Disciplina	Descrição deste documento
Extra Curricular Algoritmos com implementação em Python 3	Parte 2 Condições Simples e Compostas / Comando Condicional

Comando Condicional

É comum que em um algoritmo seja necessária a tomada de decisões baseadas em valores contidos em variáveis. Por exemplo, considere um algoritmo que tenha duas variáveis inteiras A e B previamente carregadas com algum valor. Caso seja necessário calcular a divisão de A por B e o conteúdo da variável B for zero, ocorrerá um erro, como pode ser visto na figura abaixo. Isto ocorre porque divisões por zero não são permitidas.

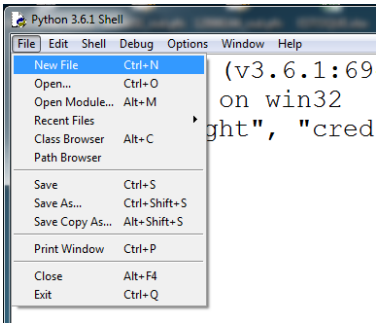
```
>>> A = 16
>>> B = 0
>>> R = A / B
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    R = A / B
ZeroDivisionError: division by zero
```

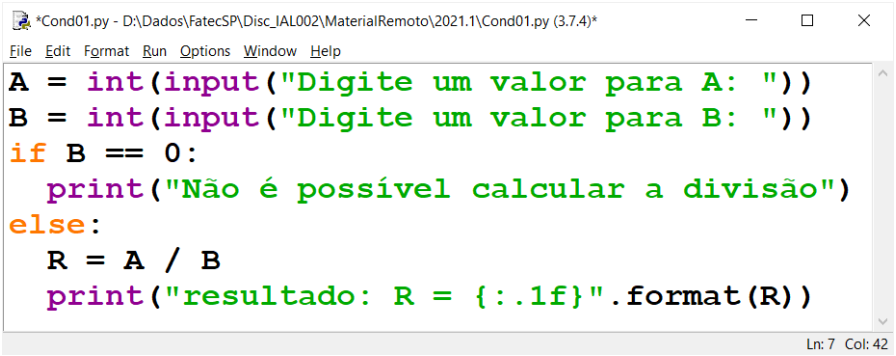
No caso deste algoritmo a situação de erro é indesejável, então é preciso tomar o cuidado de se evitá-la. Uma das formas de se conseguir isso é usar o Comando Condicional: *`if - else`*.

Ao utilizá-lo será necessário formular uma condição cujo resultado será falso ou verdadeiro e em função desse resultado o programa será escrito de modo a executar diferentes comandos em cada caso. Então pode-se formular a seguinte ideia: "se B for igual a zero, então apresente a mensagem *'Não é possível calcular a divisão*', senão (ou seja, B é diferente de zero) calcule e apresente na tela A / B. Agora é preciso escrever isso em Python. Assim, tem-se o exemplo a seguir, onde é feita a leitura das variáveis A e B usando-se o comando *`input`*.

```
A = int(input("Digite um valor para A: "))          ①
B = int(input("Digite um valor para B: "))          ②
if B == 0:                                          ③
  print("Não é possível calcular a divisão")        ④
else:                                               ⑤
  R = A / B                                         ⑥
  print("resultado: R = {:.1f}".format(R))          ⑦
```

Teste esse pequeno programa no Python escrevendo-o na forma de script. Para isso abra o Python e acione o comando do menu: File -> New File e escreva o programa exibido abaixo, salve-o e execute-o.





Caso seja fornecido o valor zero para B, será apresentada a mensagem e caso B seja diferente de zero, então será apresentado o resultado do cálculo. Rode várias vezes esse programa, testando-o com diferentes valores para A e B.

Explicando em detalhes este exemplo

Nas linhas ① e ② é feita a leitura das variáveis A e B de modo que qualquer valor inteiro pode ser inserido para qualquer uma delas.

Na linha ③ há o comando *if* (se) e sua condição. Nesta condição a pergunta que se está fazendo é se o conteúdo de B é igual a zero. Para isso foi usado o operador relacional '==' que avalia se a variável B que está do lado esquerdo contém um valor igual a zero, que está do lado direito. Esta condição será avaliada pelo processador e um resultado será gerado. Esse resultado pode ser falso ou verdadeiro. Caso seja verdadeiro, o programa seguirá para a linha ④ e executará o *print*. Caso seja falso, o programa desviará (pulará) a linha ④ e seguirá para a execução das linhas ⑥ e ⑦, que estão subordinadas ao *else* (senão) da linha ⑤.

Note que nas linhas ③ e ⑤ há um caractere ':' (dois pontos) no final da linha. Em Python é obrigatória a colocação desse caractere no comando *if – else*, pois é através dele que o interpretador Python identifica o término do cabeçalho do comando e o início dos comandos que lhe estão subordinados.

Essa relação de subordinação é importante na lógica do algoritmo. Neste exemplo a linha ④ está subordinada ao *if* da linha ③ e as linhas ⑥ e ⑦ estão subordinadas ao *else* da linha ⑤.

Identação

O interpretador Python identifica a relação de subordinação descrita no parágrafo anterior pelo recuo que há na digitação das linhas do programa. Note que as linhas subordinadas estão digitadas com alguns espaços em branco à esquerda. Isso recebe o nome de identação e, em Python, ela é obrigatória sempre que houver um ou mais comandos subordinados a outro. O *else* por sua vez não é identado e para que o programa fique correto é preciso que ele fique exatamente no mesmo alinhamento do *if* ao qual está associado.

Construindo Condições Simples

No exemplo anterior foi construída uma condição que avaliava se um valor contido na variável B era igual a zero ou não. Para isso usou-se a construção *B == 0*, onde B é uma variável e 0 é um número literal

Generalizando pode-se dizer que condições podem ser escritas da seguinte forma:

{Expressão do lado esquerdo} {operador} {Expressão do lado direito}

Onde:

As **expressões** em ambos os lados podem ser:

- um literal (geralmente número ou texto)
- uma variável
- uma fórmula (expressão aritmética)
- uma chamada de função (isso será visto mais tarde)

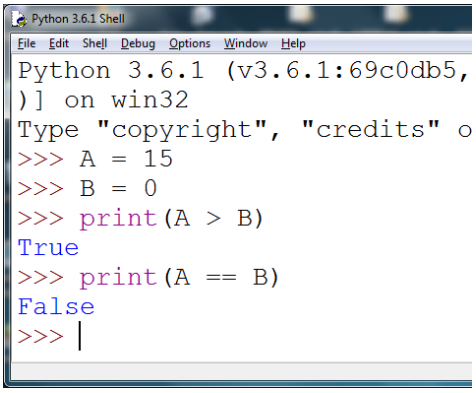
O **operador** é um dos seis operadores relacionais exibidos nesta tabela à direta. Nos caso dos operadores que contém dois caracteres, não é permitido haver espaço em branco entre eles.

==	Igual a
!=	Diferente de
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a

Exemplos de condições simples

Condição	Interpretação	Elementos envolvidos
`A > 0`	A maior que zero	Compara variável com literal numérico (0)
`X <= Y`	X menor ou igual a Y	Compara duas variáveis
`X != A + B`	X é diferente de A + B	Compara variável com o resultado da expressão aritmética
`C > 2*(A+B)`	C é maior que 2(A+B)	Compara variável com o resultado da expressão aritmética
`10*A < 100*B`	10A é maior que 100B	Compara os resultados de duas expressões aritméticas
`S == ""`	S igual a string vazio	Compara variável com literal texto vazio
`S != "SIM"`	S diferente de "SIM"	Compara variável com literal texto não vazio

Exercícios – Lote 1



Considerando os valores fornecidos, avalie cada condição e informe se o resultado é falso (False) ou verdadeiro (True). Faça o teste dessas condições no Shell do Python conforme mostrado na figura.

	Valores	Condição	Resultado
1	Para A = 0 e B = -3	A > B	
2	Para X = 3.7	X <= 10.0	
3	Para A = 3, B = 9 e C = 5	10 * A >= B * C	
4	Para A = 3, B = 6 e C = 5	10 * A >= B * C	
5	Para N = "MORANGO"	N == "BANANA"	
6	Para N = "MORANGO"	N > "BANANA"	

Construindo Condições Compostas

Muitas vezes é preciso combinar duas ou mais condições simples em uma condição composta. Dessa forma, as condições simples vistas anteriormente são a base para a construção das condições compostas. A construção de uma condição composta tem a seguinte forma:

`{Condição 1}  {Operador Lógico}  {Condição 2}`

Onde:

As condições 1 e 2 são duas condições simples já vistas

O operador lógico é um dos dois operadores ao lado. Antes de seguir adiante, é preciso saber como avaliar expressões que contenham esses operadores *and* e *or*. Para isso estão postas abaixo as tabelas verdades de ambos.

and	Operação lógica E
or	Operação lógica OU

Tabela Verdade *and* Para que *and* resulte verdadeiro ambos, C1 e C2, devem ser verdadeiros		
C1	C2	C1 *and* C2
False	False	False
False	True	False
True	False	False
True	True	True

Tabela Verdade *or* Para que *or* resulte verdadeiro pelo menos um dos dois, C1 ou C2, deve ser verdadeiro		
C1	C2	C1 *or* C2
False	False	False
False	True	True
True	False	True
True	True	True

Exemplos de condições compostas

Condição	Interpretação
`A > 0 and B > 0`	O resultado da condição composta será verdadeiro se A e B forem ambos maiores que zero no momento da avaliação
`X <= Y and Y != 0`	O resultado da condição composta será verdadeiro somente se X for menor ou igual a Y ao mesmo tempo que Y seja diferente de zero.
`X == 0 or X > 2000`	O resultado da condição composta será verdadeiro se X for igual a zero ou se X for maior que 2000

Exercícios – Lote 2

Considerando os valores fornecidos, avalie cada condição composta e informe se o resultado é falso (False) ou verdadeiro (True). Faça o teste dessas condições no Shell do Python.

	A	B	C	Condição	Resultado
1	10	15	4	`A < B and A < C`	
2	10	15	4	`A < B or A < C`	
3	1	9	0	`A >= 0 and B == C`	
4	1	9	9	`A >= 0 and B == C`	
5	1	9	0	`A >= 0 or B == C`	
6	1	9	9	`A >= 0 or B == C`	
7	0	0	0	`B != 0 and A != C`	
8	0	0	25	`B != 0 and A != C`	
9	0	0	0	`B != 0 or A != C`	
10	0	0	25	`B != 0 or A != C`	

Negação

Além dos operadores lógicos and e or, existe também o operador not. Este, no entanto, é diferente dos outros dois. Ele é aplicável apenas a uma condição, da seguinte forma:

```
not {Condição}
```

Ele tem o efeito de inverter o resultado da Condição à qual é aplicado.

Valores	Condição	Negação	Interpretação
`Para A = 0`	`A == 0` resulta verdadeiro	`not (A == 0)` resulta falso	Como A é, de fato, igual a zero então a Condição resulta verdadeiro e sua negação será falso. (Na prática é o mesmo perguntar se A é diferente de zero)
`Para A = 3`	`A == 0` resulta falso	`not (A == 0)` resulta verdadeiro	Como A é 3 a Condição resulta em falso e, portanto, sua negação será verdadeiro.

Condições Compostas Mistas

Em uma única condição composta é possível misturar not, and e or. Quando isso ocorre é necessário ter atenção à precedência com que esses operadores são considerados. Existe uma ordem de prioridade a ser respeitada. Essa prioridade segue a seguinte ordem: not primeiro, and em seguida e or por último.

Assim, na avaliação de uma condição composta mista a prioridade acima sempre será seguida. É necessário ter o devido cuidado ao construir condições assim e verificar que a falta de atenção pode levar a erros. Como exemplo veja as duas expressões abaixo e avalie-as para A = 15, B = 9, C = 9

`B == C or A < B and A < C` Resultará Verdadeiro

`(B == C or A < B) and A < C` Resultará Falso

O uso de parênteses serve para alterar a ordem de prioridade na avaliação de expressões lógicas. Uma vez inseridos, os parênteses estabelecem qual (ou quais) parte(s) serão avaliada(s) primeiro.

Exercícios – Lote 3

Considerando os valores fornecidos, avalie cada condição composta e informe se o resultado é falso (False) ou verdadeiro (True). Faça o teste dessas condições no Shell do Python.

	A	B	C	Condição	Resultado
1	10	15	4	`A < B and A < C or C != 0`	
2	10	15	4	`A < B and (A < C or C != 0)`	
3	1	9	0	`not (A >= 0 and B == C)`	
4	1	9	9	`Not (A >= 0) and not (B == C)`	
5	1	9	0	`(A >= 0 or B == C) and B > A`	

Comando Condicional Completo

Retomando agora as explicações sobre o comando condicional, abaixo está sua forma completa.

```
if {condição 1}:
  {bloco de comandos 1}
elif {condição 2}:
  {bloco de comandos 2}
elif {condição 3}:
  {bloco de comandos 3}
...
else:
  {bloco de comandos do else}
```

As partes *if* e *else* já foram explicadas anteriormente. A parte *elif* permite que sejam usadas condições adicionais e confere a possibilidade de tomada de decisão entre múltiplas opções.

A execução deste comando inicia pela avaliação da {condição 1} e se ela for verdadeira será executado o {bloco de comandos 1} e pulam-se todos os demais; caso a {condição 1} seja falsa, passa-se para a avaliação da {condição 2} e caso seja verdadeira será executado o {bloco de comando 2} pulando-se os demais, e assim sucessivamente. Ao final, se nenhuma das condições postas for verdadeira, então executa-se o {bloco de comandos do else}.

Não há limites para a quantidade de partes *elif* a serem usadas, de modo que o programador é livre para usar tantas dessas partes quanto for a necessidade do algoritmo.

E, por fim, é preciso dizer que as partes elif e else são opcionais, de modo que se o programador não precisar incluí-las em seu algoritmo, elas podem simplesmente ser omitidas.

Exemplo

```
PH = float(input("Digite um valor do PH: "))    ①
if PH < 7.0:                                    ②
  print("Solução ácida")                        ③
elif PH == 7.0:                                 ④
  print("Solução neutra")                       ⑤
else:                                           ⑥
  print("Solução básica")                       ⑦
```

Exercícios – a solução de cada exercício a seguir é um Programa completo

1. Escreva um programa que leia um número real X e apresente na tela a mensagem "Maior que zero" caso ocorra X > 0. Se X for menor ou igual a zero não faça nada.
2. Escreva um programa que leia dois números inteiros A e B e mostre na tela apenas o menor dos dois.
3. Escreva um programa que leia dois números quaisquer e mostre na tela qual é o menor e qual é o maior.
4. Escreva um programa que leia um número inteiro e apresente na tela se ele é par ou ímpar (para determinar se um número é par ou ímpar verifique se o resto da divisão dele por 2 é zero ou não. Para isso use o operador % para calcular esse resto).
5. Escreva um programa que leia um número inteiro e informe se o mesmo é positivo, zero ou negativo.
6. Escreva um programa que leia o nome de um lutador e seu peso. Em seguida informe a categoria a que pertence o lutador, conforme a tabela ao lado (note que a tabela foi criada para efeito deste exercício e não condiz com qualquer categoria de luta). A saída do programa deve exibir na tela um texto no seguinte padrão:

Peso	**Categoria**
Menor que 65 kg	Pena
Maior ou igual a 65 kg e menor que 72 kg	Leve
Maior ou igual a 72 kg e menor que 79 kg	Ligeiro
Maior ou igual a 79 kg e menor que 86 kg	Meio médio
Maior ou igual a 86 kg e menor que 93 kg	Médio
Maior ou igual a 93 kg e menor que 100 kg	Meio pesado
Maior ou igual a 100 kg	Pesado

   Nome fornecido: Pepe Jordão
   Peso fornecido: 73.4
   Saída exibida na tela: O lutador Pepe Jordão pesa 73.4 kg e se enquadra na categoria Ligeiro
7. Escreva um programa que leia três números reais A, B e C que são os coeficientes de uma equação do 2º grau ($A.x^2 + B.x + C = 0$). Calcule e apresente na tela as raízes dessa equação, considerando os três casos possíveis: Delta maior que zero (duas raízes reais), Delta igual a zero (uma raiz) e Delta menor que zero (não há raízes reais).
8. Escreva um programa que leia três números reais e informe se eles constituem os lados de um triângulo. Em caso afirmativo, informe se o triângulo é equilátero, isósceles ou escaleno. Para que três números formem um triângulo deve ocorrer que a soma dos dois lados menores deve ser maior que o lado maior. Para resolver essa questão será preciso usar os operadores and e or.