

Introdução

Nesta aula usaremos o comando condicional **if...elif...else** para introduzir conjuntos maiores de decisões em um programa. A tabela abaixo resume a sintaxe desse comando:

Algoritmo	Sintaxe Python	Exemplo
<pre>se condição_1 então: <comando(s) 1> senão se condição_2 então: <comando(s) 2> senão se condição_3 então: <comando(s) 3> ... senão: <outro(s) comando(s)></pre>	<pre>if condição_1: <comando(s) 1> elif condição_2: <comando(s) 2> elif condição_3: <comando(s) 3> ... else: <outro(s) comando(s)></pre>	<pre>if x > 0: x = x + 1 print(x) elif x < 0: y = x - 1 print(y) else: print('x nulo')</pre>

Nesse caso, se a “condição 1” for verdadeira, o bloco **<comando(s) 1>** será executado. Caso contrário, se a “condição 2” for verdadeira, o bloco **<comando(s) 2>** será executado, e assim por diante. Se nenhuma das condições dadas forem verdadeiras, o programa executará o bloco **<outro(s) comando(s)>**. Repare que o termo **elif** é apenas uma contração de “else if”, que significa “senão se”.

Veja o exemplo mais completo a seguir:

```
x = int( input('Entre com um número inteiro qualquer: ') )
if x > 0:
    x = x + 1
    print( x )
elif x < 0:
    y = x - 1
    print( y )
else:
    print('o valor digitado é igual a zero')
```

O que será escrito na tela pelo programa acima se você digitar o valor -5? E se você entrar com 0? Se estiver em dúvida, execute esse programa dentro do IDLE para ver o que acontece.

Agora suponha que você queira testar se um valor inteiro x está em um dos seguintes intervalos:

$x \leq 0$ ou $x > 30$	$1 \leq x \leq 10$	$11 \leq x \leq 20$	$21 \leq x \leq 30$
------------------------	--------------------	---------------------	---------------------

Poderíamos implementar isso assim:

```
if x <= 0 or x > 30:
    print('Valor <= 0 ou > 30')
if 1 <= x <= 10:
    print('Valor entre 1 e 10')
if 11 <= x <= 20:
    print('Valor entre 11 e 20')
if 21 <= x <= 30:
    print('Valor entre 21 e 30')
```

Prática 5 – INF100 – 2018/I – Valor: 1 ponto

Apesar de funcionar bem, repare que, se x for igual a 5, o computador imprimirá a mensagem do primeiro **if** e depois continuará testando as outras três condições desnecessariamente. Sabemos que, uma vez encontrada a condição verdadeira nesse caso, as outras não precisam mais ser testadas.

Poderíamos então implementar isso de forma mais eficiente, assim:

```
if x <= 0 or x > 30:
    print('Valor <= 0 ou > 30')
elif 1 <= x <= 10:
    print('Valor entre 1 e 10')
elif 11 <= x <= 20:
    print('Valor entre 11 e 20')
elif 21 <= x <= 30:
    print('Valor entre 21 e 30')
```

Desse modo, assim que o computador encontrar a primeira condição verdadeira (correspondente ao intervalo desejado), irá imprimir a mensagem correspondente e depois continuar a execução no comando encontrado após esse trecho todo (se houver).



Podemos simplificar ainda mais essas condições. Veja que, se a primeira condição for falsa, então obviamente o valor de x é maior ou igual a 1. Pois se não fosse, a primeira condição seria verdadeira, já que, se x não for maior ou igual a 1, então $x \leq 0$.

Podemos então implementar isso de forma ainda mais simples e eficiente, assim:

```
if x <= 0 or x > 30:
    print('Valor <= 0 ou > 30')
elif x <= 10:
    print('Valor entre 1 e 10')
elif x <= 20:
    print('Valor entre 11 e 20')
else:
    print('Valor entre 21 e 30')
```

Dicas de Indentação no IDLE

Ao digitar os dois-pontos no final da condição do **if** e apertar **Enter**, o IDLE já faz a indentação da linha de baixo automaticamente. Para remover a indentação, basta usar a tecla **Backspace** (seta para a esquerda acima da tecla **Enter**). Seguem mais alguns atalhos:

Efeito	Tecla
Indentar a linha atual	Tab 
Des-indentar a linha atual	Backspace 
Indentar várias linhas (já selecionadas)	Tab ou Ctrl +]
Des-indentar várias linhas (já selecionadas)	Ctrl + [

Roteiro de Prática

Nome do arquivo a ser entregue: **p05.py**

Obs.: Recomenda-se salvar o arquivo com certa frequência para não perder a digitação já feita em caso de uma falha na rede elétrica.

Entre os Cristãos, a Páscoa é a celebração da ressurreição de Jesus Cristo.

O Dia da Páscoa, por definição, é o primeiro Domingo após a primeira lua cheia que ocorre depois do equinócio da Primavera (no hemisfério norte, Outono no hemisfério sul), e pode cair entre 22 de março e 25 de abril. As fórmulas existentes calculam o que se convencionou chamar de "Cálculo Eclesiástico", definido pelo Concílio de Nicéia (325 d.C.).

Existem diversas fórmulas para se determinar o Domingo de Páscoa, entretanto uma das mais simples é a fórmula de Gauss, descrita a seguir.

Para calcular o dia da Páscoa (Domingo), primeiro obtemos o valor do *ANO*, que deve ser lido como um valor inteiro de 4 dígitos. Depois usamos a tabela abaixo para determinar o valor de *X* e *Y*:

ANO	X	Y
1582 a 1699	22	2
1700 a 1799	23	3
1800 a 1899	23	4
1900 a 2099	24	5
2100 a 2199	24	6
2200 a 2299	25	0
2300 a 2399	26	1
2400 a 2499	25	1

Veja que os valores de *X* e *Y* podem ser obtidos usando o comando `if/elif/else`, como foi mostrado na revisão da matéria acima, assim:

```
if ano <= 1699:
    x = 22
    y = 2
elif ano <= 1799:
    x = 23
    y = 3
...
```

} Determinar x e y

Em seguida, usamos as fórmulas abaixo. O operador **MOD** se refere ao resto da divisão inteira, que em Python pode ser obtido com o operador `%`.

```
a = ANO MOD 19
b = ANO MOD 4
c = ANO MOD 7
d = (19 * a + X) MOD 30
e = (2 * b + 4 * c + 6 * d + Y) MOD 7
```

} Calcular a, b, c, d, e

Em seguida:

Calcula-se o valor de *P* dado por $P = (22 + d + e)$.

Se *P* for menor ou igual a 31, a Páscoa será no dia *P* de março.

Caso contrário:

Calcula-se $P' = (d + e - 9)$.

Se *P'* for menor ou igual a 25 a Páscoa será no dia *P'* de abril.

Caso contrário:

Calcula-se $P'' = (P' - 7)$ e a Páscoa será a *P''* de abril.

} Determinar dia e mês da páscoa

Faça um programa que leia um valor para o ano, e diga o dia e mês que ocorreu ou ocorrerá a Páscoa naquele ano. O programa deve verificar se o ano digitado está presente na tabela acima, conforme mostrado nos exemplos adiante. Segue o algoritmo completo do programa:

```
Leia ano
Se ano estiver fora do intervalo 1582 a 2499:
    Escreva a mensagem de erro
Senão:
    Obter x e y
    Calcular a, b, c, d, e
    Determinar e escrever na tela o dia e o mês da páscoa
```

Lembre-se que cada um dos 3 últimos passos desse algoritmo estão apresentados em detalhes na página anterior.

Segue vários exemplos da “tela” de execução desse programa. As entradas de dados do usuário (pelo teclado) estão **destacadas**.

Exemplo 1:

```
Digite um ano (1582 a 2499): 2011
Em 2011 a Páscoa foi ou será em 24 de abril
```

Exemplo 2:

```
Digite um ano (1582 a 2499): 2016
Em 2016 a Páscoa foi ou será em 27 de março
```

Exemplo 3:

```
Digite um ano (1582 a 2499): 1981
Em 1981 a Páscoa foi ou será em 19 de abril
```

Exemplo 4:

```
Digite um ano (1582 a 2499): 1500
1500 está fora do intervalo previsto
```

Exemplo 5:

```
Digite um ano (1582 a 2499): 2500
2500 está fora do intervalo previsto
```

☞ A saída do programa deve obedecer à formatação **exata** mostrada nos exemplos acima.

☞ Não esqueça de preencher o cabeçalho com seus dados e uma breve descrição do programa.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p05.py**) através do sistema do LBI.