Apostila de Introdução à Programação
Prof. Eduardo Corrêa

Capítulo V –Estruturas de Seleção: if, else, elif

Sumário

V. Estruturas de Seleção: if, else, elif 2
V.1 A Estrutura de Seleção 2
V.2 Blocos de Código 3
V.3 Operadores Relacionais 4
V.4 Operadores Lógicos 5
V.5 A estrutura de seleção if-else 9
V.6 Problemas Resolvidos 12
V.7 A instrução elif 15
V.8 A sintaxe econômica 18
V.9 A operação ternária 19
Exercícios Propostos 20

V. Estruturas de Seleção: if, else, elif

Este capítulo aborda a utilização dos comandos **if, else** e **elif**, as estruturas de seleção da Linguagem Python.

V.1 A Estrutura de Seleção

Até o momento todos os nossos programas seguiram um mesmo padrão: (i) receber dados como entrada, (ii) processar estes dados e (iii) mostrar os resultados do processamento na tela. Os programas começam na instrução especificada na primeira linha do programa e **executam cada instrução ordenadamente**, encerrando ao chegar na última instrução. Esta sequência objetiva é apresentada na Figura 1: um programa que recebe a matrícula e duas notas de um aluno como entrada e calcula e exibe a média destas notas.

```
#Programa Bobo
print('CONSULTA MEDIA')
matricula = input('Digite sua matricula: ')
nota1 = float(input('Digite a primeira nota: '))
nota2 = float(input('Digite a segunda nota: '))
media = (nota1 + nota2) / 2;
print('MEDIA FINAL = ', round(media,2))
```

Figura 1. Programa que contém apenas comandos simples

Suponha, entretanto, que seja necessário alterar o programa acima da seguinte forma: ele deve "descobrir" se a média do aluno é suficiente para a sua aprovação e informar esta situação na tela. Neste caso, será preciso utilizar as **instruções de seleção** do Python: **if** e **else**, conforme apresentado na Figura 2.

```
#Programa Esperto
print('CONSULTA MEDIA')
matricula = input('Digite sua matricula: ')
nota1 = float(input('Digite a primeira nota: '))
nota2 = float(input('Digite a segunda nota: '))
media = (nota1 + nota2) / 2;
print('MEDIA FINAL = ', round(media,2))
if (media >= 7.0):
    print('APROVADO')
    print('PARABÉNS!!!')
else:
    print('NÃO FOI APROVADO')
```

Figura 2– Programa que contém estrutura de seleção

As estruturas de seleção são usadas sempre que o **programa precisar agir de forma diferente em situações diferentes**. No programa acima, a instrução **if** verifica se o aluno foi aprovado, comparando sua média com o valor 7.0. Se (*if*) a média for maior ou igual a 7.0,

então as mensagens 'APROVADO' e 'PARABÉNS!!!' são exibidas na tela. Senão (*else*), a mensagem 'NÃO FOI APROVADO' é exibida na tela.

Observe que os comandos `if` e `else` terminam com um sinal de dois pontos ":" e que as **linhas subordinadas** a cada comando estão **indentadas**. Neste exemplo, o `if` tem dois comandos subordinados e o `else` apenas um. No jargão da computação, um conjunto de comandos subordinados é chamado de **bloco de código**. A seção a seguir descreve em detalhes o esquema utilizado pelo Python para a definição de blocos de código.

V.2 Blocos de Código

Para começar, um aviso: preste atenção, muita atenção mesmo, pois será apresentada uma característica muito peculiar do Python. Ao contrário do que ocorre em qualquer outra linguagem de programação, os blocos de código que estão subordinados ao `if`, `else` e a outros comandos que ainda serão apresentados, como `for` e `while`, são definidos através da **indentação de comandos**[1]. Enquanto o Pascal usa as palavras begin e end para marcar um bloco e linguagens como C, Java e R utilizam os símbolos "{" e "}", o Python utiliza espaços em branco ou tabulações. É isso mesmo: espaços em branco ou tabulações. Isto significa que os comandos precisam estar alinhados da mesma forma para que o Python os reconheça como parte de um bloco. A convenção é utilizar 4 espaços para indentação. Entretanto, a linguagem permite qualquer padrão de alinhamento (por exemplo, um tab, três espaços, etc.), contanto que este seja repetido para todos os comandos pertencentes a um mesmo bloco. Para que o conceito fique claro, observe o exemplo a seguir. Veja que se você não indentar os comandos, o interpretador Python irá emitir uma mensagem acusando erro de indentação.

Erro de indentação:	**Indentação correta:**

`if (x > 0):`
`a=1`
 	
`IndentationError: expected an indented block`
	`if (x > 0):`
	`    a=1`

⃠ CUIDADO COM O COPIAR E COLAR

Se você tentar selecionar e copiar o código dos programas a partir desta apostila, que está no formato PDF, perceberá que eles perderão a indentação ao serem colados em seu destino. Dessa forma, o mais recomendado é sempre **digitar os exemplos**, até mesmo porque isso irá lhe ajudar a memorizar os comandos e a forma de utilizá-los.

[1] A palavra "indentação" é muito feia, mas infelizmente existe!

V.3 Operadores Relacionais

A instrução **if** toma uma decisão a partir da avaliação de uma **condição** (ou **expressão relacional**). Uma condição representa uma comparação ou um conjunto de comparações que irá resultar sempre em VERDADEIRO (**True**) ou FALSO (**False**). No Python, os operadores utilizados nas condições – denominados **operadores relacionais** – são os seguintes:

==	Igual (*veja que são dois sinais de igualdade, lado a lado*)
!=	Diferente
>	Maior
<	Menor
>=	Maior ou Igual
<=	Menor ou Igual

Os operadores relacionais permitem com que seja realizada uma comparação entre dois valores do mesmo tipo. Estes valores poderão ser variáveis, constantes, expressões aritméticas ou expressões literais. Observe os seguintes exemplos:

- 4 * 2 == 8 resulta em True.
- 9 ** 2 == 81 resulta em True.
- 5 + 2 < 7 resulta em False.
- 5 + 2 <= 7 resulta em True.
- 9 * 9 != 81 resulta em False.
- 7 % 2 != 0 resulta em True.

Conforme foi dito, é possível comparar valores armazenados em variáveis:

- media >= 7.0 lê-se: o conteúdo da variável “media” é maior ou igual a 7.0?
- A > B lê-se: o conteúdo da variável A é maior do que o conteúdo da variável B?
- pais == 'BR' lê-se: o conteúdo da variável “pais” é igual a ‘BR’ (neste caso, é claro que “pais” é uma variável string!)

O resultado de um teste pode ser atribuído a uma variável LÓGICA (isto é, uma variável do tipo **bool**), conforme ilustra o programa a seguir.

```
x = 0; y=1
v1 = (x < y)        #v1 recebe True
v2 = (y % 2 == x)   #v2 recebe False
v3 = (y // 2 == x) #v3 recebe True
print('v1:',v1 )
print('v2:',v2 )
print('v3:',v3 )
```

Saída:

```
v1: True
v2: False
v3: True
```

V.4 Operadores Lógicos

Os **operadores lógicos and** (e), **or** (ou) e **not** (não) podem ser utilizados para **unir** diferentes condições, aumentando assim, a complexidade dos testes. Os operadores lógicos funcionam da seguinte maneira:

- **and** a sentença é verdadeira se TODAS as condições forem verdadeiras.
- **or** a sentença é verdadeira se UMA das condições for verdadeira.
- **not** inverte o valor lógico de uma sentença (True -> False, False-True)

A tabela abaixo, denominada **tabela verdade**, apresenta todas as combinações possíveis entre os valores de variáveis lógicas:

Tabela verdade

Operador and	Operador or	Operador not
True **and** True = True	True **or** True = True	**not** True = False
True **and** False = False	True **or** False = True	**not** False = True
False **and** True = False	False **or** True = True	
False **and** False = False	False **or** False = False	

Analise os seguintes exemplos. Para facilitar a visualização, as condições foram colocadas entre parênteses:

- (4 * 2 == 8) **and** (9 ** 2 == 81) resulta em True.
- (1 + 1 == 2) **and** (9 ** 2 > 100) resulta em False.
- (1 + 1 == 2) **or** (9 ** 2 > 100) resulta em True.

Veja um programa com exemplos de expressões que compraram variáveis.

```
a = 10; b=5
v1 = (b < a) and (b > 0)    #v1 recebe True
v2 = (b > a) or (b < 0)     #v2 recebe False
print('v1:',v1 ); print('v2:',v2 )
```

Saída:

```
v1: True
v2: False
```

A seguir, apresenta-se um exemplo do uso do operador **not**. Esse operador simplesmente inverte o resultado de uma condição. Por exemplo, not (1 > 2) resulta em True, pois 1 > 2 é False.

```
a = 10; b=5
v1 = not (b < a) and (b > 0)        #v1 recebe False
v2 = not (b > a) or not (b < 0)     #v2 recebe True
print('v1:',v1 ); print('v2:',v2 )
```

Saída:

```
v1: False
v2: True
```

Para resolver expressões booleanas muito grandes, o Python adota o seguinte critério de prioridades:

1. Efetuar operações embutidas entre parênteses “mais internos”.
2. Efetuar funções (*qualquer função, como por exemplo alguma do módulo math*).
3. Resolver **
4. Resolver *, /, //, %
5. Resolver +, -
6. Resolver os operadores relacionais: ==, !=, >, <, >=, <=.
7. Resolver o operador lógico not
8. Resolver o operador lógico and
9. Resolver o operador lógico or

O Python resolve primeiro o que estiver dentro dos parênteses mais internos e em seguida as funções. A partir daí, observe que os operadores aritméticos têm precedência sobre os lógicos. O Python irá sempre avaliar os operadores aritméticos primeiro (começando do **, seguido dos relacionados a multiplicação e divisão e por último adição e subtração). Em seguida vêm os operadores relacionais. E finalmente, por último os operadores lógicos not, and e or, nesta ordem de precedência.

Isto significa que a expressão `x * 3 >= 100 and y - 2 <= 10` será avaliada de modo que primeiro serão realizadas as operações aritméticas e só depois serão resolvidos os operadores relacionais. Muitos programadores gostam de usar parênteses entre as expressões relacionais: `(x * 3 >= 100) and (y - 2 <= 10)`. Isto não é necessário, mas por outro lado não causa nenhum prejuízo e pode tornar mais fácil para as pessoas lerem e entenderem o código.

Para melhor compreender a prioridade de operações, observe a forma como o Python resolve a expressão apresentada no exemplo a seguir.

EXEMPLO V.1 Resolução de uma Expressão Booleana Complexa

Qual será o valor de v2 ao final da execução deste programa?

```
x = -3; y=9; v1=False;
v2 = (x ** 2 == y) or (not v1) and (x > y // (3 - 1) * 3)
print(v2)
```

RESPOSTA:

A expressão a ser resolvida é:

```
v2 = (x**2 == y) or (not v1) and (x > y // (3 - 1) * 3)
```

Seguindo a tabela de prioridades, o Python resolverá primeiro o que está entre os parênteses mais internos.

```
v2 = (x ** 2 == y) or (not v1) and (x > y // (3 - 1) * 3)

v2 = (x ** 2 == y) or (not v1) and (x > y // 2 * 3)
```

Agora existem três expressões entre parênteses. Então, inicialmente será resolvida a que está mais à esquerda. Primeiro ele determinará o valor do quadrado de x e depois irá comparar com o valor de y (a prioridade da execução de operações aritméticas é maior do que a prioridade da execução das operações relacionais).

```
v2 = (x ** 2 == y) or (not v1) and (x > y // 2 * 3)

v2 = (-3 ** 2 == 9) or (not v1) and (x > y // 2 * 3)

v2 = (9 == 9) or (not v1) and (x > y // 2 * 3)

v2 = True or (not v1) and (x > y // 2 * 3)
```

Agora é a vez de resolver o que está no segundo parênteses. Como o valor de v1 é False, not v1 resultará em True.

```
v2 =True or (not v1) and (x > y // 2 * 3)

v2 =True or (not False) and (x > y // 2 * 3)

v2 =True or True and (x > y // 2 * 3)
```

Finalmente, resolve-se o que está no terceiro parênteses, sempre respeitando a ordem indicada na tabela de prioridades e resolvendo da esquerda para direita quando houver empate na prioridade (como é o caso dos operadores // e *)

```
v2 = True or True and (x > y // 2 * 3)

v2 = True or True and (-3 > 9 // 2 * 3)

v2 = True or True and (-3 > 4 * 3)

v2 = True or True and (-3 > 12)

v2 = True or True and False
```

Agora existem apenas operadores booleanos a serem resolvidos. O Python começa resolvendo o and para depois resolver o or (o operador and tem prioridade maior do que o operador or).

```
v2 = True or True and False

v2 = True or False

v2 = True
```

RESPOSTA: o valor resultante de v2 será True.

Conforme mencionado anteriormente, muitas vezes, mesmo quando não é estritamente necessário, usamos **parênteses** para deixar o código mais legível. Veja o exemplo a seguir. O código funcionaria corretamente sem a necessidade de empregar os

parênteses, mas eles foram utilizados apenas para deixar mais claros os critérios adotados para aprovar ou reprovar um aluno.

EXEMPLO V.2 Considere um programa que possui 3 variáveis:

media: variável real – indica a média final do aluno

faltas: variável inteira – indica o total de faltas do aluno

resultado: variável lógica – indica se aluno foi ou não aprovado.

Qual a expressão que deve ser montada para que "resultado" indique a aprovação do aluno, caso a universidade adote a seguinte regra: "*o aluno é aprovado se tem média maior ou igual a 7 e total de faltas menor do que 10; ou se possui media maior do que 8 independentemente do total de faltas)"*

RESPOSTA:

```
resultado = (media > 8) or (media >= 7 and faltas < 10)
```

V.5 A estrutura de seleção if-else

Com base nas informações que recebe, um programa pode optar por diferentes cursos de execução. Para tal, basta fazer uso da **estrutura de seleção if-else**. Ela permite com que o programa execute **desvios** em diferentes direções, de acordo com o resultado da avaliação de uma condição (por esta razão **if** e **else** são também chamadas de instruções de **desvio condicional**). As subseções a seguir demonstram diferentes formas de utilização da estrutura.

V.5.1 CATEGORIA 1: DESVIO COM "if"

Esta é a forma mais simples de utilizar o desvio condicional. Ocorre quando um programa executa um bloco de código se uma condição for verdadeira. Observe o exemplo da Figura 3.

```
print('Digite a sua Idade: ')
i = int(input())
if (i >= 18):
    print(' Você não é menor de idade')
print('FIM!')
```

Figura 3. Desvio com "if" e uma instrução subordinada

O programa pede ao usuário que digite a sua idade (armazenando na variável "i") e, em seguida, a instrução **if** verifica se o conteúdo de "i" é maior ou igual a 18. Se o resultado do teste for verdadeiro, o programa executará a instrução:

```
print('Voce não é menor de idade')
```

Agora muita ATENÇÃO - Observe que o comando `if` termina com um sinal de dois pontos ":" e que a **linha** a ele **subordinada** está **indentada** (começa com 4 espaços em branco). Observe ainda que o programa executará a instrução `print('FIM!')`, independente do resultado da instrução **if**. Isso porque ela **não está subordinada** ao **if** (está no mesmo nível de indentação do if).

Recomendamos que você teste o programa digitando diferentes valores como entrada para que você entenda bem o funcionamento do programa.

Vamos avançar um pouco mais. E se for preciso executar condicionalmente mais de uma instrução em função do resultado de um teste **if**? O que devemos fazer? Neste caso, será necessário indentar todas essas instruções para indicar que elas são instruções subordinadas ao **if**. Veja o exemplo da Figura 4.

```
print('Digite a sua Idade: ')
i = int(input())
if (i >= 18):
    print('Você não é menor de idade')
    print('Você já pode dirigir!')
    print('Quer comprar um carro? (S=SIM, N=NÃO)')
    resposta = input()
print('FIM!')
```

Figura 4. Desvio com "if" e várias instruções subordinadas.

Caso a idade do usuário seja maior ou igual a 18, as instruções e **end** solicitam ao computador para executar os comandos:

```
print('Você não é menor de idade')
print('Você já pode dirigir!')
print('Quer comprar um carro? (S=SIM, N=NÃO)')
resposta = input()
```

V.5.2 CATEGORIA 2: DESVIO COM "if-else"

A instrução **if** oferece apenas um desvio, que é executado quando a expressão booleana é verdadeira. Porém, em situações práticas é muitas vezes necessário realizar dois desvios para a resolução de um problema: um que seja executado se o resultado for verdadeiro, e outro se for falso. Para expressar essa situação no Python, você deve usar a estrutura de controle **if-else**. Ela funciona assim:

- Se a avaliação for **verdadeira**, será executado o bloco de código que acompanha a instrução **if**.

- Se a avaliação for **falsa**, será executado o bloco de código que acompanha a instrução **else**.

A Figura 5 apresenta o exemplo de desvio com **if-else**.

```
print('Digite a sua Idade: ')
i = int(input())
if (i >= 18):
    print('Você não é menor de idade')
else:
    print('Você é menor de idade')

print('viu como eu conheço a lei?')
```

Figura 5. Desvio com if-else, cada qual com uma única instrução subordinada

Veja que, em qualquer um dos casos, após a execução do bloco de código selecionado, o controle do programa é passado para o comando **print('viu como eu conheço a lei?')**, que não é subordinado do `if` e nem do `else`. Observe ainda que, do mesmo jeito que ocorre com o `if`, o comando `else` termina com um sinal de dois pontos ":" e que a **linha** a ele **subordinada** está **indentada** (começa com 4 espaços em branco).

Finalizando esta seção, a Figura 6 ilustra um programa com mais de uma instrução subordinada ao **`if`** e também mais de uma subordinada ao **else**. Nenhuma surpresa, basta indentar todas as instruções que você deseja que sejam subordinadas ao `if` ou `else`.

```
print('Digite a sua Idade: ')
i = int(input())
if (i >= 18):
    print('Você não é menor de idade')
    print('Você pode dirigir!')
else:
    print('Você é menor de idade')
    print('Você não pode dirigir')

print('viu como eu conheço a lei?')
```

Figura 6. Desvio com if-else, ambos com mais de uma instrução subordinada

V.6 Problemas Resolvidos

Esta seção apresenta uma série de exercícios resolvidos que exemplificam a utilização dos comandos de desvio condicional em diversas situações práticas. Digite os exemplos, execute e faça testes utilizando diferentes entradas.

EXEMPLO V.3 Escreva um programa que leia um número inteiro e determine se ele é par ou ímpar

RESOLUÇÃO:

```
print('Digite um numero inteiro: ')
x = int(input())
if x % 2 == 0:
    print(x, 'é um numero PAR ')
else:
    print(x, 'é um numero IMPAR ')
```

EXEMPLO V.4 Escreva um programa que leia dois números inteiros e determine qual deles é o maior ou se eles são iguais.

RESOLUÇÃO:

```
print('* * O MAIOR DENTRE DOIS NUMEROS * *')
print();

#solicita x e y
x = int(input('Digite um número inteiro: '))
y = int(input('Digite outro número inteiro: '))
print();

#identifica quem é o maior ou se eles são iguais
if x > y:
    print('o primeiro é maior ')
else:
    if x < y:
        print('o segundo é maior ')
    else:
        print('são iguais ')
```

O programa acima representa um exemplo de uso de comandos if "**aninhados**" ou "**encaixados**". Isto significa a tomada de uma decisão dentro de outra decisão. No programa há um primeiro teste para determinar se "x" é maior do que "y". Caso o resultado do teste seja verdadeiro, a mensagem 'o primeiro é maior' é impressa na tela. Senão, é preciso fazer outro teste (ou seja, um teste é feito dentro de outro teste!): se "x" for menor do que "y", exibe-se 'o segundo é maior', senão exibe-se 'são iguais'.

Para implementar uma rotina deste tipo – `if` dentro de `else`, `if` dentro de `if`, ou qualquer coisa parecida – também é preciso fazer uso da indentação.

EXEMPLO V.5. Escrever um programa que funcione da seguinte maneira:

Inicialmente, o programa deve **ler** um **número inteiro** digitado pelo usuário e armazená-lo numa variável chamada x.

Em seguida, o sistema deve **calcular** e **imprimir** o valor da função **y = f(x) + g(x)**, onde:

$h(x) = x^2 - 16$

$f(x) = \begin{cases} h(x), \text{ se } h(x) \geq 0 \\ 1, \text{ se } h(x) < 0 \end{cases}$

$g(x) = \begin{cases} x^2 + 16, \text{ se } f(x) = 0 \\ 0, \text{ se } f(x) > 0 \end{cases}$

RESOLUÇÃO:

```
x = int(input('digite o valor de x (número inteiro): '))

#calcula h(x)
hx = x ** 2 - 16

#calcula f(x)
if hx >= 0:
    fx = hx
else:
    fx = 1

#calcula g(x)
if fx == 0:
    gx = x ** 2 + 16
else:
    gx = 0

#agora sim, posso calcular e imprimir y
y = fx + gx
print('y = ', y)
```

EXEMPLO V.6. Numa fábrica, trabalham operários divididos em duas classes:

A – Os que fazem até 50 peças por mês. **B – Os que fazem mais de 50 peças por mês.**

Os operários da classe A recebem R$ 1500 por mês.

Já os empregados da classe B recebem R$ 1500,00 mensais mais R$ 25 por peça adicionalmente produzida (ou seja, peça além das 50 iniciais).

Escreva um programa que leia a classe do funcionário e o número de peças produzidas por ele no mês e calcule o seu salário.

```
#PASSO 1: RECEBE A CLASSE DO FUNCIONÁRIO
print('Digite a classe do funcionário:')
classe = input()

#PASSO 2: DETERMINA E IMPRIME O SALÁRIO DO FUNCIONÁRIO
if (classe == 'A') or (classe == 'a'):
    print('O salário do funcionário é R$ 1500.00')
else:
    if (classe == 'B') or (classe == 'b'):
        print('Entre com o número de peças fabricadas: ')
        pecas = int(input())
        if (pecas > 50):
            salario = 1500 + ((pecas - 50) * 25)
        else:
            salario = 1500
        print('O salário do funcionário eh R$ ', round(salario,2))
    else:
        print('VOCÊ DIGITOU UMA CLASSE DE FUNCIONARIO INEXISTENTE! ')
```

Observações:

1. É preciso testar: `if (classe == 'A') or (classe == 'a')`, ou seja, o valor da variável "classe" em maiúsculo e minúsculo (as letras maiúsculas e minúsculas possuem códigos internos diferentes).

2. Veja que o programa é inteligente o suficiente para não fazer o cálculo quando o usuário entra com uma classe inexistente (uma classe que não seja do tipo A ou B).

EXEMPLO V.7. Escreva um programa que leia o número de um mês (ex.: 1 = Janeiro... 2 = Fevereiro) e o número de um ano (ex: 1970, 2002, 1958, etc) e imprima a quantidade de dias do mês.

OBS: é preciso considerar que o número de dias do mês de fevereiro é igual a 29 em anos bissextos. Para ser bissexto, o ano deve ser:

- **múltiplo de 4, mas não pode ser múltiplo 100;**
- **ou é múltiplo de 400.**

A resolução é apresentada na página a seguir. Analise com calma e teste o programa com diferentes entradas para facilitar o seu entendimento.

```
#PASSO 1: recebe o ano e mês
print('Digite o ano: ')
ano = int(input())

print('Digite o número do mês (1 a 12): ')
mes = int(input())

#PASSO 2: calcula e exibe a quantidade de dias
if mes > 12 or mes < 1:
    print('DADOS INVÁLIDOS!')
else:
    if (mes == 4) or (mes == 6) or (mes == 9) or (mes == 11):
        print('O MÊS TEM 30 DIAS')
    else:
        if (mes != 2):
            print('O MÊS TEM 31 DIAS')
        else: #se for fevereiro, verifica se é ano bissexto
            if (ano % 4 == 0 and ano % 100 != 0) or (ano % 400 == 0):
                print('O MÊS TEM 29 DIAS')
            else:
                print('O MÊS TEM 28 DIAS')
```

EXEMPLO V.8. Escreva um programa que receba como entrada dois números reais e os imprima em ordem crescente.

```
n1 = float(input('Digite um número real: '))
n2 = float(input('Digite outro número real: '))

if (n1 < n2):
    print(n1, n2)
else:
    print(n2, n1)
```

V.7 A instrução elif

Em muitas situações práticas, existe a necessidade de avaliar mais de duas possibilidades em um teste condicional. Neste caso, utilizando **apenas if** e **else**, o programa acaba ficando um excessivamente indentado, como acontece no exemplo abaixo, onde 4 diferentes condições precisam ser avaliadas.

EXEMPLO V. 9. Escreva um programa que receba como entrada o valor de uma temperatura em graus Celsius e defina e imprima o valor de uma variável chamada "classe" da seguinte forma:

- **Se a temperatura digitada for menor ou igual a 20ºC, o valor de classe deve ser 'FRIO'**
- **Se for maior que 20ºC até 29º, o valor deve ser 'AGRADÁVEL'**
- **30ºC a 34º, o valor será 'CALOR'**
- **Acima de 34ºC, o valor será 'CALOR ABSURDO'**

SOLUÇÃO 1 (*com indentação "chata"*)

```
#recebe o valor da temperatura
temperatura = float(input('Qual a temperatura hoje? '))

#determina a classe da temperatura usando apenas if e else
if (temperatura <= 20):
    classe_temperatura = 'FRIO'
else:
    if (temperatura > 20) and (temperatura < 30):
        classe_temperatura = 'AGRADÁVEL'
    else:
        if (temperatura >= 30) and (temperatura <= 34):
            classe_temperatura = 'CALOR'
        else:
            classe_temperatura = 'CALOR ABSURDO';

print('Eu acho que isso é', classe_temperatura)
```

A solução acima está correta, porém a indentação – espaços em branco para alinhar os comandos subordinados aos if's, else's– acaba causando um efeito visual um pouco desagradável no código. Veja que o programa fica excessivamente "gordinho" (ocupando muitas colunas), por isso, mais difícil de ser examinado. O mesmo tipo de situação acontece no Exemplo V.7 (volte nesse exemplo e veja).

Para contornar o problema, a maioria dos programadores a linguagem Python oferece um comando chamado **elif** (veja que o nome faz uma junção das palavras else e if). A instrução `elif` é usada para **checar mais de uma única condição**, parando o teste e executando os comandos subordinados à primeira condição avaliada como True. Veja abaixo o exemplo que determina a classe de temperatura, agora usando **elif**. Após o código, apresentamos uma explicação mais detalhada.

SOLUÇÃO 2 (*com elif para evitar a indentação "chata"*)

```
#recebe o valor da temperatura
temperatura = float(input('Qual a temperatura hoje? '))

#determina a classe da temperatura usando if-else-elif
if (temperatura <= 20):
    classe_temperatura = 'FRIO'
elif (temperatura > 20) and (temperatura < 30):
    classe_temperatura = 'AGRADÁVEL'
elif (temperatura >= 30) and (temperatura <= 34):
    classe_temperatura = 'CALOR'
else:
    classe_temperatura = 'CALOR ABSURDO';

print('Eu acho que isso é', classe_temperatura)
```

Explicação:

- Antes de mais nada, é importante entender que os comandos `if`, `elif` e `else` **trabalham em conjunto**.

- Quando utilizados, a primeira coisa que o computador vai checar, é se a condição associada ao comando `if` é verdadeira (resulta em True). Neste caso, os comandos (ou comando) subordinados ao `if` serão executados e nenhum dos `elifs` será processado.

- Porém, caso o teste do `if` resulte em False, o computador checará o **primeiro `elif`** da sequência. Se a condição deste `elif` resultar em True, os comandos (ou comando) a ele associados serão executados e nenhum dos `elifs` restantes será avaliado.

- Entretanto, se o primeiro `elif` tem um teste que resulta em False, o computador analisa o segundo `elif`. Se o teste do segundo `elif` resultar em True, os seus comandos subordinados são executados e os `elifs` restantes não serão avaliados.

- E assim ocorrerá para todos os comandos `elif` posteriores: o computador vai processando a sequência de cima pra baixo e quando ele acha algum `elif` que resulta em True, ele executa os seus comandos e não processa os `elifs` seguintes.

- Se nem o `if` e nem qualquer dos `elifs` resultarem em True, os comandos subordinados ao `else` serão executados. Ou seja: o código que for colocado no `else` será executado apenas se todos os testes anteriores resutarem em False. Por isso, o `else` precisa ser o último comando da sequência.

    - É importante esclarecer que quando você usa `if` ou `elif`, você **não** é obrigatório incluir o `else`. Porém, em grande parte das situações práticas, o uso do `else` acaba sendo necessário para que seja obtida a solução correta do problema. Você não é obrigado a usar o `else`, mas se fizer isso, ele tem que ser colocado no final da sua "bateria de testes".

- Por fim, agora falando sobre a sintaxe, veja que ela é parecida com a do `if` e `else`. Isto é: o comando `elif` deve terminar com um sinal de dois pontos ":" e as linhas a ele subordinadas precisam ser indentadas

Para não ficarmos em um único exemplo, veja agora o Exemplo V.7 agora resolvido com o uso do `elif`.

```
#PASSO 1: recebe o ano e mês
print('Digite o ano: ')
ano = int(input())

print('Digite o número do mês (1 a 12): ')
mes = int(input())

#PASSO 2: calcula e exibe a quantidade de dias
if mes > 12 or mes < 1:
    print('DADOS INVÁLIDOS!')
elif (mes == 4) or (mes == 6) or (mes == 9) or (mes == 11):
    print('O MÊS TEM 30 DIAS')
elif (mes != 2):
    print('O MÊS TEM 31 DIAS')
else: #se for fevereiro, verifica se é ano bissexto
    if (ano % 4 == 0 and ano % 100 != 0) or (ano % 400 == 0):
        print('O MÊS TEM 29 DIAS')
    else:
        print('O MÊS TEM 28 DIAS')
```

V.8 A sintaxe econômica

Como você já sabe, as variáveis lógicas (bool) podem armazenar apenas dois valores: True ou False. Agora chegou a hora de você aprender que estas variáveis podem ser testadas de duas formas distintas em instruções `if` e `elif`. São as formas "convencional" e "econômica", conforme mostra o exemplo abaixo.

```
x = True
y = False

#testa se é True do jeito convencional
if (x == True):
    print('x é True')

#teste do jeito econômico... não preciso especificar o "= True"
if x:
    print('x é True (teste econômico)')

#testa se é False do jeito convencional
if (y == False):
    print('y é False')

#teste do jeito econômico... usa-se "not variável"
if not y:
    print('y é False (teste econômico)')
```

```
>>>
x é True
x é True (teste econômico)
y é False
y é False (teste econômico)
```

Para testar se o valor de uma variável v do tipo bool é True, basta você utilizar **if v** – você não precisa fazer if (v==True). Se você preferir, pode também usar parênteses e escrever **if (v)**.

De maneira análoga, se você quiser testar se o valor de v é False, pode usar a sintaxe **if not v** em vez de fazer if (v==False). Se, por uma questão estética, preferir incluir parênteses, tudo ok: **if not (v)** também funciona corretamente.

A forma econômica costuma ser mais usada pelos programadores.

V.9 A operação ternária

A operação ternária é um interessante (e muito utilizado) recurso que podemos empregar para realizar uma **operação de atribuição** (definir o valor de uma variável) a partir do resultado de um teste lógico. Esse teste lógico deve ser definido em **uma única linha**, possuindo a seguinte sintaxe:

```
v = valor1 if condição else valor2
```

A sintaxe da operação ternária parece estranha, mas o funcionamento é simples. Veja só: neste exemplo a variável "v" vai receber "valor1" caso a condição associada ao if resulte no valor True. Caso contrário, ela receberá o "valor2". Veja a seguir alguns exemplos práticos:

```
a = 10
b = 20

#usa o operador ternário para atribuir o valor às variáveis menor e maior
menor = a if a < b else b
maior = a if a > b else b

print(menor); print(maior)
```

```
>>>
10
20
```

No exemplo acima, a variável "menor" receberá o valor armazenado na variável "a" porque o resultado do teste `a < b` resulta em True. Já "maior" recebe o valor armazenado em "b" pelo fato de o teste `a > b` resultar em False.

É importante registrar que a operação ternária não precisa necessariamente utilizada em uma atribuição. Ela é uma operação que retorna um valor, mas armazenar o valor em uma variável fica a critério do programador. Veja o exemplo abaixo, onde passamos a utilizar a operação dentro do comando print().

```
a = 10; b = 20

print("menor valor: ", a if a < b else b)
print("maior valor: ", a if a > b else b)
```

```
>>>
10
20
```

Exercícios Propostos

(1) Escreva um programa que receba o valor de um número inteiro positivo *x* via teclado e calcule e imprima o valor da seguinte função: $f(x) = (1 \div x) + (1 \div x^2)$

***Obs.**: Se o usuário digitar **0** para x o programa não poderá aceitar esta entrada (deverá informar ao usuário que é impossível calcular f(x)).*

(2) Faça um programa que solicite a entrada de um número inteiro e que imprima se ele é divisível por 3 e 5.

(3) Faça um programa que solicite a entrada de três números inteiros e que imprima na tela qual é o maior deles.

(4) Crie um programa que leia um número inteiro e imprima se ele está no intervalo entre 0 e 100, se está no intervalo entre 101 e 500, ou se está fora das duas faixas anteriores.

(5) Um professor calcula as medias dos alunos de forma

Media = ((2 x P1) + (3 x P2)) ÷ 5

Faça um programa para calcular a média ponderada do aluno, e exibir as seguintes mensagens:

APROVADO - se a média obtida for maior ou igual a 7
PROVA FINAL - se a média for menor que 7 e maior que 4
REPROVADO - se a média for menor que 4.

(6) Considere a seguinte função:
f(x) = x - 2 {se x > 2}
f(x) = x + 1 {caso contrário}

Escreva um programa que receba o valor de *x* (inteiro) através do teclado e obtenha o valor de f(x).

(7) Faça um programa que leia um valor de *x* (real) e imprima o valor de *y* considerando a seguinte regra:

$y = 1$, se $x \leq 1$
$y = 2$, se $1 < x \leq 2$
$y = x^2$, se $2 < x \leq 3$
$y = x^3$, se $x > 3$

(8) Elabore um programa que seja capaz de calcular a área de um quadrado ou de um retângulo. Ele deve funcionar da seguinte maneira:

- Ler o tipo de figura geométrica ("R" = Retângulo e "Q" = Quadrado).
  - Caso a figura seja um retângulo, o algoritmo deve ler os valores da altura e da largura.
  - Caso a figura seja um quadrado, o algoritmo deve ler o valor do lado.
- Após receber os dados de entrada, o programa deve calcular e imprimir o valor da área da figura geométrica.

(9) Faça um programa que leia o raio "r" de uma circunferência com centro na origem (0,0), e em seguida as coordenadas de um ponto "p", e verifique se esse ponto é **interno** ou **pertence** à circunferência.

(10) Um comerciante utiliza o seguinte critério para definir o preço dos produtos de sua loja: ele vende um produto com lucro de 45% se o tiver comprado por valor igual ou inferior a R$ 20,00; caso contrário, vende com lucro de 30%. Faça um programa que receba como entrada o valor da compra do produto e calcule e imprima o valor da venda.

(11) Considere que num determinado país exista a seguinte regra para a obtenção da aposentadoria:

- Um **homem** pode se aposentar caso:
  - Tenha idade igual ou superior a 62 anos ou;
  - Possua 35 ou mais anos de contribuição previdenciária.

- Uma **mulher** pode se aposentar caso:
  - Tenha idade igual ou superior a 60 anos ou;
  - Possua 30 ou mais anos de contribuição previdenciária.

Escreva um programa que realize as seguintes tarefas:

- Leia o nome, sexo, idade e tempo de contribuição previdenciária de uma pessoa.

- Imprima o ano em que essa pessoa poderá se aposentar.

(12) Faça um programa para ler três valores de comprimento. Em seguida, verifique se eles podem formar os lados de um triângulo. Se isto for possível, verifique se o triângulo é equilátero, isósceles ou escaleno. **AJUDA PARA A RESOLUÇÃO DO PROBLEMA:** O comprimento de cada lado de um triângulo é sempre menor que a soma dos outros dois lados.

(13) Faça um programa que solicite a entrada de quatro números inteiros e que imprima na tela qual é o maior deles.

(14) Faça um programa que informe a quantidade total de calorias de uma refeição a partir da escolha do consumidor, que deverá informar o nome do prato principal, da sobremesa e da bebida (todas como string). A seguir são relacionadas as opções vendidas:

Prato Principal:

- Vegetariano 180cal
- Frango 230cal
- Massa 260cal
- Churrasco 300cal

Sobremesa:

- Abacaxi 75cal
- Mousse 200cal

Bebida:

- Água 0cal
- Suco de Laranja 70cal
- Guaraná 180cal

(15) Elabore um programa Python que leia um determinado horário (horas e minutos) no fuso horário de **Brasília** e calcule e imprima o horário correspondente em **Roma**, na Itália.

A diferença entre os horários de Brasília e Roma é de +5 horas (desconsiderando o horário de verão), ou seja, quando são 8hs da manhã em Brasília já são 13:00hs da tarde em Roma.

O programa deve aceitar como entrada, horários no formato HH24, que é indicado na tabela a seguir:

HORÁRIO	VALOR QUE DEVE SER LIDO NO TECLADO
00:00hs (meia-noite)	0
1:00h	1
...	...
12:00hs (meio-dia)	12
13:00hs	13
...	
23:00hs	23

O programa. deve ser suficientemente inteligente para **não calcular e imprimir valores absurdos** para um horário de Roma (ex.: 25h, 30h, etc.).

Caso já seja um outro dia em Roma, o algoritmo deve imprimir ao lado do horário convertido a seguinte mensagem "dia seguinte". Considere os seguintes exemplos:

- Usuário digita o valor 23 (horas) e 25 (minutos) no teclado. O programa deve imprimir como resposta: "4 hora(s) e 25 minuto(s) em Roma (dia seguinte)".

- Usuário digita o valor 9 (horas) e 30 (minutos) no teclado. O programa deve imprimir como resposta: "14 hora(s) e 30 minuto(s) em Roma"