Raciocínio Computacional

UNIDADE 02

Estruturas condicionais

Esta unidade tem por objetivo fazer com que os programas tomem decisões a partir do uso de estruturas condicionais. A partir deste ponto, os programas começam a ficar mais "inteligentes", podendo realizar diferentes tarefas de acordo com a análise de condições sobre os dados armazenados. Quanto melhor for a habilidade do programador no trato de estruturas condicionais, mais otimizados e organizados serão seus programas.

Estruturas condicionais

Uma das funções mais importantes de um programa é tomar decisões. Basicamente, um programa recebe informações a partir de algum tipo de entrada de dados (normalmente do usuário), trata esses dados, tomando decisões sobre o que fazer com eles, e apresenta os resultados. A responsabilidade por programar essas estruturas de decisões de forma correta e otimizada fica a cargo do programador.

Toda linguagem de programação possui ao menos um comando condicional que analisa uma ou mais proposições para decidir em qual sequência algoritmal a aplicação continuará executando. Cabe ressaltar que a aplicação correta de condicionais dentro da estrutura do algoritmo de um programa faz com que ele seja mais ou menos abrangente em suas funcionalidades. Da mesma forma, um programa pode executar de forma mais ou menos otimizada conforme o algoritmo é estruturado; neste caso, a correta utilização de condicionais tem um papel decisivo para atingir uma melhor *performance* na aplicação desenvolvida.

A lógica da tomada de decisões

Como já mencionado, em qualquer linguagem de programação existe ao menos uma estrutura que permite tomar decisões. Em lógica clássica, essa ação dentro do algoritmo é realizada pelo comando **se**, cuja representação diagramal é um losango, sendo que, em um de seus vértices, entra o fluxo da aplicação e, a partir de até outros dois vértices, abrem novos fluxos de ações, de acordo com a decisão tomada.

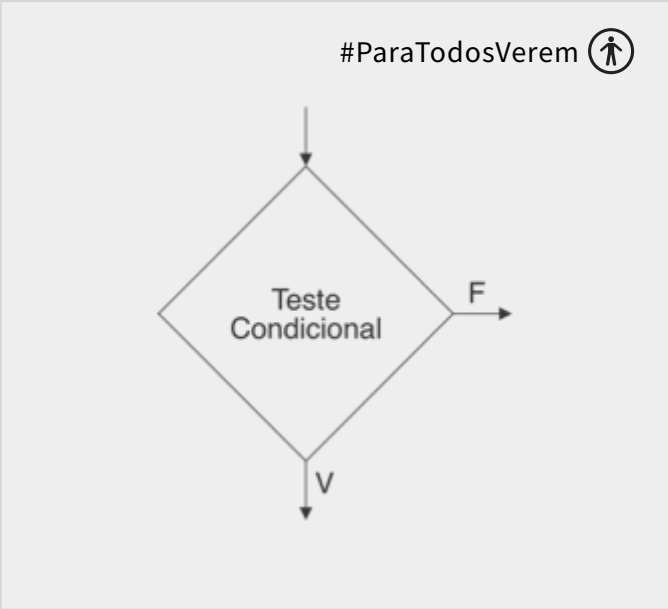


Figura 1. Representação em diagrama de fluxo do comando **se**

O comando **se** é apresentado com algumas variações de uso, o que influencia a quantidade de condições que podem ser analisadas.

Seleção simples

Esta é a forma mais simples na qual o comando **se** é apresentado. Nela, uma condição é analisada e, caso resulte em verdade, um bloco de ações é executado. Caso contrário, a seleção é encerrada.

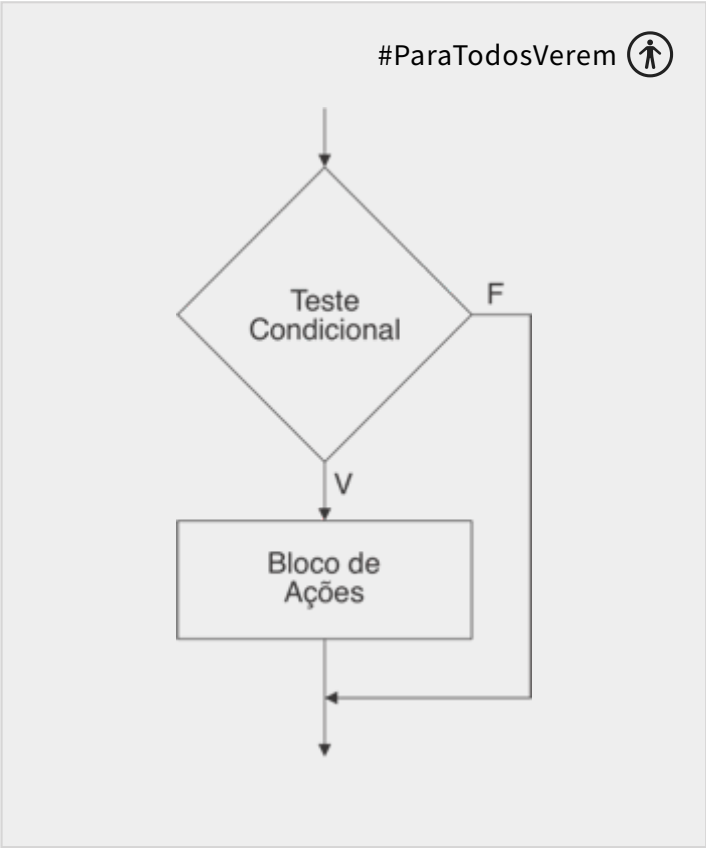


Figura 2. Representação em diagrama de fluxo da seleção simples

Exemplo de aplicação 1: Elabore um programa que solicite ao usuário seu ano de nascimento e calcule sua idade. Para ser mais assertivo, também deve perguntar se o usuário já fez aniversário neste ano e analisar a influência dessa informação no cálculo da idade.

```
 1. início
 2.     inteiro: nascimento, idade, ano_atual;
 3.     texto: resp;
 4.     ano_atual <- 2020;
 5.     escreva ("Qual é o seu ano de nascimento?");
 6.     leia (nascimento);
 7.     idade <- anoAtual – nascimento;
 8.     escreva ("Você já fez aniversário neste ano?");
 9.     leia(resp);
10.     se (resp = "não")
11.         então
12.             idade <- idade – 1;
```

```
13.     fimse;
14.     escreva ("Sua idade é ", idade);
15. fim.
```

A lógica deste exemplo é subtrair o ano atual do ano de nascimento informado pelo usuário para o cálculo da idade. Contudo, se o usuário não fez ainda aniversário neste ano, deve-se subtrair 1 desse cálculo para obter a idade correta.

É importante destacar neste exemplo a atribuição:

```
idade <- idade – 1;
```

Em programação, a atribuição inicialmente resolve a operação que está à direita do operador de atribuição e, em seguida, armazena o dado obtido na variável à esquerda.

Exemplo de aplicação 2: Elabore um programa que solicite ao usuário as notas de determinada disciplina escolar e calcule a média. Se a média for maior ou igual a 7, deve mostrar ao aluno que ele foi aprovado.

```
1. início
2.     real: nota1, nota2, nota3, nota4, media;
3.     escreva ("Digite as 4 notas bimestrais da disciplina:
   ");
4.     leia (nota1, nota2, nota3, nota4);
5.     media = (nota1 + nota2 + nota3 + nota4) / 4;
6.     escreva ("Sua média na disciplina foi ", media);
7.     se (media >= 7)
8.         então
9.             escreva ("Você está aprovado na disciplina!");
10.    fimse;
11. fim.
```

A lógica deste exemplo é solicitar todas as notas e calcular a média, somando todas as notas e dividindo pela quantidade, no caso, 4; após exibir a média, analisar se a nota é maior ou igual a 7 e, em caso positivo, informar que o usuário está aprovado. Vale perceber que, caso o aluno tenha média inferior a 7, nada acontece, pois a seleção simples executa um bloco de ações apenas quando a condição analisada retorna verdadeira.

Operadores relacionais

Analise as duas condições dos exemplos de aplicação anteriores:

```
resp = "não"
```

```
media >= 7
```

Em ambos os casos, essas análises usam operadores ditos relacionais, os quais são utilizados para comparar valores, retornando sempre um resultado booleano, ou seja, verdadeiro ou falso. Em lógica de programação, esses operadores são:

Operadores relacionais

Operação	Lógica	Python	Descrição
Igual	=	==	Compara se dois dados são iguais.
Diferente	<>	!=	Compara se dois dados são diferentes.
Menor que	<	<	Compara se um valor é menor que outro.
Maior que	>	>	Compara se um valor é maior que outro.
Menor ou igual a	<=	<=	Compara se um valor é menor ou igual a outro.
Maior ou igual a	>=	>=	Compara se um valor é maior ou igual a outro.

Importa perceber que os operadores de lógica nem sempre são os mesmos utilizados nas linguagens de programação. Como o objetivo desta disciplina é trabalhar com Python, no quadro, foram listados os operadores usados em algoritmos de lógica clássica e seus correlatos em linguagem Python.

Em Python, o sinal de igual é utilizado para atribuição de valores, funcionando da mesma forma que em lógica de programação, resolvendo inicialmente o valor da operação à direita do operador e, posteriormente, atribuindo o resultado à variável à esquerda.

No caso de comparação de **igualdade**, é usado o operador de dupla igualdade, como ocorre em várias outras linguagens de programação.

Operação	**Lógica**	**Phyton**
Igual	=	==

No caso de comparação de **diferença**, usa-se o símbolo de não igual (exclamação é um operador lógico que será visto mais à frente e significa negação em Python).

Operação	**Lógica**	**Phyton**
Diferente	<>	!=

Seleção simples – implementação em Python

O comando em Python que permite a tomada de decisões é ***if***, cuja sintaxe é:

```
if <<condição>>:
    <<bloco de ações>>
```

É importante destacar que Python não apresenta delimitador específico de início e fim de blocos de ações, como outras linguagens. O que delimita os blocos de ações é a indentação. Indentar um código significa colocar todos os códigos pertencentes a um bloco com um recuo em relação ao seu comando gerador, o que normalmente é feito com dois ou quatro espaços ou uma tabulação.

No caso do comando ***if***, as linhas que seguem o sinal de dois-pontos que pertencem ao bloco de ações devem estar recuadas. Após o primeiro retorno de linha para o mesmo nível do comando ***if***, ocorre o fim do bloco de ações.

Exemplo de aplicação 3: Elabore um programa que solicite ao usuário seu ano de nascimento e calcule sua idade. Para ser mais assertivo, também deve perguntar se o usuário já fez aniversário neste ano e analisar a influência dessa informação no cálculo

da idade.

```
1. ano_atual = 2020
2. nascimento = int(input("Qual é o seu ano de nascimento? "))
3. idade = ano_atual - nascimento
4. resp = input("Você já fez aniversário neste ano? ")
5. if resp == "Não":
6.     idade -= 1
7. print("Sua idade é", idade)
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExAplic04.py
Qual é o seu ano de nascimento? 1976
Você já fez aniversário neste ano? Não
Sua idade é 43


Process finished with exit code 0
```

Neste exemplo, na linha 05, é implementada a condicional. Em Python, não há necessidade do uso de parênteses nas condições, como ocorre em outras linguagens de programação. Aqui, é apresentado o uso da dupla igualdade como operador de comparação. Após a análise da condição, o símbolo ":" indica que se inicia o bloco de ações, caso a condição analisada resulte em verdade. Esse bloco de ações começa e termina na linha 06, uma vez que é a única linha indentada após o comando ***if***. Na linha 07, o programa volta para sua linha de execução normal.

Outro ponto importante a ser destacado é a forma compacta do operador de subtração, o qual aparece na linha 06.

```
idade -= 1
```

Isso é equivalente a:

```
idade = idade - 1
```

Essa forma reduzida de operar valores numéricos está presente em todos os operadores aritméticos.

Exemplo de aplicação 4: Elabore um programa que solicite ao usuário as notas de determinada disciplina escolar e calcule a média. Se a média for maior ou igual a 7, deve mostrar ao aluno que ele foi aprovado.

```
1. print("Digite as 4 notas bimestrais da disciplina: ")
2. nota1 = float(input())
3. nota2 = float(input())
4. nota3 = float(input())
5. nota4 = float(input())
6. media = (nota1 + nota2 + nota3 + nota4) / 4
7. print("Sua média na disciplina foi", media)
8. if media >= 7:
9.     print("Você está aprovado na disciplina!")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExAplic04.py
Digite as 4 notas bimestrais da disciplina:
8
9
7
9
Sua média na disciplina foi 8.25
Você está aprovado na disciplina!


Process finished with exit code 0
```

Neste exemplo, na linha 01, é exibida a mensagem para que sejam digitadas as quatro notas. Nas linhas 02 a 05, a função *input* é usada apenas para armazenar os valores, uma vez que tem o texto de exibição como um parâmetro opcional, conforme visto na Unidade de Estudo 1. Na linha 08, é efetuada a avaliação condicional, usando o operador “>=” para fazer uma comparação aritmética com o valor calculado na média.

Neste exemplo, caso a média calculada seja inferior a 7, nenhuma mensagem será exibida, o que dá uma sensação de código incompleto. Essa sensação será resolvida com o uso de seleção composta.

Seleção composta

Na seleção composta, além de executar um bloco de ações quando a condição retorna verdadeira, é executado um bloco de ações para o caso de a condição resultar em falsidade.

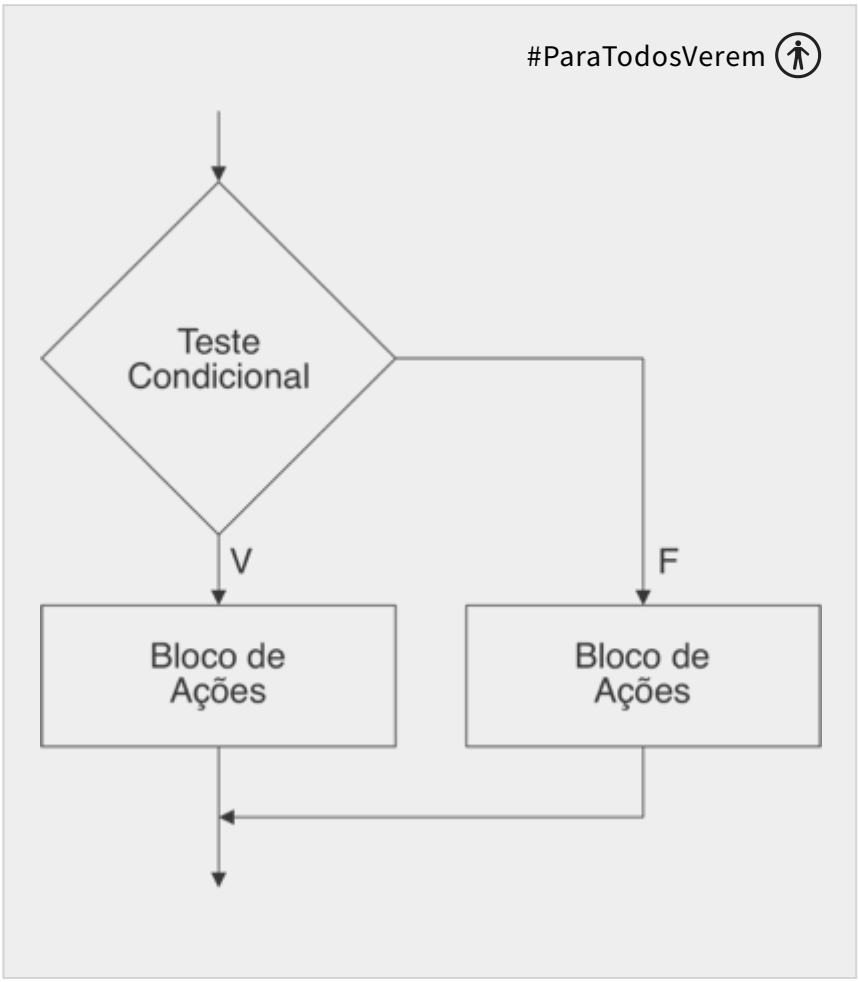


Figura 3. Representação em diagrama de fluxo da seleção composta

Estendendo o exemplo de aplicação 2 temos:

Exemplo de aplicação 5: Elabore um programa que solicite ao usuário as notas de determinada disciplina escolar e calcule a média. Se a média for maior ou igual a 7, deve mostrar ao aluno que ele foi aprovado; caso contrário, que foi reprovado.

```
1. início
2.     real: nota1, nota2, nota3, nota4, media;
3.     escreva ("Digite as 4 notas bimestrais da disciplina:
   ");
4.     leia (nota1, nota2, nota3, nota4);
5.     media = (nota1 + nota2 + nota3 + nota4) / 4;
6.     escreva ("Sua média na disciplina foi ", media);
7.     se (media >= 7)
8.         então
9.             escreva ("Você está aprovado na disciplina!");
10.        senão
11.            escreva ("Você reprovou na disciplina!");
```

```
12.     fimse;
13. fim.
```

Seleção composta – implementação em Python

A sintaxe do comando ***if*** implementando a seleção composta é:

```
if <<condição>>:
    <<bloco de ações caso retorne verdade>>
else:
    <<bloco de ações caso retorne falsidade>>
```

Na seleção composta, caso a condição seja verdadeira, um bloco de ações deverá estar indentado até encontrar a cláusula ***else***, que começará a execução de um bloco de ações caso a condição seja falsa. Da mesma forma, esse bloco de ações será delimitado pela indentação, conforme padrão do Python.

Exemplo de aplicação 6: Elabore um programa que solicite ao usuário as notas de determinada disciplina escolar e calcule a média. Se a média for maior ou igual a 7, deve mostrar ao aluno que ele foi aprovado; caso contrário, que foi reprovado.

```
1. print("Digite as 4 notas bimestrais da disciplina: ")
2. nota1 = float(input())
3. nota2 = float(input())
4. nota3 = float(input())
5. nota4 = float(input())
6. media = (nota1 + nota2 + nota3 + nota4) / 4
7. print("Sua média na disciplina foi", media)
8. if media >= 7:
9.     print("Você está aprovado na disciplina!")
10. else:
11.     print("Você reprovou na disciplina!")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExAplic06.py
Digite as 4 notas bimestrais da disciplina:
4
5
6
7
Sua média na disciplina foi 5.5
Você reprovou na disciplina!

Process finished with exit code 0
```

Seleção encadeada

A seleção encadeada é utilizada quando existe mais de uma condição a ser avaliada na tomada de decisão. Neste caso, deve ser usado mais de um comando **se** aninhado.

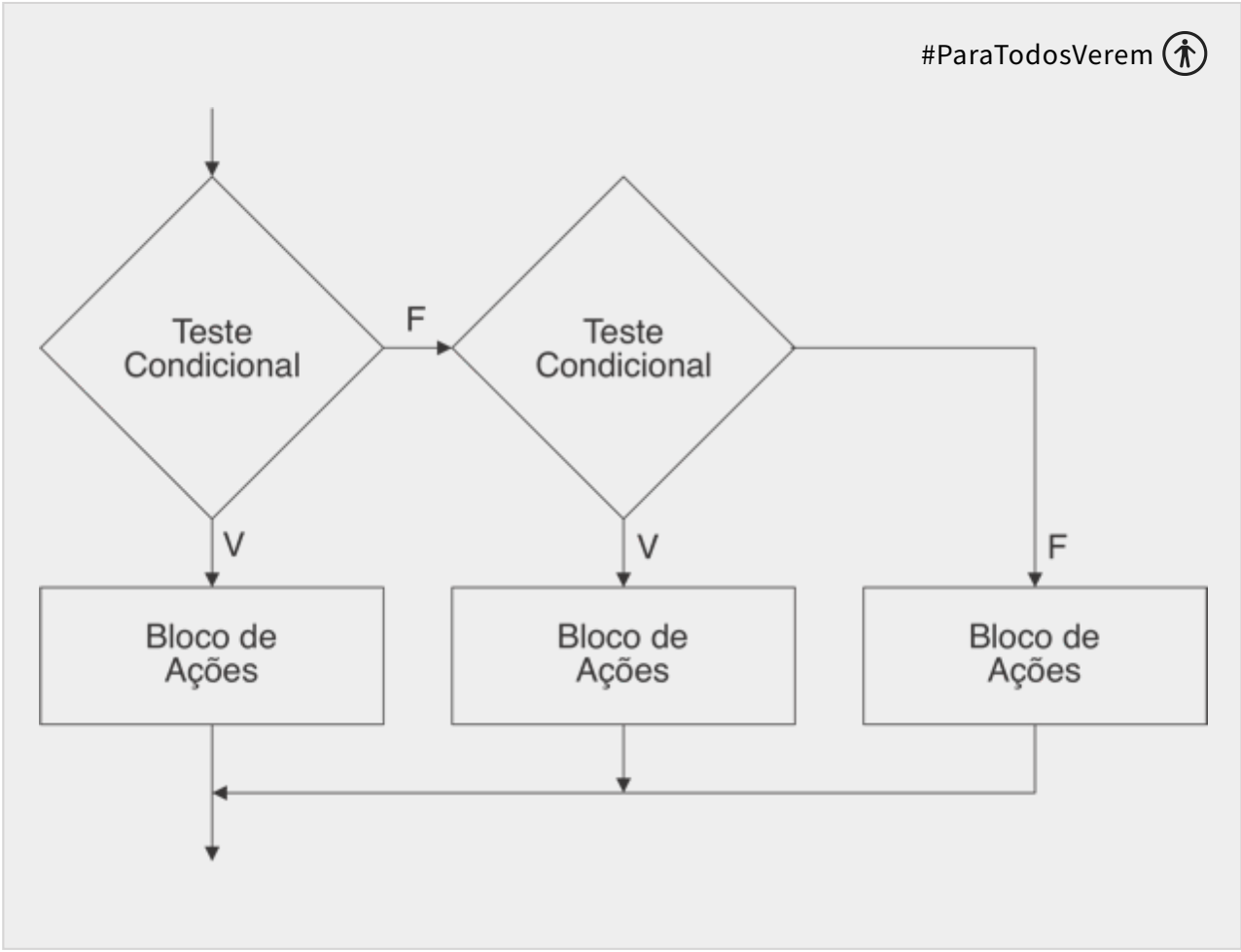


Figura 4. Representação em diagrama de fluxo da seleção encadeada

Exemplo de aplicação 7: Elabore um programa que solicite três números diferentes ao usuário e informe qual deles é o menor.

```
início
    inteiro: num1, num2, num3;
    escreva ("Digite três números diferentes: ");
    leia (num1, num2, num3);
    se (num1 < num2)
        então
            se (num1 < num3)
                então
                    escreva ("O primeiro número é o menor!");
                senão
                    escreva ("O terceiro número é o menor!");
            fimse;
        senão
            se (num2 < num3)
                então
                    escreva ("O segundo número é o menor!");
                senão
                    escreva ("O terceiro número é o menor!");
            fimse;
    fimse;
```

Neste caso, caso num1 seja menor que num2, deverá ser feita uma nova análise com relação a num3. No caso de num2 ser menor que num1, a mesma análise de num3 deverá ser efetuada. Dessa forma, todas as condições serão cobertas para a solução do problema.

Fazendo uso de seleção encadeada, qualquer número de condições pode ser avaliado para uma tomada de decisão.

Seleção encadeada – implementação em Python

A seleção encadeada em Python pode ser feita de duas formas: encadeando vários comandos ***if*** em uma mesma tomada de decisão ou utilizando a cláusula ***elif***. Segue a primeira solução.

Exemplo de aplicação 8: Elabore um programa que solicite três números diferentes ao usuário e informe qual deles é o menor (utilizar aninhamento de comandos ***if***).

```
1. print("Digite três números diferentes: ")
2. num1 = float(input())
3. num2 = float(input())
4. num3 = float(input())
5. if num1 < num2:
6.     if num1 < num3:
7.         print("O primeiro número é o menor!")
8.     else:
9.         print("O terceiro número é o menor!")
10. else:
11.     if num2 < num3:
12.         print("O segundo número é o menor!")
13.     else:
14.         print("O terceiro número é o menor!")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExAplic06.py
Digite três números diferentes:
11
67
33
O primeiro número é o menor!


Process finished with exit code 0
```

Neste exemplo, é importante destacar que as novas condicionais, nas linhas 06 e 11, devem estar indentadas em relação à condicional principal (linha 05), ou seja, caso a primeira condição seja verdadeira, seu bloco de ações vai da linha 06 à linha 09, enquanto, caso seja falsa, seu bloco de ações vai da linha 11 à linha 14.

A segunda forma de solucionar esse problema em Python é usando a cláusula ***elif***, como segue.

Exemplo de aplicação 9: Elabore um programa que solicite três números diferentes ao usuário e informe qual deles é o menor (utilizar a cláusula ***elif*** do comando ***if***).

```
1. print("Digite três números diferentes: ")
2. num1 = float(input())
3. num2 = float(input())
4. num3 = float(input())
```

```
5. if num1 < num2 and num1 < num3:
6.     print("O primeiro número é o menor!")
7. elif num2 < num1 and num2 < num3:
8.     print("O segundo número é o menor!")
9. else:
10.     print("O terceiro número é o menor!")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExAplic06.py
Digite três números diferentes:
43
32
11
O terceiro número é o menor!


Process finished with exit code 0
```

Neste exemplo, é utilizado apenas um comando ***if***, porém com mais de uma análise de condição em função da cláusula ***elif***. Para a lógica ficar mais simples, as condições das linhas 05 e 07 fazem uso do operador lógico ***and***, que indica que ambas devem ser verdadeiras para que o conjunto resulte em verdade. Os operadores lógicos serão estudados na sequência.

Operadores lógicos

Os operadores lógicos são responsáveis por realizar operações booleanas entre condições, ou seja, verificar o resultado de operações entre verdadeiro e falso. Vamos analisar a linha 05 do exemplo de aplicação 9:

```
if num1 < num2 and num1 < num3:
```

Usando os valores do exemplo (43, 32, 11), num1 < num2 resulta em falsidade, o mesmo que num1 < num3. Logo:

```
if Falso and Falso:
```

Essa operação resulta em falsidade. O mesmo ocorre para a linha 07:

```
elif num2 < num1 and num2 < num3:
```

Logo, é executado o bloco da cláusula ***else***, na linha 10.

Segue a lista de operadores lógicos:

Operadores lógicos

Operação	**Lógica**	**Python**	**Descrição**
E	E	*and*	Basta uma condição falsa para retornar falso.
OU	OU	*or*	Basta uma condição verdadeira para retornar verdadeiro.
NÃO	NÃO	*not*	Nega qualquer condição ou conjunto de condições.

O operador ***and*** tem o mesmo sentido de analisar se várias condições são verdadeiras **ao mesmo tempo** para que resulte em verdade. Caso contrário, resulta em falsidade. Segue a tabela-verdade deste operador:

Tabela-verdade do operador *and*

p	**q**	**p *and* q**
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Falso
Falso	Verdadeiro	Falso
Falso	Falso	Falso

O operador ***or*** tem o mesmo sentido de analisar se uma condição de várias é verdadeira para que resulte em verdade. Para resultar em falsidade, todas as condições devem ser falsas. Segue a tabela-verdade deste operador:

Tabela-verdade do operador *or*

p	q	p *or* q
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso

Por fim, o operador ***not*** inverte qualquer resultado de condição. Segue a tabela-verdade deste operador:

Tabela-verdade do operador *not*

p	*not* p
Verdadeiro	Falso
Falso	Verdadeiro

EXERCÍCIO

Seleção simples

Exercício de fixação 1: Crie um programa que pergunte a idade do usuário. Caso seja maior de idade, deve mostrar uma mensagem informando que pode se inscrever para fazer o teste para tirar a carteira de motorista.

 Resolução do exercício 

Resolução exercício 1

```
1. idade = int(input("Qual é a sua idade? "))
2. if idade >= 18:
```

```
3. print("Você pode se inscrever para fazer o
   teste de motorista!"
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix01.py
Qual é a sua idade? 21
Você pode se inscrever para fazer o teste
de motorista!


Process finished with exit code 0
```

Exercício de fixação 2: Crie um programa que pergunte o nome do cliente para ser inserido em um cartão de crédito, com espaço máximo de 20 caracteres. Caso o usuário informe um nome maior, deve mostrar uma mensagem informando que o nome é extenso demais e deve ser abreviado. **Dica:** para saber o tamanho de uma *string*, usar a função ***len***. Exemplo: len(nome).

 Resolução do exercício 

Resolução exercício 2

```
1. nome = input("Qual é o nome a ser inserido no
   cartão de crédito? ")
2. if len(nome) > 20:
3.     print("Seu nome é muito longo. Por favor,
   abrevie.")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix02.py
Qual é o nome a ser inserido no cartão de
crédito? João Carlos da Silva Gomes
Seu nome é muito longo. Por favor,
abrevie.

Process finished with exit code 0
```

Seleção composta

Exercício de fixação 3: Crie um programa que solicite um número ao usuário e informe se é par ou ímpar. **Dica:** para saber se um número é par ou ímpar, calcular o resto da divisão desse número por 2 (operador %). Se o resultado for 0, o número será par; se for 1, será ímpar.

 Resolução do exercício 

Resolução exercício 3

```
1. num = int(input("Digite um número: "))
2. if num % 2 == 0:
3.     print("Este número é par.")
4. else:
5.     print("Este número é ímpar.")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix03.py
Digite um número: 3
Este número é ímpar.

Process finished with exit code 0
```

Exercício de fixação 4: Crie um programa que solicite ao usuário o seu salário. Se o valor for inferior a R$ 5.000, calcule um abono de fim de ano de 15%. Caso contrário, o abono será de 10%. Informe ao usuário seu valor de abono de fim de ano.

 Resolução do exercício 

Resolução exercício 4

```
1. salario = float(input("Qual é o seu salário?
   "))
2. abono = 0
3. if salario < 5000:
4.     abono = 5000 * 0.15
5. else:
6.     abono = 5000 * 0.1
7. print(f"Valor do seu abono de fim de ano: R$
   {abono:.2f}")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix04.py
Qual é o seu salário? 1000
Valor do seu abono de fim de ano: R$
750.00

Process finished with exit code 0
```

Seleção encadeada

Exercício de fixação 5: Crie um programa que pergunte ao usuário em que turno trabalha. Formato da entrada: M – manhã, T – tarde ou N – noite. Mostre a mensagem “Bom dia!”, “Boa tarde!”, “Boa noite!” ou “Valor inválido!”, conforme o caso.

 Resolução do exercício 

Resolução exercício 5

```
1. turno = input("Qual é o seu turno de trabalho
   (M/T/N)? ")
2. if turno == "M":
3.     print("Bom dia!")
4. elif turno == "T":
5.     print("Boa tarde!")
6. elif turno == "N":
7.     print("Boa noite!")
8. else:
9.     print("Valor inválido!")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix05.py
Qual é o seu turno de trabalho (M/T/N)? N
Boa noite!

Process finished with exit code 0
```

Exercício de fixação 6: Crie um programa que solicite ao usuário dois números e a operação que deseja executar entre eles. Mostre o resultado dessa operação no formato: num1 op num2 = resultado.

 Resolução do exercício 

Resolução exercício 6

```
1. num1 = int(input("Digite o primeiro número:
   "))
2. num2 = int(input("Digite o segundo número:
   "))
3. op = input("Qual operação deseja realizar ( +
   - * / )? ")
4. result = 0
5. if op == "+":
6.     result = num1 + num2
7. elif op == "-":
8.     result = num1 - num2;
9. elif op == "*":
10.    result = num1 * num2
11. elif op == "/":
12.    result = num1 / num2
13. else:
14.    op = "erro"
15. if op == "erro":
16.    print("Operação inválida!")
17. else:
18.    print(num1,op,num2,"=",result)
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix06.py
Digite o primeiro número: 3
Digite o segundo número: 4
Qual operação deseja realizar ( + - * / )?
*
3 * 4 = 12

Process finished with exit code 0
```

Operadores lógicos

Exercício de fixação 7: Crie um programa que pergunte o tamanho de três lados de um triângulo e informe o tipo de triângulo, a saber:

a. Só será um triângulo quando a soma de dois lados sempre for maior que o terceiro lado.
b. Equilátero: triângulo com todos os lados iguais.
c. Isósceles: triângulo com dois lados iguais.
d. Escaleno: triângulo com todos os lados diferentes.

 Resolução do exercício 

Resolução exercício 7

```
1. print("Informe os três lados de um triângulo:
   ")
2. lado1 = int(input())
3. lado2 = int(input())
4. lado3 = int(input())
5. if lado1 + lado2 < lado3 or lado2 + lado3 <
   lado1 or lado3 + lado1 < lado2:
6.     print("Estes lados não formam um
   triângulo.")
7. else:
8.     if lado1 == lado2 and lado1 == lado3:
9.         print("É um triângulo equilátero")
```

```
10.     elif lado1 == lado2 or lado2 == lado3 or
    lado3 == lado1:
11.         print("É um triângulo isósceles.")
12.     else:
13.         print("É um triângulo escaleno.")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix07.py
Informe os três lados de um triângulo:
3
4
5
É um triângulo escaleno.

Process finished with exit code 0
```

Exercício de fixação 8: Crie um programa que solicite ao usuário as quatro notas bimestrais de uma matéria e o número de faltas. Informe se o aluno foi aprovado ou reprovado, bem como o motivo, a saber:

a. A média anual é 7.
b. A disciplina possui 40 aulas.
c. O mínimo exigido é 75% de presença.

 Resolução do exercício 

Resolução exercício 8

```
1. print("Digite as quatro notas bimestrais da
   disciplina: ")
2. nota1 = float(input())
3. nota2 = float(input())
4. nota3 = float(input())
5. nota4 = float(input())
6. faltas = int(input("Digite o número de
   faltas: "))
7. media = (nota1 + nota2 + nota3 + nota4) / 4
8. print("Sua média na disciplina foi", media)
9. pres = 100 - (faltas * 100) / 40
```

```
10. print("Sua porcentagem de presença é de",
    pres, "%")
11. if media >= 7 and pres >= 75:
12.     print("Você está aprovado na
    disciplina!")
13. elif media >= 7 and pres < 75:
14.     print("Você foi reprovado por faltas!")
15. elif media <=7 and pres >= 75:
16.     print("Você foi reprovado por nota!")
17. else:
18.     print("Você foi reprovado por nota e por
    faltas!")
```

C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix08.py
Digite as quatro notas bimestrais da
disciplina:
5
5
5
5
Digite o número de faltas: 13
Sua média na disciplina foi 5.0
Sua porcentagem de presença é de 67.5%
Você foi reprovado por nota e por faltas!

Process finished with exit code 0
```

Desafio

Exercício de fixação 9: Crie um programa que simule um caixa eletrônico. O programa deverá perguntar ao usuário o valor do saque e depois informar quantas notas de cada valor serão fornecidas, a saber:

a. Notas disponíveis: 1, 5, 10, 50 e 100 reais.
b. Valor mínimo de saque: R$ 10,00.
c. Valor máximo de saque: R$ 600,00.

 Resolução do exercício 

Resolução exercício 9

```
valor = int(input("Qual valor deseja sacar? "))
resto = valor
#Inicia número de notas em zero
n1 = n5 = n10 = n50 = n100 = 0
if valor < 10 or valor > 600:
    print("Saque menor que o valor mínimo ou
excede o valor máximo.")
else:
    if resto >= 100:
        sobra = resto % 100
        n100 = int((resto - sobra) / 100)
        resto = sobra
    if resto >= 50:
        sobra = resto % 50
        n50 = int((resto - sobra) / 50)
        resto = sobra
    if resto >= 10:
        sobra = resto % 10
        n10 = int((resto - sobra) / 10)
        resto = sobra
    if resto >= 5:
        sobra = resto % 5
        n5 = int((resto - sobra) / 5)
        resto = sobra
    n1 = resto
    print("Notas de 100:", n100)
    print("Notas de 50:", n50)
    print("Notas de 10:", n10)
    print("Notas de 5:", n5)
    print("Notas de 1:", n1)
```

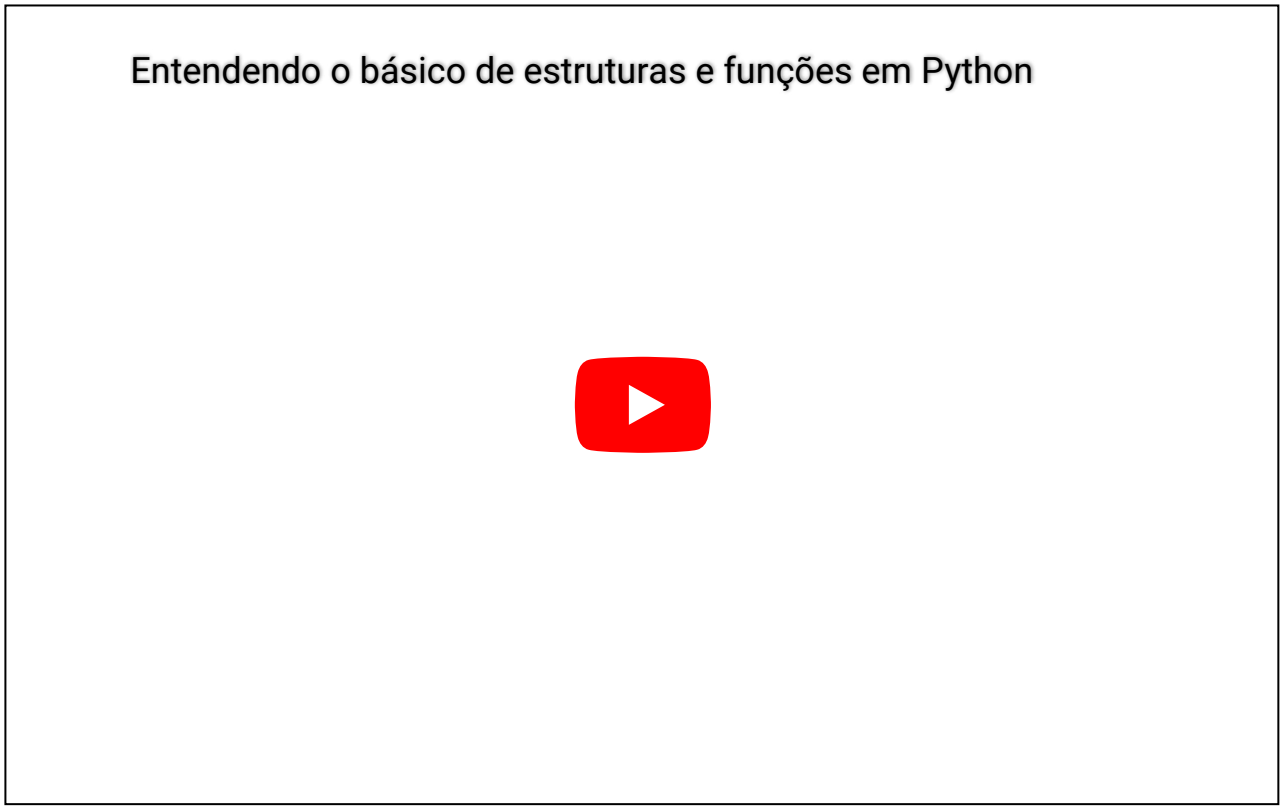
C:\

```
/Users/user/untitled1/bin/python
/Users/user/projects/untitled1/ExFix09.py
Qual valor deseja sacar? 597
Notas de 100: 5
Notas de 50: 1
Notas de 10: 4
Notas de 5: 1
Notas de 1: 2

Process finished with exit code 0
```

Entendendo o básico de estruturas e funções em Python

Neste vídeo, vamos aprender a criar funções. Veremos a estrutura de um sistema em Python e como se divide código nessa linguagem.



Conclusão

A respeito do uso de estruturas condicionais na programação em Python, podemos concluir que:

- As estruturas condicionais permitem que sejam tomadas decisões dentro dos programas.
- Em lógica de programação, o comando condicional é o comando ***se***.
- Na linguagem Python, o comando condicional é o comando ***if***.
- Para as análises condicionais, são utilizados os operadores relacionais.
- As estruturas condicionais podem ser de seleção simples, seleção composta ou seleção encadeada.

- Seleção simples analisa uma condição e executa um bloco de ações, caso resulte em verdade.
- Seleção composta analisa uma condição e executa um bloco de ações caso resulte em verdade e outro bloco de ações caso resulte em falsidade.
- Seleção encadeada analisa mais de uma condição por aninhamento de condicionais ou, em Python, pelo uso da cláusula ***elif*** do comando ***if***.
- Condições podem ser analisadas em conjunto a partir dos operadores lógicos.

Referências

BANIN, S. L. **Python 3**: conceitos e aplicações uma abordagem didática. São Paulo: Érica, 2018.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. **Lógica de programação**: a construção de algoritmos e estruturas de dados. São Paulo: Prentice Hall, 2005.

PYTHON SOFTWARE FOUNDATION. **Documentação Python 3.9.0**. Disponível em: https://docs.python.org/pt-br/3/. Acesso em: 27 out. 2020.