ESTRUTURAS CONDICIONAIS

Este capítulo trata das estruturas condicionais ou, simplesmente, desvios do fluxo de execução de um programa. Estamos falando das estruturas: `if`, `else` e `elif`. No começo deste capítulo, trataremos das explicações necessárias ao entendimento das estruturas condicionais. Na sequência, apresentaremos os operadores comparativos/relacionais e lógicos. Logo em seguida, explicaremos, de forma individualizada, cada estrutura e sua sintaxe. Concluiremos comentando exercícios resolvidos.

ONDE ENCONTRAMOS ESTRUTURAS CONDICIONAIS?

Conforme observado no capítulo anterior, por padrão, instruções de um programa Python são executadas na ordem em que foram inseridas no código, ou seja, uma após a outra, do início ao fim. Esse procedimento é conhecido como execução sequencial. Entretanto, em várias ocasiões, é necessário decidir a ordem de execução dos comandos a partir de condições pré-estabelecidas. Isso acontece, por exemplo, em situações cotidianas. Imagine o seguinte: você tem uma consulta com horário marcado e, no deslocamento até o local da consulta, tem que decidir qual o caminho mais rápido para não se atrasar. Bom, você conhece dois caminhos, um mais

distante e outro mais curto. Porém, ouve no rádio que o caminho mais curto se encontra congestionado. O que fazer? Você terá que decidir qual caminho deve escolher: o mais longo (sem congestionamento) ou o mais curto (com congestionamento). E aí? Algo similar acontece com os códigos de linguagem de programação.

Em outro exemplo, você foi convidado a criar um programa para calcular a média aritmética das notas de um aluno qualquer e, em seguida, apresentar o resultado de aprovação do aluno com base nas seguintes regras: se a média aritmética for maior ou igual a 60 o aluno é considerado aprovado, caso contrário estará reprovado. Como você resolveria isso? Vamos lá. Seu programa deve receber duas notas, as quais você armazenaria em duas variáveis distintas, por exemplo, **`nota1`** e **`nota2`**. Em seguida, calcularia a média aritmética das notas e, só agora, se preocuparia com os testes da nota para aprovação ou reprovação.

Até o momento, com os recursos estudados no Capítulo 2, como você resolveria isso? Vejamos o Programa 14.

```
1    nota1 = int(input("Informe a nota do bimestre 1 (0-100): "))
2    nota2 = int(input("Informe a nota do bimestre 2 (0-100): "))
3    media = (nota1 + nota2) / 2
4    estado_Aprovacao = media >= 60
5    # O que fazer?
```

Programa 14 – Cálculo da média aritmética de duas notas de um aluno qualquer.

Perceba que, até agora, conseguimos calcular a média e até saber o estado de aprovação, com base em um teste

realizado com a variável **media**. No entanto, não conseguimos avançar com respeito ao resultado da aprovação. Como faremos para apresentar o resultado "Aprovado" ou "Reprovado"? É aí que entram as estruturas condicionais, ou seja, a partir do resultado de um teste condicional, executaremos comandos e/ou deixaremos de executar outros. Certamente, agora é aquele momento que você pensa: “Hamm?”. Calma! Agora, vamos contar com o auxílio das palavras chaves **if** e **else**. Com elas, conseguiremos decidir se apresentaremos o resultado "**Aprovado**" ou "**Reprovado**", como se vê no Programa 15.

```
1    nota1 = int(input("Informe a nota do bimestre 1 (0-100): "))
2    nota2 = int(input("Informe a nota do bimestre 2 (0-100): "))
3    media = (nota1 + nota2) / 2
4    if media >= 60:
5       print("Aprovado")
6    else:
7       print("Reprovado")
```

Programa 15 – Cálculo da média aritmética de duas notas usando testes condicionais.

Observe que o código reflete nossa estratégia mental de solução do problema. Após a declaração e leitura das variáveis, calculamos a média e testamos esse valor para decidir se o aluno foi aprovado ou reprovado. O referido teste (linhas 4 – 7) pode ser interpretado da seguinte maneira: Se (**if**) a média for maior ou igual a 60, imprima a palavra Aprovado, senão (**else**), imprima a palavra Reprovado. Portanto, a palavra-chave **if** é um teste que vai decidir que comandos serão executados: **print("Aprovado")** ou **print("Reprovado"**).

Para consolidar melhor o seu entendimento, vamos

pensar em outro exemplo. Imagine que, mais uma vez, você foi convidado para criar um programa Python, desta vez, para um banco que realiza empréstimos. Para isso, o banco lhe forneceu uma árvore de decisão[13], conforme Figura 17, indicando quais clientes podem conseguir empréstimo.

Por onde começamos? Inicialmente, idealize como seria a solução do problema com os recursos que você já possui na sua "caixa de ferramentas de programação". Nós fazemos referência aos comandos que você já aprendeu, tais como: criação de variáveis, leitura e impressão de valores, estrutura sequencial de comandos e agora, é claro, estruturas condicionais. Mãos à obra!

```
1   print("## Programa de empréstimos ##. Responda: (0 – Não e 1 - Sim) ")
2    nomeNegativado = int(input("Possui nome negativado? "))
3    if nomeNegativado == 1:
4       print("Não pode realizar empréstimo")
5    else:
6       carteiraAssinada = int(input("Possui carteira assinada? "))
7    if carteiraAssinada == 0:
8       print("Não pode realizar empréstimo ")
9    else:
10      possuiCasaPropria = int(input("Possui casa própria? "))
11   if possuiCasaPropria == 0:
12      print("Não pode realizar empréstimo")
13   else:
14      print("Conceder empréstimo")
```

Programa 16 – Exemplo de implementação de uma árvore de decisão.

13 Uma árvore de decisão é uma representação de uma tabela de decisão sob a forma de uma árvore, porém podem existir outras aplicações.

No Programa 16, por meio dos comandos **`if`** e **`else`**, observe que foi possível converter a árvore de decisão em um programa que, por sinal, parece muito com a Figura 17. Inicialmente, realizamos a leitura do estado de **`nomeNegativado`** do cliente. Logo em seguida, verificamos se o estado é negativado ou não.

Provavelmente, você já notou que um comando **`if`** quer saber se algo é verdadeiro ou falso. A utilização de operadores lógicos e de comparação (discutidos logo mais) sempre geram como resultado um valor lógico, isto é, **`True`** ou **`False`**. Isso permite decidir que comandos serão executados e quais deixaram de ser. Portanto, no pseudocódigo do Programa 17, temos a seguinte análise.

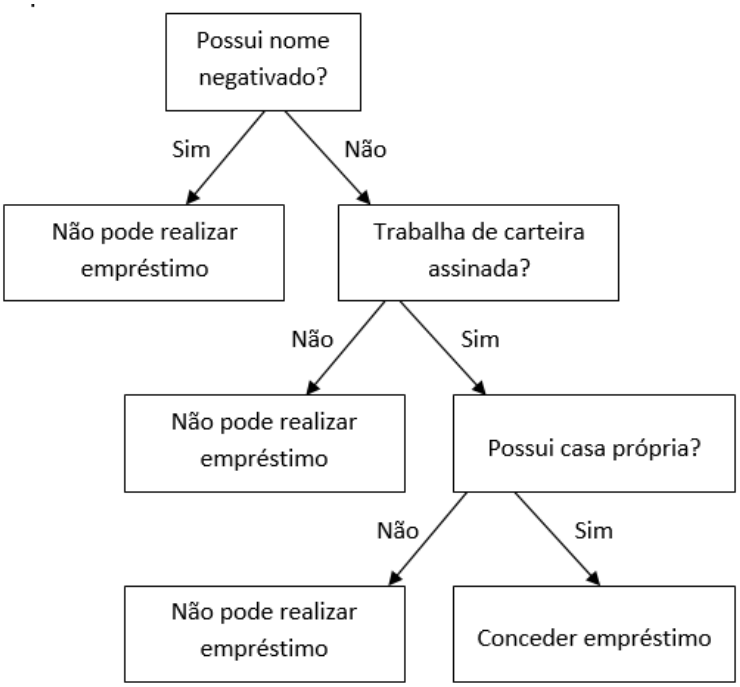

Figura 17 – Exemplo de árvore de decisão para criar programa de empréstimo.

```
1    if True:   #Se verdade
2       #Instrução 1 que será executada
3       #Instrução 2 que será executada
4       #...
5       #Instrução N que será executada
6    else:              #Se falso
7       #Instrução 1 que será executada
8       #Instrução 2 que será executada
9       #...
10      #Instrução N que será executada
```

Programa 17 – Pseudocódigo da estrutura **`if/else`**.

OPERADORES

Em programação, logo no início do processo de aprendizagem, quando começamos a receber dados do usuário, percebemos a necessidade de realizar testes ou operações com eles. Por exemplo, comparar a igualdade ou diferença entre dois valores, comparações do tipo maior ou menor com ou sem igualdade e, por vezes, precisamos realizar testes complexos envolvendo mais de uma comparação. É nessa hora que surgem os operadores de comparação (também chamados de relacionais) e os operadores lógicos. Talvez você esteja se perguntando: por que estamos tratando de operadores neste capítulo? Nós prontamente responderemos. Operadores comparativos e lógicos estão diretamente ligados com estruturas condicionais, pois na maioria das vezes que utilizamos comandos **`if`** ou **`elif`**, temos um teste associado com a utilização de operadores comparativos. Inclusive, a maioria dos exemplos, já tratados neste capítulo, relataram isso. Pois bem, vamos conhecer estes operadores.

OPERADORES DE COMPARAÇÃO OU RELACIONAIS

Os operadores de comparação, também chamados de operadores relacionais, como o próprio nome sugere, comparam ou relacionam valores. A linguagem Python possui operadores comuns encontrados em outras linguagens, o que facilita a migração da aprendizagem. Inclusive, não difere muito do aprendizado matemático visto, por exemplo, em assuntos como inequações ou intervalos. Conheça os operadores apresentados no Quadro 3.

OPERADOR	DESCRIÇÃO	EXEMPLO DE APLICAÇÃO
`==`	Igual	`3 == 2 #resulta em False`
`!=`	Diferente	`3 != 2 #resulta em True`
`>`	Maior que	`3 > 2 #resulta em True`
`<`	Menor que	`3 < 2 #resulta em False`
`>=`	Maior ou igual a	`3 >= 2 #resulta em True`
`<=`	Menor ou igual a	`3 <= 2 #resulta em False`

Quadro 3 – Lista de operadores de comparação.

OPERADORES LÓGICOS

Python, assim como outras linguagens, possui três operadores lógicos para realização de testes compostos. Veja quem são esses operadores, conforme Quadro 4.

OPERADOR	EXPRESSÃO	EXEMPLO DE APLICAÇÃO
`and`	X and Y	`True and False #resulta em True`
`or`	X or Y	`True or False #resulta em True`
`not`	not X	`not False #resulta em True` `not True #resulta em False`

Quadro 4 – Lista de operadores lógicos.

Veja como funcionam os operadores lógicos no Programa 18.

```
1    print(not False)          #Imprime True
2    print(not True)           #Imprime False
3    print(False and False)     #Imprime False
4    print(False and True)      #Imprime False
5    print(True and False)      #Imprime False
6    print(True and True)       #Imprime True
7    print(False or False)      #Imprime False
8    print(False or True)       #Imprime True
9    print(True or False)       #Imprime True
10   print(True or True)        #Imprime True
```

Programa 18 – Exemplo do uso de operadores lógicos.

Para consolidar o seu entendimento, imagine que você precisa ler três diferentes números inteiros e, ao fim, informar qual é o maior deles. Em Python, uma maneira de resolver este problema é apresentado no Programa 19.

```
1    numero1 = int(input("Digite o número 1: "))
2    numero2 = int(input("Digite o número 2: "))
3    numero3 = int(input("Digite o número 3: "))
4    if numero1 == numero2 or numero2 == numero3 or numero1 == numero3:
5       exit()   #Encerra o programa
6    if numero1 > numero2 and numero1 > numero3:
7       print("O primeiro número é o maior")
8    if numero2 > numero1 and numero2 > numero3:
9       print("O segundo número é o maior")
10   if numero3 > numero1 and numero3 > numero2:
11      print("O terceiro número é o maior")
```

Programa 19 – Exemplo para encontrar maior valor.

No Programa 19, perceba que temos quatro estruturas condicionais. A primeira quer saber se temos pelos menos dois números iguais e, caso verdade, o programa será encerrado utilizando a função **`exit()`**. As demais estruturas condicionais querem saber se um dos números é maior que os demais. Foi implementado um teste para o **`numero1`**, um teste para o **`numero2`** e um teste para o **`numero3`**. Veja a diferença entre os operadores lógicos, na primeira estrutura condicional, onde utilizamos o operador **`or`**, se apenas um teste for verdadeiro, o resultado será verdadeiro. Nas demais estruturas condicionais que utilizam o operador **`and`**, é necessário que os dois testes sejam verdadeiros para que o resultado seja verdade.

TABELA VERDADE

A tabela verdade é um recurso utilizado no estudo da lógica matemática. Com o uso desta tabela, é possível definir o valor lógico de uma proposição[14], isto é, saber quando dois ou mais testes comparativos resultam em verdadeiro ou falso, o que é bastante comum em programação. Veja o seguinte exemplo: para doar sangue, um requisito mínimo é a avaliação da idade: o doador tem que ter entre 16 e 69 anos. Um teste lógico que compreende esse requisito, em termos de programação Python, seria: **`idade >= 16 and idade <= 69`**. Veja que não é aceitável uma idade menor que 16 e nem maior que 69 anos, ou seja, os dois testes precisam ser verdadeiros. Portanto, para combinar testes simples e formar testes compostos, são utilizados conectivos lógicos. Estes conectivos

14 Proposição é uma sentença declarativa cujo conteúdo será considerado verdadeiro ou falso. Então, as proposições “A é uma vogal” resulta em um valor verdadeiro e “3 é par” resulta em um valor falso.

representam operações lógicas que formam as tabelas verdades. Na matemática, existem os conectivos lógicos, que são representados simbolicamente, como visto no Quadro 5.

CONECTIVO	SÍMBOLO	OPERAÇÃO LÓGICA	VALOR LÓGICO
Não	~	Negação	Valor falso quando o teste for verdadeiro e vice-versa
E	^	Conjunção	Verdadeiro quando todos os testes da composição forem verdadeiros
Ou	V	Disjunção	Verdadeiro quando pelo menos um dos testes for verdadeiro

Quadro 5 – Lista de conectivos lógicos.

Assumindo V como verdadeiro, F como falso, T1 como Teste1 e T2 como Teste2, nos próximos 3 quadros, observe a tabela verdade para cada um dos conectivos lógicos.

NÃO	
T1	~T1
F	V
V	F

Quadro 6 - Tabela verdade do conectivo lógico Não.

E		
T1	T2	T1^T2
F	F	F
F	V	F
V	F	F
V	V	V

Quadro 7 – Tabela verdade do conectivo lógico E.

OU		
T1	T2	T1vT2
F	F	F
F	V	V
V	F	V
V	V	V

Quadro 8 – Tabela verdade do conectivo lógico Ou.

Em Python, para cada um dos conectivos apresentados anteriormente, a tabela verdade pode ser reescrita por meio da utilização de operadores lógicos, apresentados anteriormente.

NOT	
T1	not T1
False	True
True	False

Quadro 9 - Tabela verdade do operador lógico not em Python.

AND		
T1	T2	T1 and T2
False	False	False
False	True	False
True	False	False
True	True	True

Quadro 10 – Tabela verdade do operador lógico and, em Python.

OR		
T1	T2	T1 or T2
False	False	False
False	True	True
True	False	True
True	True	True

Quadro 11 – Tabela verdade do operador lógico or em Python.

ESTRUTURA IF...ELSE

Em Python, a estrutura básica de um comando **`if`** é bastante simples. Temos a palavra-chave **`if`**, seguida de um teste lógico, valor ou variável e o caractere dois pontos (:). No Programa 20, observe que nas linhas 3 e 4, temos um bloco de instruções, que em Python, é caracterizado por uma ou várias instruções indentadas (isto é, o recuo entre a margem e o início da instrução). Na prática, você utiliza a tecla TAB para realizar o alinhamento de todas as instruções abaixo do **`if`**, criando assim um bloco. Veja o exemplo a seguir.

```
1    nome = input("Digite seu nome completo: ")
2    if len(nome) > 50: #len() retorna o número de caracteres da string
3       print("Seu nome é grande, ele possui mais de 50 letras. ")
4       print(f"Ele possui {len(nome)} caracteres.")
```

Programa 20 – Exemplo da sintaxe do comando **`if`**.

Em outras linguagens de programação, como Java ou C++, por exemplo, um bloco de instruções é definido pelo envolvimento de um par de chaves { } para o início e o fim do bloco. No caso de Python, por questões de melhoria da visualização do código, a indentação de instruções é utilizada. Então, de hoje

em diante, quando você observar código indentado em Python, tenha certeza de que é um bloco de instruções. Inclusive, cada instrução **`if`** terá o seu bloco indentado.

Uma informação importante: em Python, assim como em outras linguagens de programação, uma expressão pode ser considerada verdadeira se ela possui valor diferente de zero. Caso contrário, é considerada falsa. Veja o exemplo abaixo:

```
1    valor = 10
2    if valor:
3        print("valor é diferente de zero => verdade.")
4    valor = 0
5    if not valor:
6        print("Valor é falso, mas not inverteu o resultado.")
```

Programa 21 – Considerando inteiros como expressões do comando **`if`**.

Já a palavra-chave **`else`** é aplicada nas condições em que é necessário executar instruções quando o teste do **`if`** não for satisfeito, ou seja, o resultado do teste é falso. Para nós, a palavra-chave **`else`** é mais bem entendida quando interpretada como a expressão “senão”. Sua sintaxe é simples, quando encerradas as instruções do **`if`**, no mesmo alinhamento do **`if`**, adiciona-se a palavra-chave **`else`** e o caractere dois pontos **`(:)`**. Abaixo dessa linha, vêm as instruções indentadas formando um bloco. Veja o Programa 22.

```
1    valor = int(input("Número: "))
2    if valor % 2 == 0:
3        print("O valor é par")
4    else:
5        print("O valor é ímpar")
```

Programa 22 – Exemplo da sintaxe do comando **`else`**.

Deixemos de falar muito e vamos praticar. Para clarear o seu entendimento, começaremos do simples para o avançado. Inicialmente, construiremos o Programa 23 para avaliar se um número digitado pelo usuário é par ou ímpar.

```
1    valor = int(input("Número: "))
2    if valor % 2 == 0:
3       print("O valor é par")
4    else:
5       print("O valor é ímpar")
```

Programa 23 – Programa par ou ímpar.

O código de resposta do problema é bem simples. Na linha 1, temos a leitura de um valor inteiro qualquer. Já na linha 2, para saber se um número é par, recorremos à seguinte lógica: para todo número par, quando dividido por 2, o resto da sua divisão é zero. Portanto, utilizamos o operador `%` que retorna o resto da divisão. Caso o resultado de valor `% 2` seja zero, imprimimos a mensagem "`O valor é par`", caso contrário (exemplo de uso do **`else`**), imprimimos a mensagem "`O valor é impar`".

Agora, você foi convidado a resolver o seguinte problema: uma loja de roupas está vendendo camisas básicas com descontos, mas seus funcionários têm dificuldades no cálculo do valor final a ser cobrado. Por isso, seu Tanaka, dono da loja, convidou você para criar um programa para calcular o preço final a partir do número de camisas. Seu Tanaka definiu as seguintes regras de desconto:

- Até 5 camisas, desconto de 3%
- Acima de 5 camisas e até 10 camisas, desconto de 5%; e
- Acima de 10 camisas, desconto de 7%.

Pronto, calcule e imprima o valor a ser pago pelo cliente, sabendo que o preço da camisa é R$ 12,50. Veja a solução no Programa 24.

```
1    numeroCamisas = int(input("Quantidade de camisas: "))
2    valorCamisa = 12.50
3    valorFinal = numeroCamisas * valorCamisa
4    if numeroCamisas <= 5:
5       valorFinal = valorFinal * (1 - 3/100)
6    else:
7       if numeroCamisas <= 10:
8               valorFinal = valorFinal * (1 - 5/100)
9       else:
10              valorFinal = valorFinal * (1 - 7/100)
11   print(f"Valor final: R$ {valorFinal:.2f}")
```

Programa 24 – Programa da loja do Seu Tanaka.

Analisando o código, percebemos o seguinte: na primeira linha, temos o código para leitura da quantidade de camisas. Na segunda linha, declaramos uma variável para conter o valor individual de uma camisa. Na terceira linha, calculamos o valor final sem descontos. A partir da linha 4, iniciamos as análises para aplicação de desconto. O primeiro teste foi para verificar se o número de camisas é menor ou igual a 5, caso verdade, aplicamos 3% de desconto. Caso contrário, realizaremos um novo teste para saber se a quantidade de camisas é menor ou igual a 10 já que, certamente, a quantidade foi maior que 5. Nesse caso, se verdade, aplicaremos o desconto de 5%, caso contrário, ou seja, acima de 10, aplicaremos 7%.

ESTRUTURA ELIF

No Programa 24, utilizamos um comando **`if`** após o pri-

meiro comando **else**. Nesses casos, perceba que o código cresce lateralmente por causa do excesso de indentação, o que pode provocar dificuldades no entendimento do código. Para minimizar isso, o Python oferece uma solução elegante que diminui o uso de indentação, facilitando a visualização do código. Estamos falando da palavra-chave **elif**, que é uma espécie de contração do comando **else** e um comando **if**. Logo, ela é utilizada em substituição ao comando **else**, mas permitindo a realização de um novo teste. Veja a solução do mesmo problema agora usando a estrutura **elif**.

```
1    numeroCamisas = int(input("Número de camisas: "))
2    valorCamisa = 12.50
3    valorFinal = numeroCamisas * valorCamisa
4    if numeroCamisas <= 5:
5       valorFinal = valorFinal * (1 - 3/100)
6    elif numeroCamisas <= 10:
7       valorFinal = valorFinal * (1 - 5/100)
8    else:
9       valorFinal = valorFinal * (1 - 7/100)
10   print(f"Valor final: R$ {valorFinal:.2f}")
```

Programa 25 – Programa da loja do Seu Tanaka usando **elif.**

EXERCÍCIOS RESOLVIDOS

1. Uma empresa, que presta serviço à companhia de energia elétrica do estado, necessita de um programa que auxilie os seus eletricistas no cálculo das principais grandezas da Eletricidade que são Tensão, Resistência e Corrente. Sabe-se que

$$U=Ri,$$

onde, U é a Tensão (em V), R é a Resistência (em Ω) e i a Corrente (em A).

Você foi contratado(a) pela empresa para atender a essa solicitação. Construa um programa que apresente o seguinte menu:

```
************************************************
        CÁLCULO DE GRANDEZAS ELÉTRICAS
************************************************
1. Tensão (em Volt)
2. Resistência (em Ohm)
3. Corrente (em Ampére)
************************************************
Qual grandeza deseja calcular?
```

Em seguida, o programa deve solicitar que o eletricista informe o valor das outras duas grandezas para realizar o cálculo. Quando o eletricista escolher:

a. Tensão, o programa deve solicitar que ele informe os valores da Resistência e da Corrente
b. Resistência, o programa deve solicitar que ele informe os valores da Tensão e da Corrente
c. Corrente, o programa deve solicitar que ele informe os valores da Tensão e da Resistência

Por fim, o programa deve calcular e apresentar o valor encontrado para a grandeza escolhida.

Comentários sobre a resolução

Para solução deste problema, a primeira coisa a ser feita é a construção do menu. Conseguimos isso por meio de um conjunto de comandos de impressão, ou seja, uma boa utilização do comando **`print`**.

Observe a linha 1. Nela temos uma **string** formatada que possui três campos de substituição "**{} {} {}**". O primeiro e o último campo são substituídos por uma **string** montada a partir do operador *****, que quando utilizado com **string**, realiza a concatenação do seu conteúdo, o número de vezes do valor do operando que está ligado ao operador. Nesse caso, o valor 4 "********". As linhas 2 – 4 realizam a impressão das opções que compõem o menu. A linha 5 utiliza novamente o recurso de concatenação obtido com a utilização do operador *****, que nesse caso realiza a impressão de 48 asteriscos na mesma linha.

Na linha 6, temos a utilização da função **input** para entrada de dados. A função **input** é envolvida pela função **int**, que converterá a **string** devolvida pela função **input** em um inteiro, que será atribuído a variável **op** (opção). Por que isso? Simplesmente, porque é melhor trabalhar com comparação de inteiros do que comparação de **strings**.

Na linha 7, temos um teste para validar o valor da variável **op**. Ela precisa estar com o valor entre 1 e 3, que são as opções do menu. No teste foi utilizado o operador lógico **or** para que quando o valor de **op** for menor que 1 ou maior que 3, o programa imprima a frase "**Opção inválida**". Caso contrário, o programa deve continuar com as análises do valor de **op**.

O primeiro teste de **op**, utilizando a palavra-chave **elif**, é com o valor 1 e o operador comparativo de igualdade **==**. Caso seja verdade, ou seja, **op** é igual a 1, segue-se a leitura das variáveis solicitadas e o cálculo da variável selecionada pela opção do menu. No caso da opção 1, temos a leitura do valor da resistência que

é atribuída a variável **`r`**, a leitura do valor de corrente que é atribuído a variável **`i`** e, por fim, o cálculo da variável **`v`** (**`v = r * i`**). Logo em seguida, ocorre a impressão da **`string`** formatada com o resultado da variável calculada.

Para as demais opções, ocorre algo semelhante. O usuário escolhe valor 2 ou 3, informa as variáveis solicitadas, calcula o valor da variável escolhida e imprime uma **`string`** formatada com o resultado da operação.

Vamos ao código:

```
1    print(f"{'*' * 18} {'GRANDEZAS ELÉTRICAS '} {'*' * 18}")
2    print("1. Tensão (em Volt)")
3    print("2. Resistência (em Ohm)")
4    print("3. Corrente (em Ampére)")
5    print("*" * 48)
6    op = int(input("Qual grandeza deseja calcular? "))
7    if op < 1 or op > 3:
8       print("Opção inválida.")
9    elif op == 1:
10      R = float(input("Digite o valor da corrente (em Ohm): "))
11      i = float(input("Digite o valor da corrente (em Ampére): "))
12      U = R * i
13      print(f"\nU = {U:.2f}")
14   elif op == 2:
15      U = float(input("Digite o valor da tensão (em Volt): "))
16      i = float(input("Digite o valor da corrente (em Ampére): "))
17      R = U / i
18      print(f"\nR = {R:.2f}")
19   else:
20      U = float(input("Digite o valor da tensão (em Volt): "))
21      R = float(input("Digite o valor da corrente (em Ohm): "))
22      i = U / R
23      print(f"\ni = {i:.2f}")
```

Programa 26 – Exercício resolvido 3.1

2. Considere um triângulo T, representado no sistema de coordena-

das cartesianas, definido por três pontos: $P_1 (x_1, y_1)$, $P_2 (x_2, y_2)$ e $P_3 (x_3, y_3)$. Cada lado do triângulo é obtido por meio do cálculo da distância entre os pontos, conforme equação abaixo

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

Para que T exista, é necessário que:
a) todos os lados sejam maiores que zero e;
b) um dos lados seja menor que a soma dos outros dois e maior que o valor absoluto da diferença entre os lados. Veja as restrições:

i. |B-C|<A<B+C
ii. |A-C|<B<A+C
iii. |A-B|<C<A+B

Para cada conjunto de três pontos informados, o programa deve escrever o tipo do triângulo formado (Isósceles, Escaleno ou Equilátero), bem como os lados dos triângulos. Caso os pontos informados não formem um triângulo, o programa deve apresentar a mensagem "**`Nenhum triângulo formado com os pontos informados.`**".

Comentários sobre a resolução

Vamos resolver este belo problema? Na primeira linha, ocorre a importação da função **`sqrt`** (cálculo de raiz quadrada) do módulo **`math`**, para evitar a escrita constante do comando **`math`**. Agora, você só precisa utilizar **`sqrt(valor)`**.

Na linha 2, temos um comentário de explicação das duas próximas linhas. Dica: comentário é sempre bom, quando bem utilizado. Nas linhas 3, 4, 6, 7, 9 e 10 temos a leitura dos pontos do triângulo, no plano cartesiano. Na ocasião, utilizamos a função **`float`** para

converter a saída da função **input** para um valor real, porque será necessário trabalhar com números reais (engloba não só os inteiros e os fracionários, positivos e negativos, mas também todos os números irracionais).

A seguir, nas próximas três linhas, calculamos os lados do triângulo a partir dos pares formados pelos três pontos, ou seja, (P1 e P2), (P2 e P3) e (P1 e P3). O cálculo é o seguinte: **L1 = sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2).** Observe a boa utilização dos parênteses. Assim como na matemática, Python resolve as operações contidas nos parênteses mais internos, para no final calcular a operação do parêntese mais externo. Isso quer dizer que ele vai resolver primeiro as operações de subtração. Na sequência, por uma questão de precedência (vir primeiro), a operação de potência do operador ****** é realizada antes da operação de soma. Para concluir o cálculo da expressão, ocorre a solução da raiz quadrada com a função **sqrt**, cujo valor resultante será atribuído a variável **L1** (lado 1).

Nas linhas 16, 17 e 18 ocorre a criação de três variáveis auxiliares com valor inicial **True**. Em programação, é comum a utilização de variáveis auxiliares para solução de problemas. No caso das variáveis citadas, elas serão utilizadas para checar se algum problema ocorreu durante a avaliação conjunta dos lados. No caso, a primeira avaliação é útil para saber se pelo menos um dos lados é zero. Caso verdade, a variável **cond1** recebe o valor **False**, indicando que a formação de um triângulo não será possível. O segundo teste verifica se pelo menos um dos lados é maior que a soma dos outros dois. Caso isso ocorra, a variável **cond2** recebe o valor **False**, indicando que não será possível a formação de triângulo. O terceiro teste sequencial quer saber se algum dos lados é maior que a

soma absoluta dos outros dois. Caso isso ocorra, a variável **cond3** recebe valor **False**, indicando que não será possível a formação de triângulo.

Na linha 28, ocorre mais uma declaração de variável auxiliar. No caso em questão, a variável **triangulo** recebe inicialmente o valor **True**, indicando que até o momento, antes dos testes finais, é possível a formação de triângulo. Na linha 30, verificamos se pelo menos uma das variáveis **cond1**, **cond2** e **cond3** possui valor **False**. Caso verdade, ocorre a impressão da frase "**Nenhum triângulo formad**o". Logo à frente, nas linhas 33, 35 e 37 ocorrem os testes individualizados de cada variável para impressão dos motivos que impediram a formação dos triângulos.

Finalmente, se o resultado da linha 30 for **False**, analisaremos as condições que indicarão o tipo do triângulo. Primeiro se os três lados são iguais (linha 40). Caso seja verdade, ocorrerá a impressão da frase "**Triângulo equilátero**", indicando o tipo do triângulo. Na linha 42, ocorre o teste que verifica se todos os lados são diferentes para informar, caso verdade, que o triângulo é escaleno. Caso não seja equilátero e nem escaleno, obviamente será isósceles. Na linha 47, a variável **triangulo** é testada e, se ela for verdadeira, os lados do triângulo serão impressos. Ufa! Este foi grande.

Para testar a existência de um triângulo:

- escaleno, use: P1 = (1, 2), P2 = (1, 7) e P3 = (3, 3)
- isósceles, use: P1 = (2, 1), P2 = (5, 1) e P3 = (2, 4)
- equilátero, use: P1 = (1, 2), P2 = (1, 7) e P3 = (5.330127018922193, 4.5)

Vamos ao código:

```
from math import sqrt
#Coordenadas x e y do Ponto 1
x1 = float(input("Digite a coordenada x do Ponto 1: "))
y1 = float(input("Digite a coordenada y do Ponto 1: "))
#Coordenadas x e y do Ponto 2
x2 = float(input("\nDigite a coordenada x do Ponto 2: "))
y2 = float(input("Digite a coordenada y do Ponto 2: "))
#Coordenadas x e y do Ponto 3
x3 = float(input("\nDigite a coordenada x do Ponto 3: "))
y3 = float(input("Digite a coordenada y do Ponto 3: "))
#Calcula os lados do triângulo
L1 = sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2) #Distância entre P1 e P2
L2 = sqrt((x3 - x1) ** 2 + (y3 - y1) ** 2) #Distância entre P1 e P3
L3 = sqrt((x3 - x2) ** 2 + (y3 - y2) ** 2) #Distância entre P2 e P3
#Considera que as três condições de existência são verdadeiras
cond1 = True
cond2 = True
cond3 = True
#Verifica se algum lado é igual a zero
if L1 == 0 or L2 == 0 or L3 == 0:
   cond1 = False
#Verifica se algum lado é maior que a soma dos outros dois
if L1 > (L2 + L3) or L2 > (L1 + L3) or L3 > (L1 + L2):
   cond2 = False
#Algum lado não é maior que o módulo da diferença entre os outros?
if L1 <= abs(L2 - L3) or L2 <= abs(L1 - L3) or L3 <= abs(L1 - L2):
   cond3 = False
triangulo = True #Inicialmente, considera a existência do triângulo
#Alguma condição de inexistência foi identificada?
if cond1 == False or cond2 == False or cond3 == False:
   triangulo = False #Triângulo inexistente
   print("\nNenhum triângulo formado.\nMotivo(s):")
   if cond1 == False:
      print("Pelo menos um dos lados é igual a 0.")
   if cond2 == False:
      print("Pelo menos um lado é maior que a soma dos outros 2.")
   if cond3 == False:
     print("Um dos lados é menor ou igual ao módulo da diferença.")
#Triângulo existe
elif L1 == L2 == L3:
   print("\nTriângulo equilátero.")
elif L1 != L2 and L1 != L3 and L2 != L3:
```

```
43      print("\nTriângulo escaleno.")
44   else:
45      print("\nTriângulo isósceles.")
46   #Todas as condições de existência do triângulo foram satisfeitas
47   if triangulo:
48      print(f"Medida do lado 1: {L1:.2f}")
49      print(f"Medida do lado 2: {L2:.2f}")
50      print(f"Medida do lado 3: {L3:.2f}")
```

Programa 27 – Exercício resolvido 3.2.

3. Escreva um programa que solicite ao usuário a estatura de 3 pessoas. Ao fim, o programa deve imprimir as estaturas em ordem decrescente.

Comentários sobre a resolução

Oba! Um problema comum em programação é a ordenação de elementos. Por isso, não poderia faltar aqui um exemplo de ordenação. Até sugerimos que você pesquise sobre os métodos de ordenação e pesquisa. Existem vários. É uma excelente maneira de você se aperfeiçoar em programação. Agora, tratando do código, adotamos a seguinte estratégia: nas três primeiras linhas, ocorre a leitura de três valores atribuídos às variáveis **`alt1`**, **`alt2`** e **`alt3`**. Na sequência, foram criadas três variáveis auxiliares: **`mais_alto`**, **`est_mediana`** e **`mais_baixo`**. Essas variáveis são utilizadas para considerar que inicialmente o valor mais alto, mais baixo e a mediana correspondem ao primeiro valor lido, no caso **`alt1`**.

Na linha 7, analisamos se a variável **`alt1`** possui o maior valor entre os três elementos. Caso sim, vamos considerar que **`alt1`** é o maior valor, armazenando o seu valor na variável **`maior`**. Aproveitamos também na linha 9 para perguntar se **`alt2`** é maior que **`alt3`**. Caso verdade, isso garante que **`alt2`** é o valor do meio e, por consequência, **`alt3`** o menor valor.

Os testes citados se repetem para a variável **`alt2`** e **`alt3`**, ou seja, primeiro identificamos o maior número e na sequência, encontramos o valor do meio e o menor valor. Fazendo assim, conseguimos reconhecer a ordem decrescente dos números.

Dica: Python permite a seguinte escrita de código: **`if 10 > maior < 100`**. Apesar de funcionar, recomendamos, por questões de compatibilidade com outras linguagens, que você escreva da seguinte maneira: **`if maior < 10 and maior < 100`**.

Vamos ao código:

```
1    alt1 = float(input("Digite a estatura da 1ª pessoa (em metros): "))
2    alt2 = float(input("Digite a estatura da 2ª pessoa (em metros): "))
3    alt3 = float(input("Digite a estatura da 3ª pessoa (em metros): "))
4    mais_alto   = alt1
5    est_mediana = alt1
6    mais_baixo  = alt1
7    if alt1 > alt2 and alt1 > alt3:
8       mais_alto   = alt1
9       if alt2 > alt3:
10         est_mediana = alt2
11         mais_baixo  = alt3
12      else:
13         est_mediana = alt3
14         mais_baixo  = alt2
15   elif alt2 > alt1 and alt2 > alt3:
16      mais_alto   = alt2
17      if alt1 > alt3:
18         est_mediana = alt1
19         mais_baixo  = alt3
20      else:
21         est_mediana = alt3
22         mais_baixo  = alt1
23   else:
24      mais_alto   = alt3
25      if alt1 > alt2:
26         est_mediana = alt1
```

```
27          mais_baixo  = alt2
28       else:
29          est_mediana = alt2
30          mais_baixo  = alt1
31    print(f"\n{mais_alto}m, {est_mediana}m e {mais_baixo}m")
```

Programa 28 – Exercício resolvido 3.3.

EXERCÍCIOS PROPOSTOS

1. Construa um programa que receba um número inteiro positivo informado pelo usuário. Caso ele seja par, o programa deve calcular o seu quadrado. Mas, se ele for ímpar, deve ser calculado o seu cubo. Ao fim, o programa deve imprimir o valor calculado.

2. Construa um programa que solicite ao usuário dois números positivos. Em seguida, o programa deve apresentar o seguinte menu:

1. **Média ponderada, com pesos 2 e 3, respectivamente**
2. **Quadrado da soma dos 2 números**
3. **Cubo do menor número**

Escolha uma opção:

De acordo com a opção informada, o programa deve calcular a operação apresentada no menu. Se a opção escolhida for inválida, o programa deve mostrar a mensagem “Opção inválida” e ser encerrado.

3. Construa um programa que apresente para o usuário o menu abaixo:

****** TABELA VERDADE ******

1. **Operador AND**

2. **Operador OR**
3. **Operador NOT**

****

Escolha uma opção:

Se a opção escolhida for 1 ou 2, o programa deve solicitar que o usuário informe 2 bits e, no caso da opção ser 3, o programa deve solicitar que o usuário informe apenas 1 bit. Ao fim, o programa deve usar o(s) bit(s) informado(s) e apresentar o resultado da operação selecionada, com base, na tabela verdade.

4. Suponha que o professor Fábio possui 2 *logins* na rede acadêmica da instituição. Construa um programa que valide o acesso do professor à rede. Caso o par usuário/senha informado esteja correto, o programa deve imprimir a mensagem “Seja bem vindo!”. Caso contrário, “Usuário e senha não conferem”.

login 1	**login 2**
usuário: procopio	usuário: paiva
senha: 12345	senha: 54321

5. Uma empresa concederá um aumento de salário aos seus funcionários, variável de acordo com o cargo, conforme a tabela abaixo:

Cargo	Aumento (%)
Programador de Sistemas	30
Analista de Sistemas	20
Analista de Banco de Dados	15

Crie um programa que solicite ao usuário o salário e o cargo de

um determinado funcionário. Na sequência, o programa deve calcular e imprimir o seu novo salário. Caso o cargo informado não esteja na tabela, o programa deve imprimir “Cargo inválido”.

6. Em sua programação semanal, uma rede de televisão apresenta os seguintes telejornais:

- Bom Dia Nação, apresentado por Zé PINHEIRO e por Ana Carla ARAÚJO
- Jornal Brasileiro, apresentado por Bill BONNER E CARLA VASCONCELOS

Crie um programa no qual o usuário informe o sobrenome de um dos apresentadores. Se o sobrenome informado não estiver na lista acima, o programa deve mostrar a mensagem “Apresentador(a) desconhecido(a).”. Em caso positivo, o programa deve mostrar o nome do telejornal apresentado pelo apresentador(a).

7. Crie um programa que solicite ao usuário a informação de 3 estaturas. Caso existam estaturas iguais, o programa deve apresentar a mensagem “Há, pelo menos, 2 pessoas com a mesma estatura”. Caso contrário, o programa deve informar a maior estatura.

8.Considere uma equação do segundo grau, representada pela expressão

$$ax^2 + bx + c = 0.$$

Construa um programa no qual o usuário informe os valores das constantes **a**, **b** e **c**. Ao fim, o programa deve calcular e imprimir as raízes reais da equação. Caso não existam raízes reais, o programa deve apresentar a mensagem “Não existem raízes reais.”.

9. Na última *Black Friday*, o gerente de uma loja de perfumes colocou todo o seu estoque em promoção, de acordo com a tabela a seguir:

Código	Condição de Pagamento	Desconto (%)
1	À vista (em espécie)	15
2	Cartão de débito	10
3	Cartão de crédito (vencimento)	5

Construa um programa que solicite ao operador do caixa o preço total da venda, bem como a forma de pagamento. Ao fim, o programa deve informar o valor final a ser pago.

10. O IMC (Índice de Massa Corporal) é utilizado para avaliar o peso de um indivíduo em relação à sua altura e, como resultado, indica se o indivíduo está com o peso ideal, acima ou abaixo do recomendado. A tabela a seguir indica a situação de um indivíduo adulto em relação ao seu IMC:

IMC	Situação
Abaixo de 18,5	Abaixo do peso
Acima de 18,5 e menor que 25	Peso normal
A partir de 25 e menor que 30	Sobrepeso
Acima de 30 e menor que 35	Obesidade grau 1
Acima de 35 e menor que 40	Obesidade grau 2
Acima de 40	Obesidade grau 3

Para calcular o IMC, é usada a fórmula $IMC = \frac{peso}{altura^2}$. Construa um programa no qual um usuário informe seu peso e sua altura. A aplicação deve indicar o IMC calculado e a situação do indivíduo.