**

Este módulo fornece acesso a um conjunto de funções matemáticas.

Para utilizá-las no Python inclua, no início do script: **`import math`**

Método	Descrição	Exemplo
math.ceil(x)	Arredonda para cima	math.ceil(5.3) → 6
math.copysign(x, y)	Obtém um float com o valor absoluto de x, mas com o sinal de y	math.copysign(-5.3, 1) → 5.3
math.fabs(x)	Valor absoluto de x	math.fabs(-5.3) → 5.3
math.floor(expr)	Arredonda para baixo	math.floor(5.3) → 5
math.fmod(x, y)	Resto da divisão de x por y (usar quando x ou y forem float, caso contrário usar %)	math.fmod(5.4, 2) → 1.4
math.trunc(x)	Parte inteira de x	math.trunc(5.6) → 5

Biblioteca **math**

Este módulo fornece acesso a um conjunto de funções matemáticas.

Método	Descrição	Exemplo
math.exp(x)	e**x	math.exp(2) → 7.38905609893065
math.log(x)	Logaritmo natural de x (base e)	math.log(2) → 0.693147180559453
math.log(x, y)	Logaritmo de x na base y	math.log(2, 10) → 0.301029995663981 14
math.pow(x, y)	x**y	math.pow(2, 3) → 8.0
math.sqrt(x)	Raiz quadrada de x	math.sqrt(16) → 4.0

Biblioteca **math**

Este módulo fornece acesso a um conjunto de funções matemáticas.

Função	Descrição	Exemplo
math.sin(x)	Seno	math.sin(0) → 0.0
math.asin(x)	Arco seno	math.asin(1) → 1.5707963267948966
math.cos(x)	Cosseno	math.cos(0) → 1.0
math.acos(x)	Arco cosseno	math.acos(-1) → 3.141592653589793
math.tan(x)	Tangente	math.tan(1) → 1.557407724654902 3
math.atan(x)	Arco tangente	math.atan(1) → 0.7853981633974483
math.degrees(x)	Converte radianos para graus	math.degrees(math.pi) → 180.0
math.radians(x)	Converte graus para radianos	math.radians(180) → 3.141592653589793

Biblioteca **random**

Este módulo implementa geradores de números pseudoaleatórios. Geração de número aleatórios inteiros,

```
random.randint(inicio, fim)
```

inicio: indica o valor inicial, que faz parte dos resultados possíveis

fim: representa o ponto de parada, no qual o valor indicado também pode ser selecionado como retorno

```
#random.seed()
```

Praticando Python...

1) Construa um programa que tem como dados de entrada dois pontos quaisquer no plano cartesiano: P1 e P2. Considere que P1 é definido pelas coordenadas x1 e y1, enquanto P2 por x2 e y2 . O programa deve calcular e escrever a distância entre os pontos P1 e P2. A fórmula que calcula a distância entre os dois pontos é dada por:

A função que calcula a raiz quadrada é a **sqrt()** (**sq**uare **r**oo**t**), veja **pow()**

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Praticando Python...

2) Uma imobiliária paga aos seus corretores um salário base de R$ 1.500,00. Além disso, uma comissão de R$ 200,00 por cada imóvel vendido e 5% do valor de cada venda. Construa um programa que solicite o nome do corretor, a quantidade de imóveis vendidos e o valor total de suas vendas. Ao fim, o programa deve calcular e escrever o salário final do corretor de imóveis.

3) Elabore um código fonte que calcule a hipotenusa de um triângulo retângulo, cujos catetos serão fornecidos pelo usuário.

Controle de Fluxo

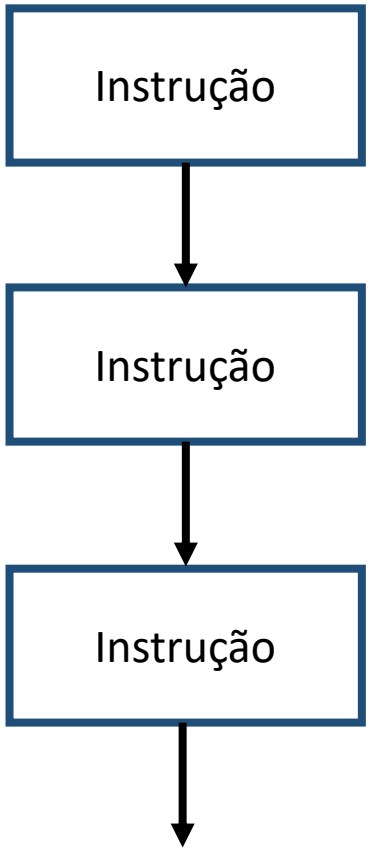
O controle de fluxo é o controle da sequência de comandos executados pela rotina.

Um código é sempre executado linha por linha, na ordem que as encontram.

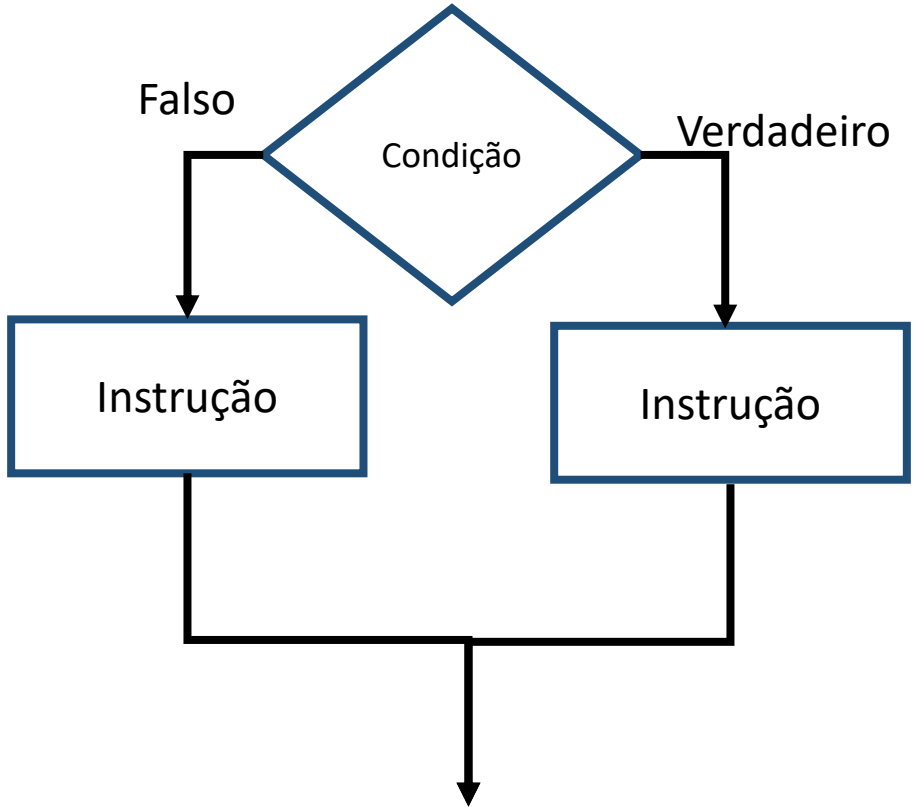
Desta forma, se existem somente procedimentos simples, o funcionamento da rotina é linear.

Controle de Fluxo

Sequência



Condição



Controle de Fluxo - Estrutura

No Python, o início de uma dessas estruturas é indicada pelas palavras reservadas e o uso de dois pontos.

Os procedimentos subordinados às estruturas devem estar indentados com um recuo de tab em referência a linha que indica o início da estrutura.

A própria quebra de indentação indicará o fim da estrutura.

Controle de Fluxo - Estrutura

Sendo assim, uma estrutura deve-se parecer com:

```
ESTRUTURA   <expressao>   :
      procedimento1
      procedimento2
      ...
```

Estruturas Condicionais em Python

O primeiro conjunto de estruturas de controle de fluxo são as estruturas condicionais.

As responsáveis por tomadas de decisão, que indicarão alguns procedimentos ou outros, dependendo da condição avaliada.

As mais populares estruturas condicionais são if else e switch. No Python, a estrutura switch é representada por match case.

Estruturas Condicionais – Exemplo

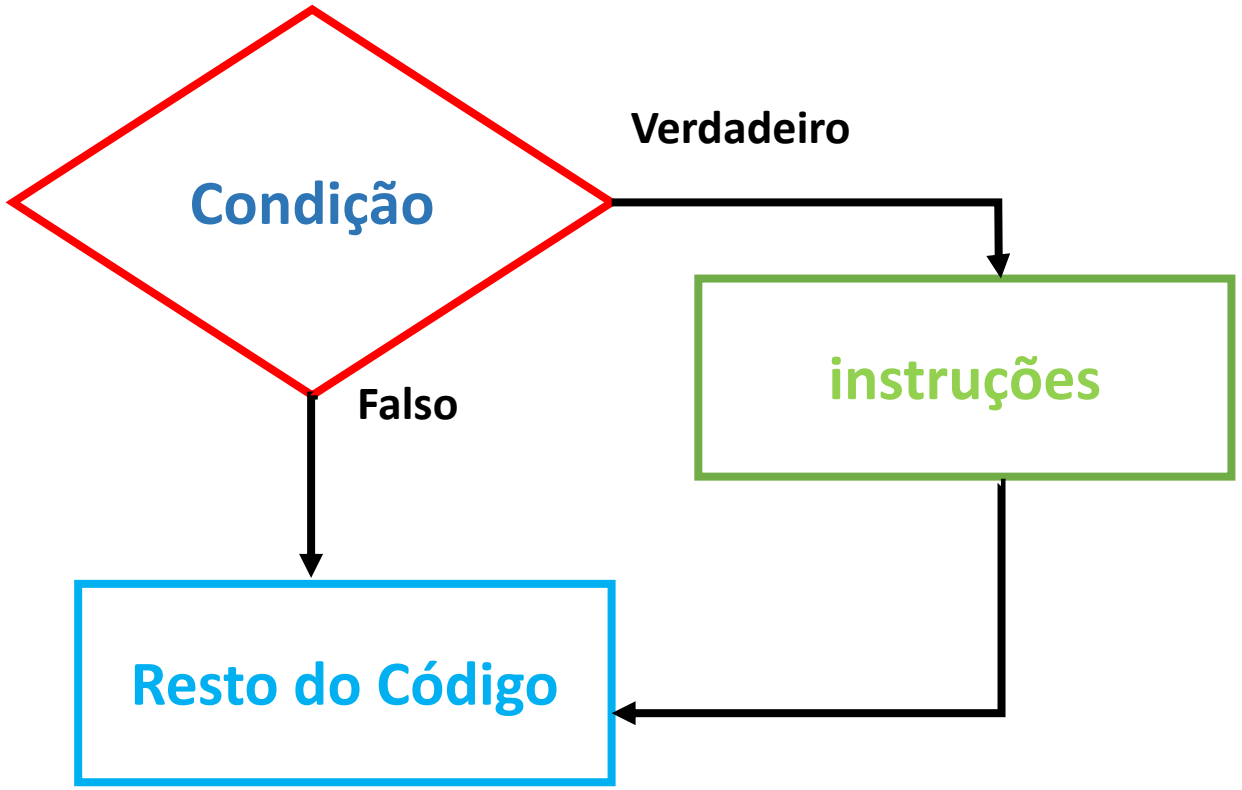
Em Python temos as seguintes estruturas de decisão:

- **`if`** `(se)`
- **`if..else`** `(se..senão)`
- **`if..elif..else`** `(se..senão se..senão)`

Estruturas Condicionais – if

A estrutura if avalia uma expressão, caso a expressão seja verdadeira, os procedimentos subordinados são executados.

```
if <condição>:
        instruções
        instruções
        instruções

resto do código
```



Estruturas Condicionais – if

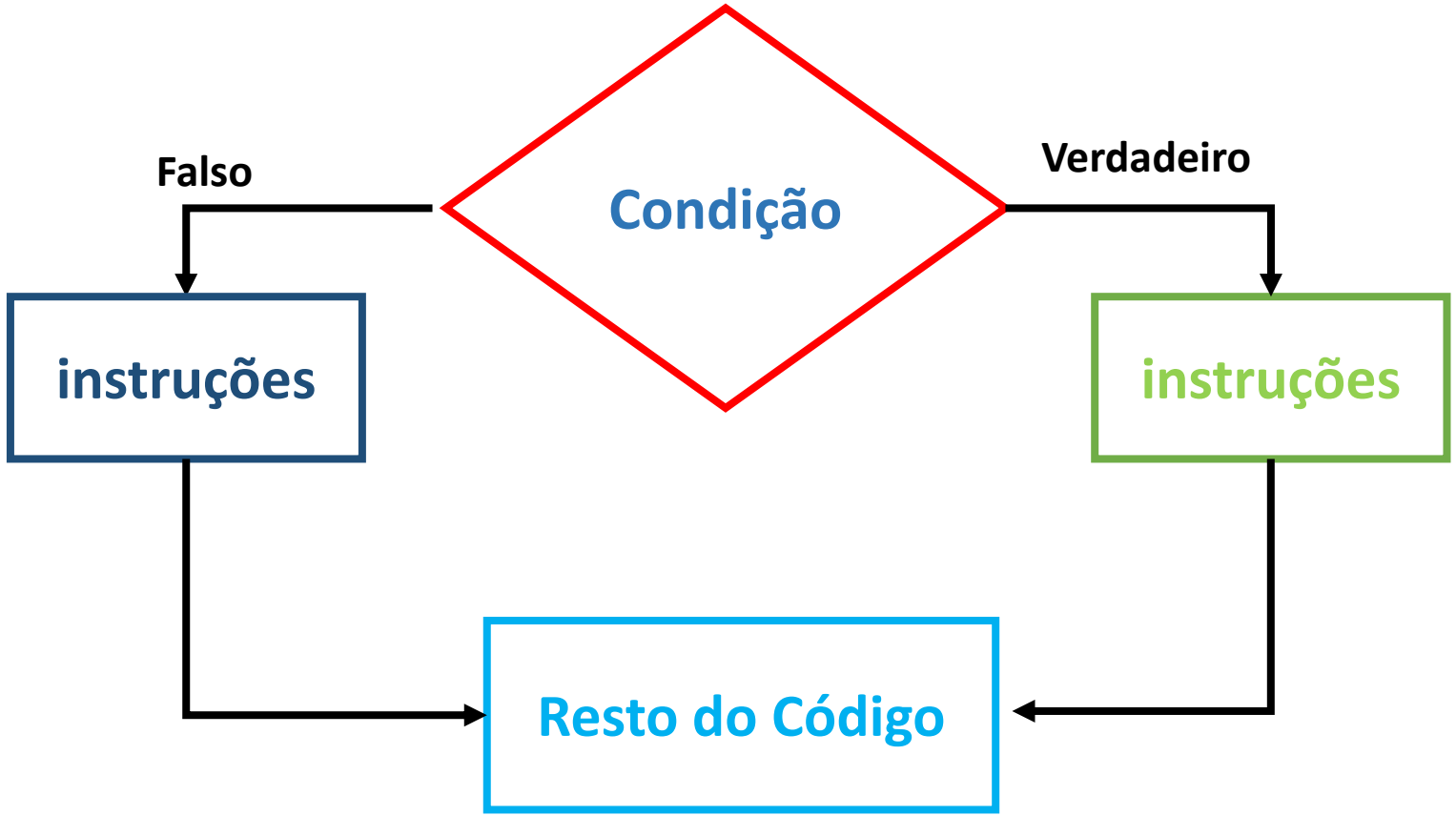
A expressão a ser avaliada não necessita obrigatoriamente estar entre parênteses, no entanto ao removê-los, você precisa ter pelo menos um espaço entre o **if** e a **condição**. Ex.: `if idade<18:`

```
idade = int(input('Digite sua idade ? '))

if(idade<18):
    print('Você não tem idade para dirigir...')

print('Verificação de idade concluída !!! ')
```

Estruturas Condicionais – if-else

Nesta estrutura if-else, um trecho de código será executado se a condição for verdadeira e outro se a condição for falsa.

```
if <condição>:
        instruções
        instruções
else:
        instruções
        instruções
resto do código
```



Estruturas Condicionais – if-else

A expressão a ser avaliada não necessita obrigatoriamente estar entre parênteses, no entanto ao removê-los, você precisa ter pelo menos um espaço entre o **if** e a **condição**. Ex.: `if idade<18:`

```
idade = int(input('Digite sua idade ? '))

if(idade<18):
    print('Você não tem idade para dirigir...')
else:
    print('Pode acelerar...')

print('Verificação de idade concluída !!! ')
```

Estruturas Condicionais – if-elfi-else

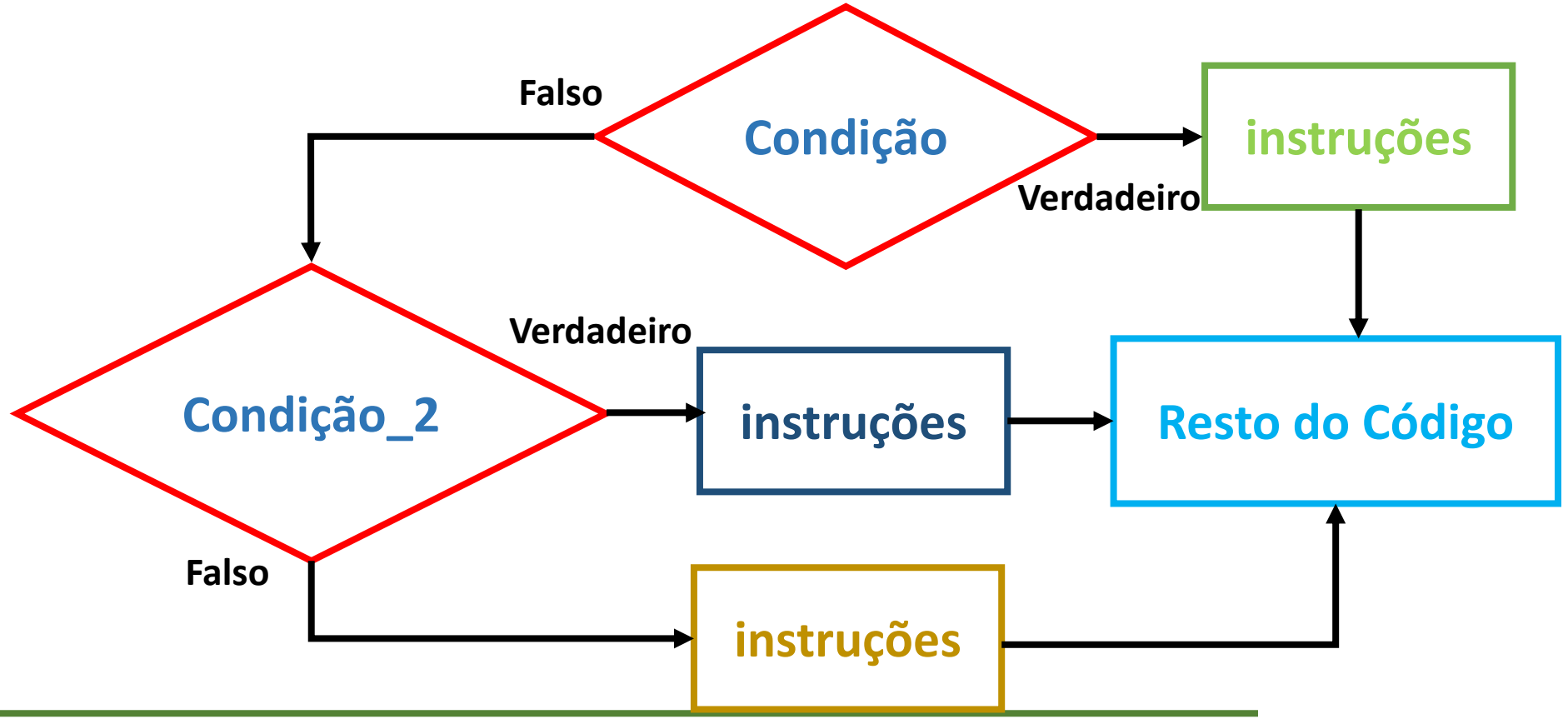
Se houver diversas condições, cada uma associada a um trecho de código, utiliza-se o elif.

```
if <condição>:
        instruções
elif <condição_2>:
        instruções
else:
        instruções
resto do código
```

Estruturas Condicionais – if-elfi-else

Se houver diversas condições, cada uma associada a um trecho de código, utiliza-se o elif.

```
if <condição>:
        instruções
elif <condição_2>:
        instruções
else:
        instruções
resto do código
```



Estruturas Condicionais – if-elfi-else

A expressão a ser avaliada não necessita obrigatoriamente estar entre parênteses, no entanto ao removê-los, você precisa ter pelo menos um espaço entre o **if** e a **condição**, entre o **elif**, a **condição** e o operador lógico **and**.

```
idade = int(input('Digite sua idade ? '))

if(idade<18):
    print('Você não tem idade para dirigir...')
elif ((idade>=18) and (idade<=95)):
    print('Pode acelerar...')
else:
    print('Cuidado, o senhor já está com uma certa idade.')

print('Verificação de idade concluída !!! ')
```

Estruturas Condicionais – if-elfi-else

Para a estrutura condicional **if**, podem ser utilizados quantos ***elif*** forem necessários, mas somente um **else**.

```
if <condição> :
        <bloco de código>
elif <condição>:
        <bloco de código>
elif <condição>:
        <bloco de código>
else:
        <bloco de código>
```

Estruturas Condicionais – if-elfi-else

Veja que nesse caso temos que escrever várias vezes as estruturas **if**, **elif** e **else** para chegar a esse resultado.

```
dia = int(input('Numero do dia da semana: '))

if dia == 1:
        print("Domingo")
elif dia == 2:
        print("Segunda-Feira")
elif dia == 3:
        print("Terça-Feira")
else:
        print(f"Valor {dia} inválido!")
```

Estruturas Condicionais – match case

Ao invés de ficar escrevendo **if**, **elif** e **else** nós temos apenas o **match** com a variável e o **case** com os valores que a variável pode assumir.

```
dia = int(input('Numero do dia da semana: '))

match dia:
      case 1:
            print("Domingo")
      case 2:
            print("Segunda-Feira")
      case 3:
            print("Terça-Feira")
      case _:
            print(f"Valor {dia} inválido!")
```

Praticando com o Python...

1) Desenvolva um script que solicite dois números quaisquer e mostre o maior entre eles.

2) Crie um programa que peça um valor e imprima na tela se o valor é positivo, negativo ou ainda igual a zero.

3) Elabore um programa que verifique se uma letra digitada é "F" ou "M". Conforme a letra escrever: F - Feminino, M – Masculino ou Sexo Inválido.

4) Escreva um programa que verifique se uma letra digitada é vogal ou consoante. Ou ainda se não está nestes grupos.

Praticando com o Python...

5) Crie um programa em Python para calcular a média de três notas inseridas pelo usuário e dar feedback baseado na média calculada.

Peça ao usuário para inserir as notas de três avaliações.
Calcule a média das notas e arredonde-a para duas casas decimais.
Exiba a média na tela.
Dê um dos seguintes feedbacks de acordo com a média:
Média maior ou igual a 7.0: "Parabéns! Sua média é alta."
Média maior ou igual a 5.0: "Sua média é razoável."
Média abaixo de 5.0: "Sua média é baixa. É uma boa oportunidade para melhorar.".

Praticando com o Python...

6) Um posto de abastecimento está comercializando combustíveis com a seguinte tabela de descontos:

Álcool:	até 20 litros, desconto de 2% por litro; acima de 20 litros, desconto de 5% por litro;
Gasolina:	até 20 litros, desconto de 4% por litro; acima de 20 litros, desconto de 6% por litro;

Desenvolva um programa em Python que leia o número de litros vendidos e o tipo de combustível (codificado com A – Álcool e G – Gasolina), calcule e imprima o valor a ser pago pelo cliente, sabendo que o litro da gasolina está em R$ 5,57 e do álcool R$ 4,98.

Praticando com o Python...

7) Neste exercício, você vai criar um programa em Python que verifica se um componente elétrico está obedecendo à Lei de Ohm. A Lei de Ohm relaciona a tensão (V), a corrente (I) e a resistência (R) de um componente elétrico através da fórmula V = I * R.

Peça ao usuário para inserir o valor da tensão (V) em volts.
Peça ao usuário para inserir o valor da corrente (I) em amperes.
Peça ao usuário para inserir o valor da resistência (R) em ohms.
Calcule a tensão esperada usando a fórmula V = I * R.
Compare a tensão calculada com o valor inserido de tensão (V).
Se não houver diferença entre a tensão calculada e o valor de tensão inserido, exiba: "O componente obedece à Lei de Ohm." Caso contrário, exiba: "O componente não obedece à Lei de Ohm."

Praticando com o Python...

8) Criar um programa em Python que ajuda a verificar se um parafuso está apertado corretamente de acordo com o torque especificado. O torque é uma medida de força rotacional aplicada a um objeto, e é especialmente importante na engenharia mecânica para garantir a segurança das montagens.

Peça ao usuário para inserir o valor do torque aplicado (em Nm).
Peça ao usuário para inserir o valor do torque de aperto recomendado (em Nm) para o parafuso em questão.
Compare o torque aplicado com o torque de aperto recomendado.
Se o torque aplicado estiver dentro de 10% acima ou abaixo do torque recomendado, exiba: "O parafuso está apertado corretamente." Caso contrário, exiba: "O parafuso não está apertado corretamente."


Obrigado!