Curso Técnico Nível Médio Subsequente

Informática Para Internet

Fundamentos de Lógica e Algoritmos

Aula 07

Estruturas de Decisão e Introdução às Estruturas de Repetição

Bruno Emerson Gurgel Gomes

Instituto Federal de Educação, Ciência e Tecnologia
do Rio Grande do Norte
Natal-RN
2015

Presidência da República Federativa do Brasil

Ministério da Educação

Secretaria de Educação a Distância

Este Caderno foi elaborado em parceria entre o Instituto Federal de Educação, Ciência e Tecnologia e o Sistema Escola Técnica Aberta do Brasil – e-Tec Brasil.

Equipe de Elaboração
Cognitum

Coordenação Institucional
COTED

Professor-autor
Thiago Medeiros Barros

Projeto Gráfico
Eduardo Meneses e Fábio Brumana

Diagramação
Yann Valber

Ficha catalográfica

G633c Gomes, Bruno Emerson Gurgel.

Curso Técnico Nível Médio Subsequente Informática para Internet : Fundamentos de Lógica e Algoritmos - Aula 07 : Estruturas de decisão e introdução às estruturas de repetição / Bruno Emerson Gurgel Gomes. – Natal : IFRN Editora, 2015.

24 f. : il. color.

1. Fundamentos de Lógica e Algoritmos - EaD. 2. Controle de fluxo - Programação. 3. Estrutura de decisão. 4. Estrutura de repetição. I. Título.

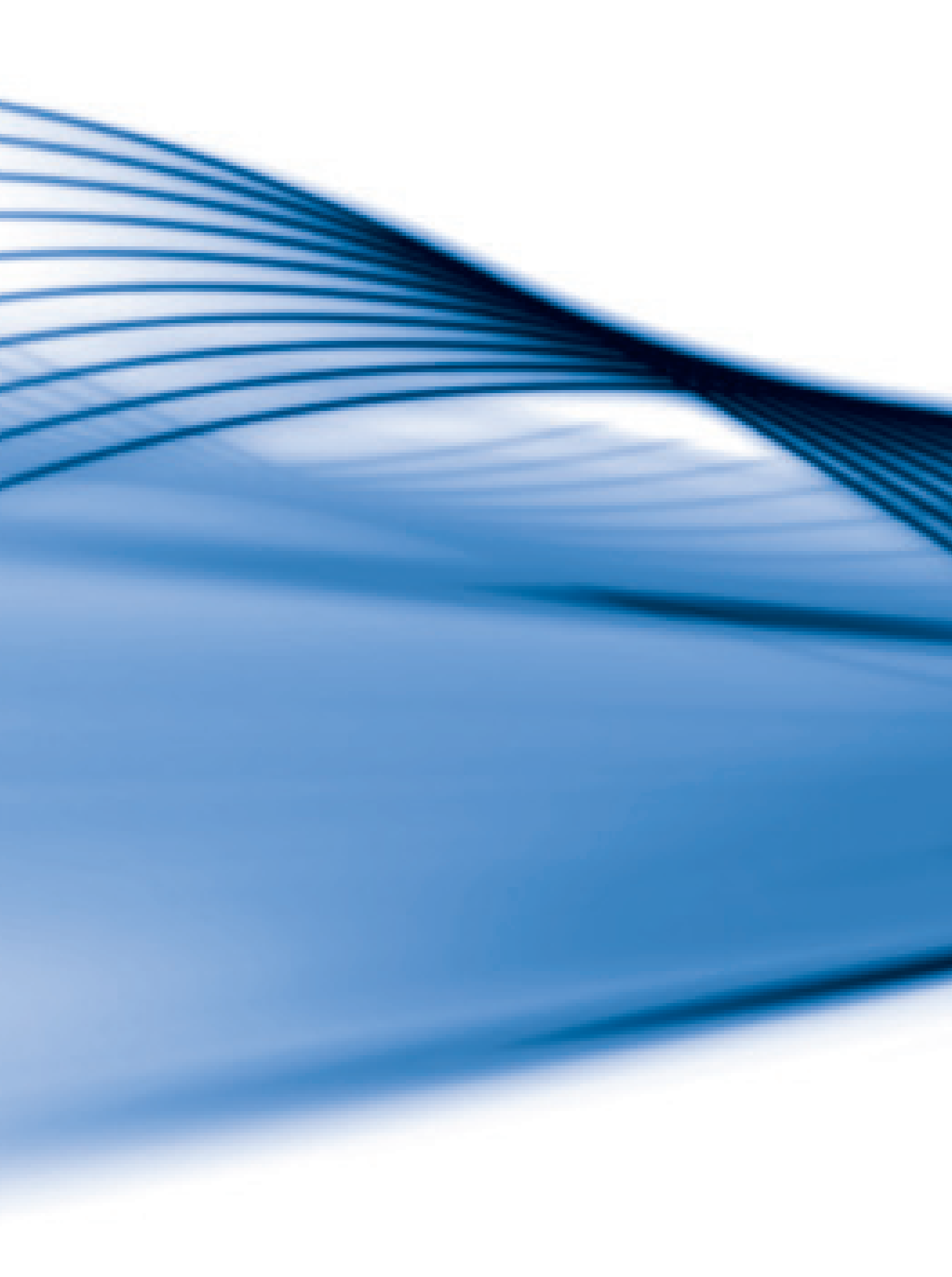
RN/IFRN/EaD CDU 004.421

Ficha elaborada pela bibliotecária Edineide da Silva Marques, CRB 15/488

Apresentação da disciplina

Olá, aluno(a), seja bem-vindo(a) a mais uma aula de programação em Python. Aprendemos até aqui a reconhecer e trabalhar com valores dos diversos tipos de dados e a armazenar esses valores em variáveis usando o sinal de atribuição (=). Vimos também como interagir com o usuário da sua aplicação através da leitura de valores com o teclado, usando o comando input e a imprimir mensagens que serão exibidas no monitor do computador através do comando print.

Nesta aula, vamos dar um passo adiante e aprender a trabalhar com estruturas que são extremamente importantes na programação de computadores. Iremos começar pelas estruturas de decisão, que como o nome sugere, permitem desviar o fluxo de execução do programa com base em uma decisão a ser tomada. Posteriormente, vamos começar a explorar a repetição de instruções. Juntas, essas duas estruturas permitem resolver uma vasta gama de problemas. Então, vamos nessa.



Aula 07 - Estruturas de Decisão e Introdução às Estruturas de Repetição.

Objetivos

Esta aula tem por objetivo apresentar as estruturas de decisão e realizar uma breve introdução às estruturas de repetição. Esse último tópico será explorado em maiores detalhes na aula seguinte. Ao final da aula, o aluno deve estar apto a:

- Compreender os princípios do desvio do fluxo de instruções de um programa através do uso das estruturas de decisão e repetição;
- Reconhecer e criar blocos de instruções;
- Resolver uma gama maior de problemas através do uso da estrutura de decisão IF (se) e suas variações.

Desenvolvendo o conteúdo

Controle do fluxo de execução do programa

Até o presente momento, nossos programas são formados por uma sequência de instruções simples que são interpretadas uma depois da outra, em um fluxo linear, sem nenhum desvio. Como ponto de partida, vamos resolver o seguinte problema: "Calcular o consumo médio de um veículo a partir da quantidade de combustível em litros que ele gasta para percorrer uma determinada distância em quilômetros (km)". Com o que sabemos até agora, é possível resolver esse problema apenas usando instruções em sequência. Primeiro, é preciso ler os dados de entrada do teclado, depois calcular a quantidade de combustível gasta e, por fim, gerar uma saída compreensível para o usuário do programa.

Pois bem, vamos resolvê-lo por partes. Primeiro, é preciso saber da distância percorrida pelo veículo e da quantidade de combustível gasta. Nesse caso, como essas informações podem mudar de acordo com cada usuário do programa, elas devem ser fornecidas por ele. Assim, devemos usar o comando *input* e guardar as informações em variáveis, conforme o Código 1. Do lado esquerdo do código, inserimos a numeração das linhas para facilitar a explicação do programa. Essa numeração não deve ser inserida no interpretador *Python*.

Código 1 – Consumo médio de combustível: leitura dos dados

1. >>> dist = float (input (“Distância percorrida: “))
2. Distância percorrida: 45.0
3. >>> comb = float (input (“Combustível gasto: “))
4. Combustível gasto: 3.2

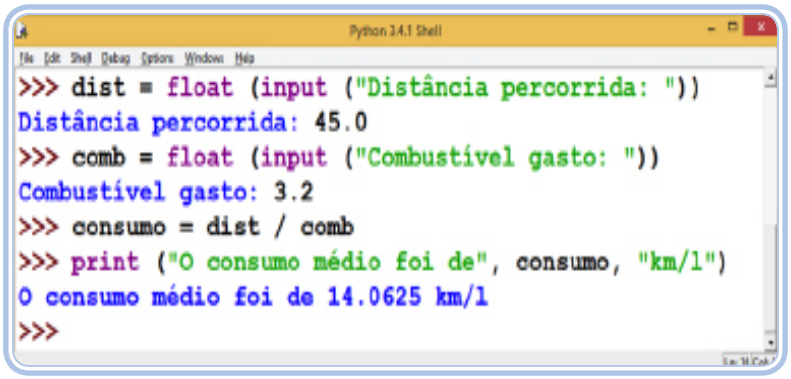
No código acima, na linha 1, a variável dist recebe a informação da distância percorrida pelo veículo. Ao pressionar o botão ENTER do teclado, o programa pede ao usuário que ele digite esse valor. Nesse caso, o valor inserido foi 45.0, representando 45.0 quilômetros. A quantidade de combustível gasta é armazenada na variável *comb*. Nesse caso, na linha 4, o valor lido foi 3.2, representando 3.2 litros de combustível.

Uma vez que já temos as informações que o programa necessita, estamos aptos a calcular a quantidade média de combustível gasta, que é o objetivo do programa. O Código 2 exibe esse cálculo e a informação do resultado do programa para o usuário.

Código 2 – Consumo médio de combustível: cálculo e impressão de resultados

1. >>> consumo = dist / comb
2. >>> print (“O consumo médio foi de”, consumo, “km/l”)
3. O consumo médio foi de 14.0625 km/l

O consumo médio do veículo é salvo na variável consumo (linha 1), sendo o resultado do cálculo da distância percorrida dividida pelo combustível consumido para percorrer essa distância (dist / comb). Após o cálculo, um comando print da linha 2 cria a mensagem que será exibida para o usuário do programa. O resultado dessa mensagem, incluído o consumo calculado é exibido na linha 3. Na Figura 1, é possível ver o código completo do programa conforme exibido no interpretador.



Fonte: Autoria própria.

Figura 01: Programa do consumo médio de combustível visto no interpretador Python

Esse exemplo ilustra um programa sequencial e linear. Esse programa precisa de apenas duas informações para realizar um cálculo simples e devolver um resultado ao usuário. Agora, prezado aluno(a), suponha que esse mesmo programa pudesse ter a possibilidade de escolher entre fornecer o resultado do cálculo em quilômetros por litro (km/l) ou em metros por litro (m/l). Nesse caso, precisaríamos tomar uma decisão para imprimir a mensagem ao usuário. Para desviar o fluxo do programa a partir de uma decisão, devemos usar uma estrutura da decisão.

Estruturas de decisão (ou seleção)

Olá, Certamente, você já deve ter tomado alguma decisão importante em sua vida, não é mesmo? Em verdade, tomamos pequenas decisões diariamente. Onde vamos almoçar hoje? Vamos sair para o cinema ou ficar em casa assistindo televisão? Hoje é dia de estudar programação ou matemática? Toda decisão tem por base um julgamento de valor.

Por exemplo, podemos decidir se vamos ao restaurante A ou B a partir da preferência pela comida de um deles. É possível decidir se vamos ao cinema com base na quantidade de dinheiro que nós temos. Esse dinheiro será sufi-

ciente para pagar o cinema e a pipoca? Ele não fará falta para pagar alguma conta?

Fonte: VALBER (2015)

Figura 02: Tirinha "Decisão"

Pois bem, em um programa de computador por diversas vezes temos que tomar uma decisão ou realizar um teste. Essa decisão ou teste irá guiar a execução de um trecho do código do programa a partir da análise do seu resultado. Por exemplo, é possível testar se o valor fornecido pelo usuário é um número antes de se realizar uma conta matemática. Ou podemos perguntar se um valor obedece a um certo critério, como a idade de uma pessoa ser maior ou igual a 18 anos.

Assim, estamos prontos para apresentar a forma básica da estrutura de decisão IF na linguagem Python. Essa estrutura também é conhecida como estrutura de seleção ou simplesmente "comando SE", sendo "se" o significado em português de *if*.

Estrutura de decisão IF ("se")

Permite executar um conjunto de comandos a partir da avaliação de uma condição. Se a condição for avaliada verdadeira, os comandos dentro do corpo do IF são executados. Caso contrário, os comandos dentro do corpo do IF não são executados. Em ambos os casos, a execução do programa prossegue após o comando IF.

A forma (sintaxe) do comando IF em Python é a seguinte:

```
if condição :

        comandos
```

Na instrução IF, a condição é uma expressão que deve devolver verdadeiro *(True)* ou falso *(False)* e *comandos* corresponde a zero ou mais comandos que devem ser executados caso a condição do IF seja avaliada verdadeira.

LEMBRE-SE!

Uma expressão é uma constante, uma variável ou uma combinação de variáveis, constantes e operadores que após ser avaliada devolve um valor para o usuário.

Antes de prosseguirmos com um exemplo, há algumas considerações a se fazer nesse momento. Primeiro, a palavra IF no comando deve sempre ser grafada em letra minúscula. Segundo, o sinal de dois pontos (":") após a condição marca o início de um bloco de comandos. Mas, o que vem a ser um bloco de comandos?

Bloco de comandos

Um bloco de comandos corresponde a um ou mais comandos que estão definidos dentro de uma determinada estrutura. Em Python, um bloco de comandos é delimitado por identação (ou endentação), ou seja, um recuo que você deve realizar através de tabulações ou espaços em branco. Essas tabulações ou espaços devem ser consistentes. Assim, um comando (ou instrução) que está no mesmo bloco do comando anterior deve estar na outra linha, mas alinhado com o seu antecessor através do mesmo número de tabulações ou espaços.

Pode parecer muita informação até o momento, mas vamos explicar em detalhes cada um desses conceitos através de um exemplo simples. O programa do Código 3 pede a leitura de dois números e informa qual deles é o maior. Esse programa foi adaptado de (MASANORI, 2015).

Código 3 – Exibe o maior entre dois números fornecidos pelo usuário

```
1. a = int ( input ("Primeiro número: ") )
2. b = int ( input ("Segundo número: ") )
3. if a > b :
4.     print ("O primeiro número é maior")
5. if b > a :
6.     print ("O segundo número é maior")
```

Nas linhas 1 e 2 do Código 3, temos a leituras dos dois números, armazenados nas variáveis "*a*" e "*b*", respectivamente. A linha 3 exibe uma decisão com base no valor das variáveis "a" e "b". Suponha que o usuário digitou 5 para o valor de "a" e 2 para o valor de "b".

Observe que, para esses valores fornecidos (a = 5 e b = 2), a expressão (a > b) na instrução IF da linha 3 devolve o valor verdade verdadeiro (*True*), pois 5 > 2 (5 é maior que 2). Assim, de acordo com a definição do IF, o programa irá para o bloco de comandos após o sinal de dois pontos (":"). Nesse caso, temos apenas um comando, a instrução *print* da linha 4.

Correto, mas como o interpretador *Python* sabe que a instrução faz parte do bloco de comandos do IF da linha 3? Ele sabe, pois a instrução está identada, ou se preferir, está recuada com uma tabulação (tecla TAB do seu teclado) ou com espaços em branco em relação a linha anterior, que corresponde ao início do comando IF. Assim sendo, a mensagem "*O primeiro número é maior*" é impressa e o programa prossegue analisando a instrução IF da linha 5. Veja que a condição de teste do IF da linha 5 é a expressão (b > a), que irá resultar no valor verdade falso (*False*). Nesse caso, como o teste falhou, a mensagem do corpo do segundo IF não é executada e o programa termina.

Teste com "senão"

No caso do exemplo do Código 3, nós temos duas condições que são complementares. Ou seja, se uma for satisfeita, a outra não precisa ser verificada. No caso, se "a" for maior que "b", então você não precisa testar com outro IF se "b" é maior que "a". Para esses casos, a estrutura IF pode ser usada com um senão (ELSE).

Estrutura de decisão IF ("se") com a parte ELSE ("senão")

A forma (sintaxe) do comando IF em com ELSE em Python é a seguinte:

```
if condição :
    comandos_verdadeiro
else :
    comandos_falso
```

Na instrução IF com a parte ELSE (IF-ELSE), a condição é uma expressão que deve devolver verdadeiro (*True*) ou falso (*False*) e *comandos_verdadeiro* corresponde a zero ou mais comandos que devem ser executados caso a condição do IF seja avaliada verdadeira. No caso da condição ser avaliada falsa (*False*), os comandos da parte ELSE são executados (*comandos_falso*).

O Código 4, a seguir, demonstra uma versão melhorada do código 3, agora inserindo o senão (ELSE) no comando IF. Nesse caso, é possível eliminar um teste (o segundo IF).

Código 4 – Maior entre dois números com IF-ELSE

```
1. a = int ( input ("Primeiro número: ") )
2. b = int ( input ("Segundo número: ") )
3. if a > b :
4.     print ("O primeiro número é maior")
5. else :
6.     print ("O segundo número é maior")
```

No caso do Código 4, se formos descrever em português o seu funcionamento a partir da linha 3, seria: Se "a" for maior que "b", então será impressa a mensagem "O primeiro número é maior", caso contrário (senão), será impressa a mensagem "O segundo número é maior".

ATIVIDADE

1. Leia a idade de uma pessoa e imprima uma mensagem dizendo se ela é maior ou menor de idade. Considere que uma pessoa maior de idade é aquela que tem 18 anos ou mais.

2. Tendo como entrada a altura em centímetros (p.ex. 1.70m = 170cm) e o sexo de uma pessoa ("m" para masculino ou "f" para feminino), calcule o seu peso ideal de acordo com a fórmula abaixo, conhecida como fórmula de Lorentz. Nessa fórmula, o valor "k" é igual a 4 para homens e 2 para mulheres.

$$peso = altura - 100 - ((altura - 150) / k)$$

É possível ter um teste dentro do outro, usando mais de um comando IF. Isso é possível, pois o bloco de comandos do IF (instruções identadas e contidas após o ":"), como qualquer outro comando de bloco na linguagem *Python*, aceita qualquer tipo de instruções válidas na linguagem, até mesmo outros IFs.

Para exemplificar, suponha que queremos resolver o problema de saber se um número é par ou ímpar. Nesse caso, há uma restrição a mais, pois o número tem que ser maior que zero. Então, antes de testar se o número é par ou ímpar, temos que testar (inserir um IF) para saber se o número é maior que zero. A solução para este problema pode ser vista no Código 5.

Código 5 – Maior entre dois números com IF-ELSE

```
1. x = int (input ("Digite um número inteiro positivo (>0): "))
2. if x > 0 :
3.     if x % 2 == 0 :
4.         print ("O número", x, "é par!")
5.     else :
6.         print ("O número", x, "é ímpar!")
7. else :
8.     print ("O número deve ser maior que 0!")
```

Observe que o IF mais interno (linha 3) está identado em relação ao IF mais externo (linha 2) que verifica se o número é maior que zero. Nesse caso, o IF da linha 3 é parte do corpo do IF da linha 2 e será executado apenas se a condição de teste do IF da linha 2 devolver o valor verdadeiro. Caso contrário, a mensagem "O número deve ser maior que 0!" será exibida para o usuário.

No caso do número ser maior que zero, o teste do IF da linha 3 é feito. Esse

teste verifica se o resto da divisão do número lido (x) por 2 é igual a zero (x % 2 == 0). Caso essa expressão seja verdade (*True*), então o número é par, pois o número é divisível por 2, e a mensagem da linha 4 é exibida para o usuário. Caso contrário, é exibida a mensagem da parte ELSE, na linha 6, dizendo que o número é ímpar.

Testes aninhados

Outra situação que pode ocorrer é quando temos vários casos de testes aplicados a uma situação e apenas um deles deve ser executado. Nesse caso, teríamos vários testes, um dentro do outro. Por exemplo, suponha que em uma competição de natação, os nadadores sejam classificados em diversas categorias, de acordo com a sua idade. Podemos criar um programa que receba a idade de um nadador e imprima a qual categoria ele pertence. Nesse caso, temos que testar cada caso, um por vez, até que se encontre a categoria na qual o nadador se enquadra. Assim, é necessário um IF para cada categoria, conforme pode ser visto no Código 6.

Código 6 – Imprime a categoria em uma competição de natação dada a idade do nadador

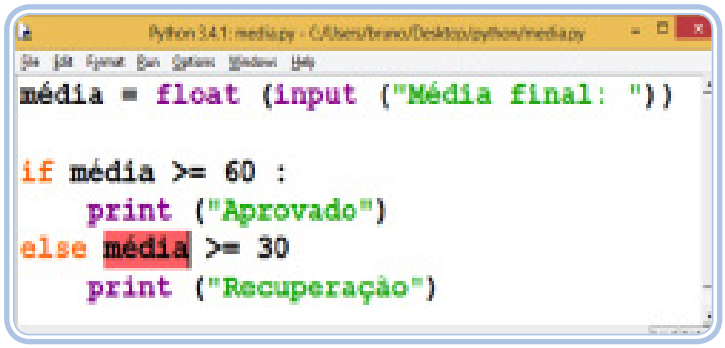
```
1. idade = int (input (“Idade: “))
2. if idade >= 5 and idade <= 7 :
3.     print (“Infantil A”)
4. else :
5.     if idade >= 8 and idade <= 10 :
6.         print (“Infantil B”)
7.     else :
8.         if idade >= 11 and idade <= 13 :
9.             print (“Juvenil A”)
```

```
10.         else :
11.             if idade >= 14 and idade <= 17 :
12.                 print (“Juvenil B”)
13.             else :
14.                 print (“Não existe categoria para a idade informada!”)
```

Prezado aluno, espero que você não tenha se assustado com o Código 6. Em verdade, apesar de um pouco extenso, ele é simples. Para cada intervalo de idade em cada categoria é feito um teste. Se a idade informada estiver dentro de uma das categorias, então a mensagem do IF correspondente é impressa. Caso contrário, a mensagem no corpo do último ELSE é impressa, informando ao usuário que não existe categoria para a idade informada.

Observe que no exemplo do Código 6 as expressões estão em pares, conectadas por um “and”, como no caso de “idade >= 8 and idade <= 10”. Isso é necessário, pois o operador de maior ou igual (>=), assim como os demais operadores de comparação, pode(m) receber apenas dois operandos, no caso a variável idade e uma constante inteira (p.ex.: idade >= 8). Não é possível fazer como na matemática, cuja expressão seria algo da forma (8 <= idade <= 10). Assim, para dizer que as duas condições têm que ser satisfeitas, nós usamos o operador “and”, que irá devolver o valor verdadeiro apenas se a idade estiver dentro do intervalo estabelecido.

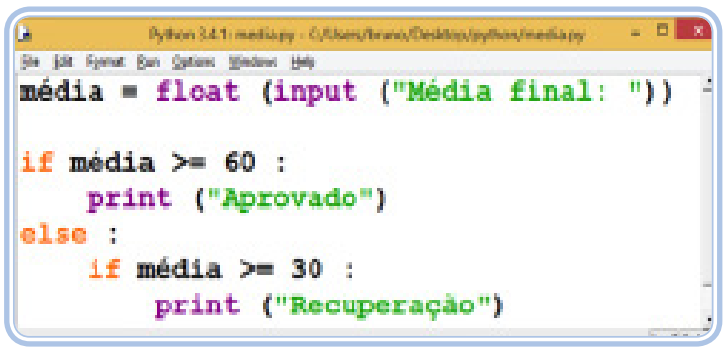
Outra situação comum que pode levar o usuário a erro, é tentar fazer um teste com a parte ELSE. É importante frisar que isso não é possível, conforme está destacado no programa da Figura 3, que diz se um aluno foi aprovado ou ficou em recuperação, dada a sua média. Isso leva a um erro de sintaxe na linguagem, pois esse tipo de construção não existe. Portanto, sempre que você precisar realizar um teste, ele deve ser feito em um IF. Por isso, no caso de um teste ser falso, na parte ELSE inserimos um outro IF para testar o próximo caso.



Fonte: Autoria própria.

Figura 03: ERRO: Tentando inserir um teste na parte ELSE

A versão correta do programa que exibe o resultado da média pode ser vista na Figura 4. Nesse caso, foi inserido mais um IF para realizar o teste que verifica se a média é maior ou igual que 30, situação em que o aluno fica em recuperação.



Fonte: Autoria própria.

Figura 04: Versão correta do programa da média, com um teste a mais inserido em um IF

Esses casos em que temos vários testes em sequência, relacionados entre si, sendo que apenas um deve ser executado, são chamados de testes aninhados. Para esse tipo de estrutura, a linguagem Python possui a palavra ELIF que abrevia a parte ELSE IF (senão-se), tornando o código menor e mais legível. Assim sendo, é possível obter no Código 7 a mesma funcionalidade apresentada no Código 6 usando ELIF com o ganho do código ficar mais enxuto e organizado para o programador.

Código 7 – Imprime as categorias em uma competição de natação (versão com ELIF)

1. idade = **int** (**input** (“Idade: “))

2. **if** idade >= 5 **and** idade <= 7 :

```
3.      print ("Infantil A")
4.  elif idade >= 8 and idade <= 10 :
5.      print ("Infantil B")
6.  elif idade >= 11 and idade <= 13 :
7.      print ("Juvenil A")
8.   elif idade >= 14 and idade <= 17 :
9.      print ("Juvenil B")
10. else :
11.     print ("Não existe categoria para a idade informada!")
```

Introdução à Repetição

Prezado aluno(a), com o repertório de técnicas e elementos da linguagem Python que vimos até agora já somos capazes de criar diversos programas que envolvam cálculos e tomada de decisões com base no valor de expressões.

Pois bem, a partir deste momento vamos iniciar no estudo de uma nova instrução que permite repetir trechos de código do nosso programa. Nesta aula, nós vamos apenas contextualizar o uso da instrução de repetição para que possamos detalhá-la na próxima aula.

Você pode estar se perguntando: mas como assim repetir código do programa? Como isso pode ser de fato aplicado? Isso não apenas pode ser aplicado como é uma ferramenta extremamente útil, que ajuda a evitar a repetição de código e possibilita resolver uma gama muito maior de problemas.

Para entender a repetição, suponha o seguinte problema: seu professor lhe pediu para que você calculasse a média aritmética das notas da sua turma na disciplina de programação. Se a turma for composta por 8 alunos, a média corresponde a soma das notas dividido por 8. Com o que vimos até

agora, como poderíamos ler todas as notas e efetuar o cálculo? Simples, nós teríamos que criar 8 variáveis, para armazenar cada nota antes de efetuar o cálculo. O código 3 demonstra a solução desse problema.

Código 8 – Média das notas de uma turma de 8 alunos

```
1.  nota1 = float (input ("Nota1: "))
2.  nota2 =  float (input ("Nota2: "))
3.  nota3 =  float (input ("Nota3: "))
4.  nota4 =  float (input ("Nota4: "))
5.  nota5 =  float (input ("Nota5: "))
6.  nota6 =  float (input ("Nota6: "))
7.  nota7 = float (input ("Nota7: "))
8.  nota8 = float (input ("Nota8: "))
9.  média=(nota1 + nota2 + nota2 + nota4 + nota5 + nota6 + nota7 +
    nota8) / 8
10. print ("Média da turma=", média)
```

Na solução do Código 9, a soma das notas foi feita logo após a leitura de cada nota. Na linha 1 desse programa, iniciamos a soma com 0. Isso foi feito devido ao papel da variável "soma" como acumulador de uma soma de vários valores, no caso as notas dos alunos. Assim, na leitura da primeira nota, a soma recebe o valor dessa nota, uma vez que a expressão fica soma = 0 + nota1, o que resulta em soma = nota1. Na linha 5, a soma recebe o valor que ela tinha antes (a nota 1), mais o valor lido para a segunda nota, e assim por diante.

Perceba que o processo de receber um valor e acumular esse valor na variável soma é bastante semelhante, da linha 2 até a linha 17. Na próxima aula, vamos ver como realizar esse mesmo programa com apenas uma variável para receber a nota dos alunos e repetir, automaticamente, esses trechos de

código para realizar o acúmulo da soma das notas na variável "soma". Para tanto, vamos ver em detalhes e como mais exemplos a estrutura de repetição while (enquanto).

RESUMINDO

Prezado aluno, nesta aula detalhamos a estrutura de decisão "se". Ela é utilizada sempre que precisamos fazer um teste sobre uma determinada expressão ou simplesmente, como o próprio nome sugere, tomar uma decisão sobre qual caminho o programa deve seguir. Em Python, a estrutura se corresponde ao comando IF. Exibimos a versão do IF com e sem a parte ELSE. No caso do IF com ELSE, os comandos que estão nessa última parte são feitos apenas se o teste da parte IF devolver o valor falso (false). Mostramos também como testar diversas condições complementares usando o IF com o ELIF, que significa ELSE-IF. Você deve lembrar também a importância de identar o código sempre que estiver em uma estrutura de bloco, tal como o IF ou ELSE. Identar o código corresponde a inserir uma tabulação (tecla TAB do teclado) ou uma certa quantidade de espaços em branco de modo que todos os comandos em um bloco fiquem na mesma identação. Isso faz com que o comando pertença aquele bloco. Por fim, fizemos uma breve introdução para motivar o uso de repetição de trechos de código, assunto esse que será detalhado na próxima aula.

LEITURAS COMPLEMENTARES

MELANDA, J. C. E. Blocos de código em Python. 2011 (http://programeempython.blog.br/blog/blocos-de-codigo-em-python/). Nesta página, você pode encontrar mais informações sobre a identação de código para estabelecer os blocos de código em Python.

D! TUTORIAIS (https://dtutoriais.wordpress.com/2010/07/20/aprendendo-python-4-estruturas-de-decisao/) – Nesta página, você pode ler mais a respeito das estruturas de decisão em Python e suas variações.

AVALIANDO SEUS CONHECIMENTOS

1. Suponha que um funcionário ganha R$ 800,00 por semana por 30 horas de trabalho. Esse mesmo funcionário recebe R$150,00 para cada hora extra de trabalho durante a semana, sendo que ele pode trabalhar no máximo até 40 horas. Faça um programa que receba a quantidade de horas trabalhas por um funcionário (horas >= 30 e horas <= 40) e informe o seu salário final naquela semana.

2. Faça um algoritmo para calcular a área de um círculo, dado o valor do seu raio, que deve ser positivo (maior que 0). A fórmula da área do círculo é:

área = π * r2 , onde π = 3.14 e "r" é o raio.

3. Faça um programa em Python para calcular a média parcial de um aluno em uma disciplina semestral de acordo com as instruções a seguir.

- O programa deve informar se o estudante foi aprovado, ficou em avaliação final ou foi reprovado;
- O aluno é considerado aprovado se obtiver média igual ou superior a 60 pontos;
- O aluno deve ir para a prova final caso a sua média seja maior ou igual a 30;
- O aluno está reprovado se a sua média for menor que 30;
- A fórmula para o cálculo da média está descrita abaixo. A variável n1 representa a nota do primeiro bimestre e n2 a nota do segundo bimestre.

$$Média = \frac{2n_1+3n_2}{5}$$

4. Faça um programa que calcule o que deve ser pago por um produto, considerando o preço normal de venda e a escolha da condição de pa-

gamento. Utilize os códigos da tabela abaixo para ler qual a condição de pagamento escolhida (1, 2 ou 3) e efetuar o cálculo do valor do preço final do produto.

5.

Código	Condição de Pagamento
1	À vista em dinheiro ou cheque, recebe 10% de desconto.
2	À vista no cartão de crédito, 5% de desconto.
3	Em 2 vezes, preço normal de venda, sem juros.

6. Determinar o tipo de um triângulo dados os valores dos seus três lados. Você deve inserir testes de acordo com as expressões abaixo para verificar se os valores fornecidos para os lados formam um triângulo e, caso formem, escrever qual o tipo do triângulo. Nas expressões abaixo, "a", "b" e "c" representam os lados do triângulo.

- **Triângulo**: Figura geométrica de três lados, em que cada lado é menor que a soma dos outros dois.

  (a < b + c) **and** (b < a + c) **and** (c < a + b)

- **Triângulo equilátero**: Os três lados são iguais.

  (a == b) **and** (b == c)

- **Triângulo isósceles**: Apenas dois lados são iguais.

  (a == b) or (a == c) or (b == c)

- **Triângulo escaleno**: Todos os lados são diferentes.

  (a != b) **and** (b != c) **and** (c != a)

CONHECENDO AS REFERÊNCIAS

D! TUTORIAIS. **Aprendendo Python 4**: Estruturas de Decisão. Disponível em: <https://dtutoriais.wordpress.com/2010/07/20/aprendendo-python-4--estruturas-de-decisao/>. Acesso em: 12 mar. 2015

MASANORI, F. **Python para Zumbis**. Disponível em: <http://pycursos.com/python-para-zumbis/>. Acesso em: 10 mar. 2015.

MELANDA, J. C. M. **Blocos de Código em Python**. Disponível em: <http://programeempython.blog.br/blog/blocos-de-codigo-em-python/>. Acesso em: 12 mar. 2015.

PYTHON SOFTWARE FOUNDATION. Disponível em: <https://www.python.org/>. Acesso em: 10 mar. 2015.

SEVERANCE, C. **Python for Informatics**: Exploring Information. 2013. Disponível em: <http://www.pythonlearn.com/book.php>. Acesso em: 12 mar. 2015.