Comandos Condicionais

CIÊNCIA DA COMPUTAÇÃO

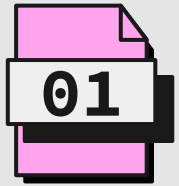
ATITUS EDUCAÇÃO

- Aula 03 - Pensamento Computacional

Prof. Me. Lucas R. C. Pessutto

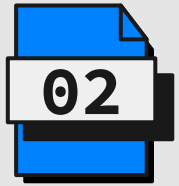
ATITUS EDUCAÇÃO

Na aula de hoje...



Expressões Lógicas

Operadores Relacionais e Lógicos
Precedência de Operadores



Estruturas Condicionais

Condicional Simples
Condicional Duplo

Operadores Relacionais

* Utilizados para comparar variáveis e valores

* Apresentam como saída uma resposta lógica:
  - Verdadeiro (True)
  - Falso (False)

* Exemplos:

8 é maior do que 5?	`8 > 5`	`=> True`
3 é negativo?	`3 < 0`	`=> False`
10 é diferente de 10?	`10 != 10`	`=> False`
nota é igual a 5?	`nota == 5`	`=> ? (Depende)`

Operadores Relacionais

Operador	Exemplo	Descrição
==	x == y	Verifica se o conteúdo de x é igual ao de y
!=	x != y	Verifica se o conteúdo de x é diferente do de y
<=	x <= y	Verifica se x é menor ou igual à y
>=	x >= y	Verifica se x é maior ou igual à y
<	x < y	Verifica se x é menor do que y
>	x > y	Verifica se x é maior do que y

Exemplo:

```
a = 4
resp = (a == 10)
print(resp)
```

```
False
```

Operadores Relacionais

* Avalie as seguintes expressões usando o modo iterativo:

```
>>> 10 == 20
>>> 10 != 20
>>> 10 <= 20
```

```
>>> '2' < '9'
>>> '12' < '9'
>>> '12' > '9'
```

```
>>> 'Ola' == "Ola"
>>> 'Ola' < 'Zebra'
>>> 'Ola' < 'ZEBRA'
```

Comparação de strings

* As strings são ordenadas de acordo com a codificação de seus caracteres

* Em geral, essa codificação segue uma ordenação lexicográfica

* Por exemplo: ‘Ana’ < ‘Lucas’ porque a representação em Unicode (ASCII) para o caractere ‘A’ é 65, enquanto que a representação do caractere ‘L’ é 76.

* Porém, ‘ana’ > ‘Lucas’ pois a representação das letras minúsculas costuma vir após a das letras maiúsculas. ‘a’ (97) > ‘L’ (76)

ASCII Table

https://en.wikipedia.org/wiki/ASCII

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Comparação de strings

* Python possui duas funções para converter um caractere para seu encoding e vice-versa:

`ord(caractere)`: retorna a codificação do caractere informado

`chr(codigo)`: retorna o caractere que possui o código informado

```
>> ord('A')
65

>> chr(65)
A
```

Operadores Lógicos ou Booleanos

* Utilizados para combinar variáveis lógicas

* Apresentam como saída uma resposta lógica:
  Verdadeiro (True)
  Falso (False)

* Exemplo:
* Vou à praia se tiver sol **e** tiver dinheiro
* Vou visitar meu pai se for dia dos pais **ou** final de semana

* Hoje tem sol, mas não tenho dinheiro. Vou à praia? NÃO
* Hoje é sábado. Visito meu pai? SIM

Operadores Lógicos ou Booleanos

Operador	Tipo	Significado
and	Binário	“e” lógico
or	Binário	“ou” lógico
not	Unário	Negação

* Tabelas Verdade:

Operador **or**		
Operando 1	Operando 2	**Resultado**
True	True	**True**
True	False	**True**
False	True	**True**
False	False	**False**

Operador and		
Operando 1	Operando 2	Resultado
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

Operador **not**	
Operando	**Resultado**
True	**False**
False	**True**

Expressões Lógicas

```
'''
Programa que exemplifica a utilização de expressões lógicas em Python
'''
a = 1
b = 3
c = 5
d = 7
valor = 9
e1 = (a > b) or (c < d) => True
e2 = valor > 10 => False
e3 = not (a + c < b) => True
e4 = (a == b) => False
e5 = (a == b) and (a >= c) => False
e6 = (a != b) => True
e7 = not ((a >=b) or (a < c)) => False

print(e1, e2, e3, e4, e5, e6, e7)
```

Operadores Lógicos

Ex: Qual expressão lógica faz o que é pedido?

1. codigo entre 1 e 100 (valores limites inclusos)

```
codigo >= 1 and codigo <= 100
```

```
codigo > 0 and codigo < 101
```

2. contador maior ou igual a máximo ou valor igual a procurado

```
contador >= maximo or valor == procurado
```

3. valor igual a 1 ou 2 ou entre 4 e 8 (sem considerar valores limites)

```
valor == 1 or valor == 2 or (valor > 4 and valor < 8)
```

Observações sobre Operadores

* É permitido “encurtar” expressões lógicas, por exemplo:

  `a < b < c` é o mesmo que `a < b and b < c`

* Os valores verdadeiro e falso são constantes chamadas `True` e `False`

* Não confundir o operador de atribuição `(=)` com o operador lógico de igualdade `(==)`

Precedência de Operadores

Ordem	Operador
1°	( )
2°	* / // %
3°	+ -
4°	== != > >= < <=
5°	not
6°	and
7°	or

Na expressão: `(x != 10 or y > 1 and y < 10)`
Pela precedência dos operadores, primeiro serão resolvidos > e <:

```
y > 1 e y < 10
```

O próximo operador a ser considerado, pela precedência será !=:

```
x != 10
```

Restam os operadores and e or, como and é o de maior prioridade, será o próximo:

```
y < 1 and y < 10
```

Finalmente o or será resolvido.

A versão a seguir com parênteses mostra a ordem de execução da expressão:

```
((x != 10) or (( y > 1) and (y < 10)))
```

RECOMENDAÇÃO: usar parênteses não só para alterar a precedência de operadores, como também para tornar mais claras as expressões.

Exemplo

`(x != 10 or y > 1 and y < 10)` Para x = 10 e y = 1

`y > 1`	`y < 10`	`x != 10`	`y > 1 and y < 10`	`(x != 10 or y > 1 and y < 10)`
False	True	False	False	False

`(x != 10 or y > 1 and y < 10)` Para x = 12 e y = 5

`y > 1`	`y < 10`	`x != 10`	`y > 1 and y < 10`	`(x != 10 or y > 1 and y < 10)`
True	True	True	True	True

Programas com Estruturas Condicionais

Comandos de Seleção

Condicional Simples

```
if condicao:
    <comando>
```

Seleção Dupla

```
if condicao:
    <comando>
elif condicao:
    <comando>
else:
    <comando>
```

Seleção Múltipla

```
match expressão:
    case padrao2: <comando>
    case padrao2: <comando>
    ...
```

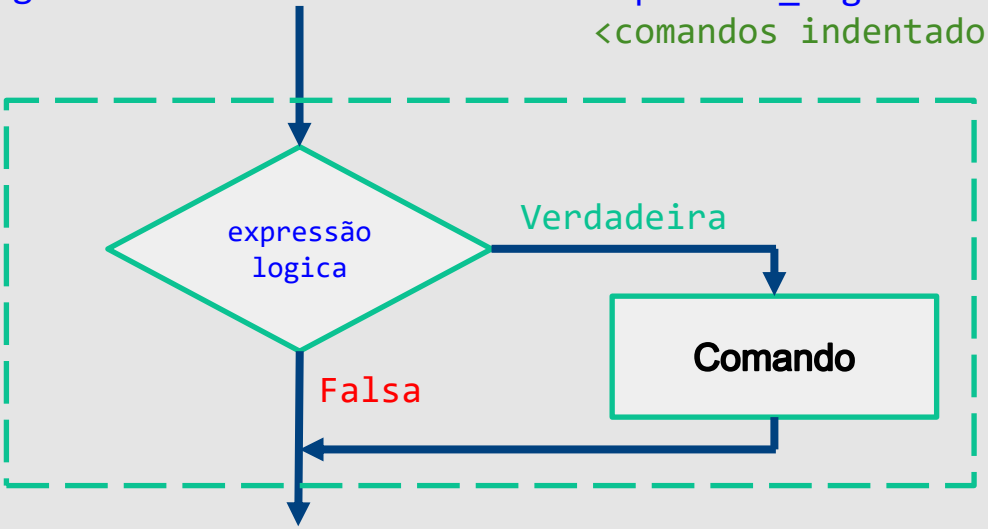
Comando Condicional Simples

Linguagem Algorítmica

```
se <expressão_logica> entao
inicio
    <comando>
fimse
```

Linguagem Python

```
if <expressão_logica>:
    <comandos indentados>
```



Indentação

* Forma de hierarquizar as linhas do programa para tornar o código mais legível
* Em Python a indentação é obrigatória
* Utiliza determinado espaçamento a partir da margem esquerda

```
1   #Início do bloco 0
2   comandos do bloco 0
3     #Início do bloco 1
4     comandos do bloco 1
5       #Início do bloco 2
6       comandos do bloco 2
7       ...
8       #Fim do bloco 2
9     #Fim do bloco 1
10  #Fim do bloco 0
11
```

Indentação

* O comando if em Python é um exemplo de utilização obrigatória de indentação

* Geralmente utiliza-se o comando TAB

```
<comando fora do bloco>
if <expressão lógica>:
    ⇢ <comando dentro do bloco>
    ⇢ <comando dentro do bloco>
<comando fora do bloco>
```

Indentações com o comando TAB indicando que os comandos pertencem ao bloco do if acima

Problema 01: Mensagem

Enunciado de um problema:

Ler um valor e, no caso de ser menor do que 10, emitir uma mensagem

```python
numero = int(input("Informe um valor"))

if numero < 10:
    print(f"{numero} é menor do que 10")
```

Indentação!

```
>> python mensagem.py
Informe um valor: 15

>> python mensagem.py
Informe um valor: -15
-15 é menor do que 10
```

Problema 02: Livraria

Enunciado de um problema:

Processar uma venda de livros em uma livraria. Obter código do tipo de livro vendido (A, B, C) e número de unidades, calcular e informar valor a pagar. Valores por tipo: A (R$10,00); B (R$20,00); C (R$30,00). Caso tenham sido vendidos mais de 10 livros, emitir uma mensagem.

Problema 02: Livraria

```
'''
Programa que processa uma venda e avisa caso tenha sido vendidas mais de 10 unidades
'''
PRECO_A = 10
PRECO_B = 20
PRECO_C = 30
codigo = input("Informe o tipo do livro (A, B ou C): "
quantidade = int(input("Informe a quantidade vendida:

if codigo == "a" or codigo == "A":
    total = quantidade * PRECO_A

if codigo == "b" or codigo == "B":
    total = quantidade * PRECO_B

if codigo == "c" or codigo == "C":
    total = quantidade * PRECO_C

print(f"Total a ser pago: R${total:6.2f}

if quantidade > 10:
    print(f"Foram vendidos mais de 10 li
```

```
>> python 03-livraria.py
Informe o tipo do livro (A, B ou C): A
Informe a quantidade vendida: 4

Total a ser pago: R$ 40.00
```

```
>> python 03-livraria.py
Informe o tipo do livro (A, B ou C): b
Informe a quantidade vendida: 18

Total a ser pago: R$360.00
Foram vendidos mais de 10 livros do tipo b
```

```
>> python 03-livraria.py
Informe o tipo do livro (A, B ou C): j
Informe a quantidade vendida: 8
```

Erro!

Problema 03: Plano Cartesiano

Enunciado de um problema:

Dados um par de valores x e y, que representam as coordenadas de um ponto no plano, determinar a localização do ponto: se em um quadrante, em um dos eixos ou na origem.

Definição dos elementos algoritmo:

Entradas: coordenadas de um ponto no plano cartesiano (x,y)

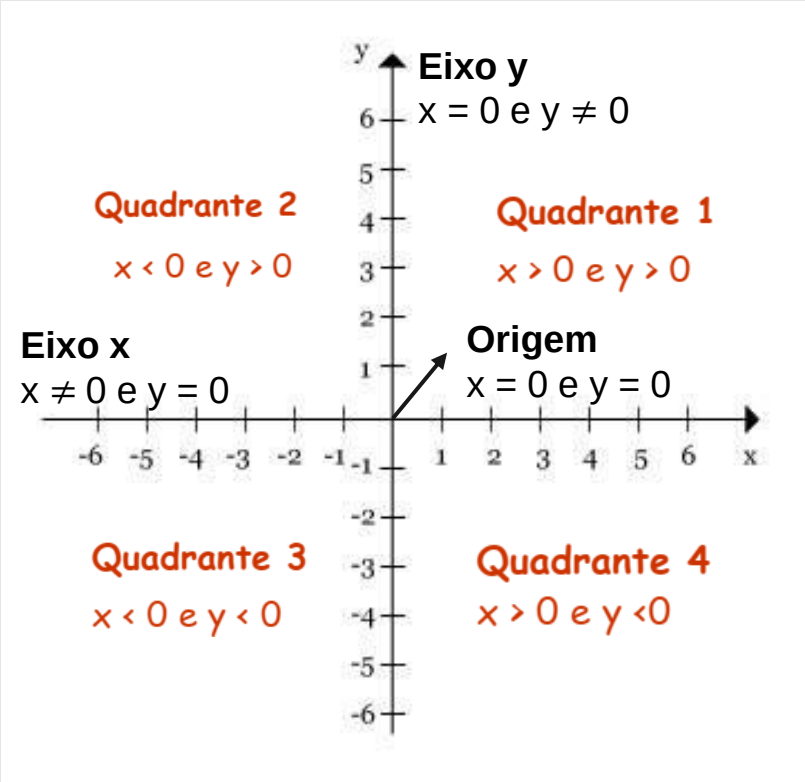
Saída: Mensagem, indicando a localização do ponto no plano

Processamento: testes, de modo a cobrir todas as possíveis localizações do ponto

Problema 03: Plano Cartesiano

Análise do Problema

1. leitura dos valores de x e y.
2. determinação, pela avaliação de condições, de onde o ponto se encontra: se em um quadrante, em um eixo ou na origem.
3. escrita da mensagem, onde é indicada a localização do ponto – apenas 1 mensagem por par de valores.



Problema 03: Plano Cartesiano

```python
x, y = input("Informe as coordenada do ponto: ").split(" ")
x, y = int(x), int(y)

if x == 0 and y == 0:
    print("Ponto na origem")
if x > 0 and y > 0:
    print("Quadrante 1")
if x < 0 and y > 0:
    print("Quadrante 2")
if x < 0 and y < 0:
    print("Quadrante 3")
if x > 0 and y < 0:
    print("Quadrante 4")
if x != 0 and y == 0:
    print("Ponto sobre o Eixo x")
if x == 0 and y != 0:
    print("Ponto sobre o Eixo y")
```

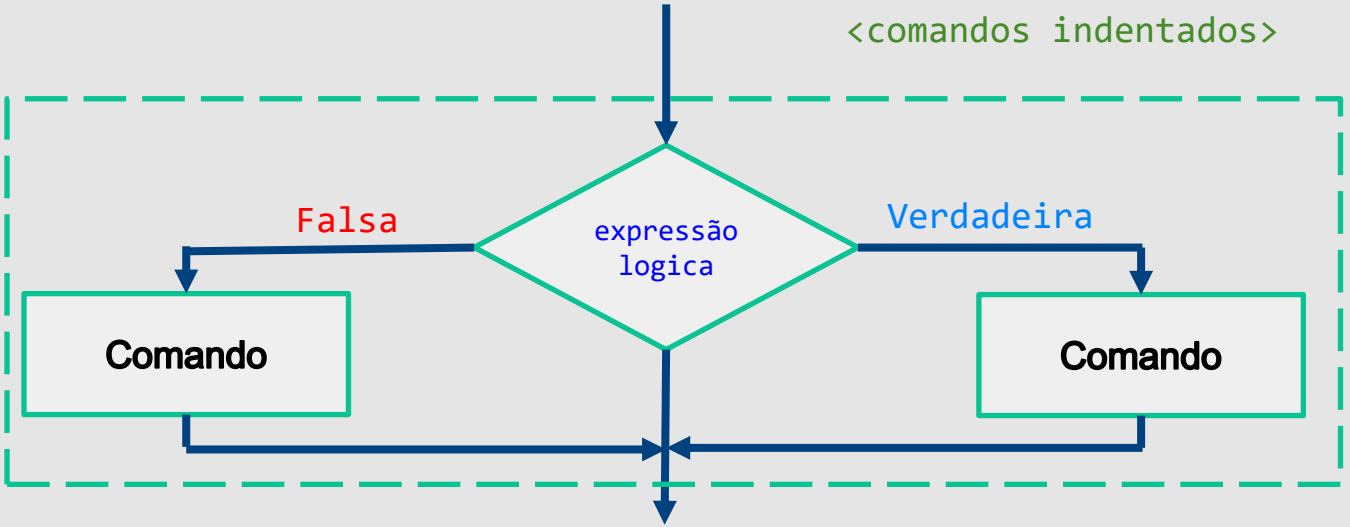
Comando Condicional Duplo

Linguagem Algorítmica

```
se <expressão_logica> entao
    inicio
        <comandos>
    fim
senao
    inicio
        <comandos>
    fim
fimse
```

Linguagem Python

```
if <expressão_logica>:
    <comandos indentados>
else:
    <comandos indentados>
```



Problema 04: Par ou ímpar

Enunciado de um problema:

Fazer um programa que leia um número natural e informe se o número lido é par ou ímpar.

```python
valor = int(input("Digite o valor a ser testado: "))

if valor % 2 == 0:
    print(f"{valor} é par!")
else:
    print(f"{valor} é ímpar!")
```

```
>> python 05-par-impar.py
Digite o valor a ser testado: 25
25 é ímpar!

>> python 05-par-impar.py
Digite o valor a ser testado: 98
98 é par!
```

Problema 02: Livraria

```
'''
Programa que processa uma venda e avisa caso tenha sido vendidas mais de 10 unidades
'''
PRECO_A = 10
PRECO_B = 20
PRECO_C = 30
codigo = input("Informe o tipo do livro (A, B ou C): ")
quantidade = int(input("Informe a quantidade vendida: "))

if codigo == "a" or codigo == "A":
    total = quantidade * PRECO_A

if codigo == "b" or codigo == "B":
    total = quantidade * PRECO_B

if codigo == "c" or codigo == "C":
    total = quantidade * PRECO_C

print(f"Total a ser pago: R${total:6.2f}")

if quantidade > 10:
    print(f"Foram vendidos mais de 10 livros do tipo {codigo}")
```

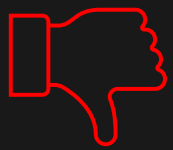
Códigos Redundantes!
Nesse caso a melhor solução é encadear os testes.

Problema 02: Livraria

```
...

if codigo == "a" or codigo == "A":
    total = quantidade * PRECO_A
else:
    if codigo == "b" or codigo == "B":
        total = quantidade * PRECO_B
    else:
        if codigo == "c" or codigo == "C":
            total = quantidade * PRECO_C
        else:
            print("Código Inexistente!")

print(f"Total a ser pago: R${total:6.2f}")

if quantidade > 10:
    print(f"Foram vendidos mais de 10
```

Verificação de consistência da entrada. Isso resolve o problema que estávamos tendo?

```
>> python 06-livraria-v2.py
Informe o tipo do livro (A, B ou C): j
Informe a quantidade vendida: 8
Código Inexistente!
```



ATITUS EDUCAÇÃO

Comando Condicional Duplo com elif

* Funciona como um else seguido de um if
* Significa “se a condição anterior não foi verdadeira então teste essa”
* Serve para encadear ifs de forma a melhorar a legibilidade do código

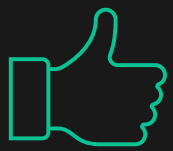
```
if <expressão_logica>:
    <comandos indentados>
elif <expressão_logica>:
    <comandos indentados>
elif <expressão_logica>:
    <comandos indentados>
...
elif <expressão_logica>:
    <comandos indentados>
...
else:
    <comandos indentados>
```

Problema 02: Livraria

```
...
if codigo == "a" or codigo == "A":
    total = quantidade * PRECO_A
elif codigo == "b" or codigo == "B":
        total = quantidade * PRECO_B
elif codigo == "c" or codigo == "C":
    total = quantidade * PRECO_C
else:
    total = 0
    print("Código Inexistente!")

if total != 0:
    print(f"Total a ser pago: R${total:6.2f}")

    if quantidade > 10:
        print(f"Foram vendidos m
```

Nova forma de verificar consistência.
Problema resolvido?

```
>> python 07-livraria-v3.py
Informe o tipo do livro (A, B ou C): j
Informe a quantidade vendida: 8
Código Inexistente!
```



ATITUS EDUCAÇÃO

Uso da biblioteca de strings

* Múltiplos testes relacionais entre maiúscula e minúscula

```
if codigo == "a" or codigo == "A":
    total = quantidade * PRECO_A
elif codigo == "b" or codigo == "B":
    total = quantidade * PRECO_B
elif codigo == "c" or codigo == "C":
    total = quantidade * PRECO_C
```

* As funções `upper` (ou `lower`) podem ser utilizadas para simplificar este teste

```
codigo = codigo.upper()

if codigo == "A":
    total = quantidade * PRECO_A
elif codigo == "B":
    total = quantidade * PRECO_B
elif codigo == "C":
    total = quantidade * PRECO_C
```

Programar é um processo iterativo e incremental!

ifs encadeados (aninhados)

```
if <condição_1>:
    <comando 1>;
    if <condição_2>:
        <comando 2>;
        <comando 3>;
    <comando 4>;
```

```
if <condição_1>:
    if <condição_2>:
        if <condição_3>:
            <comando 1>;
```

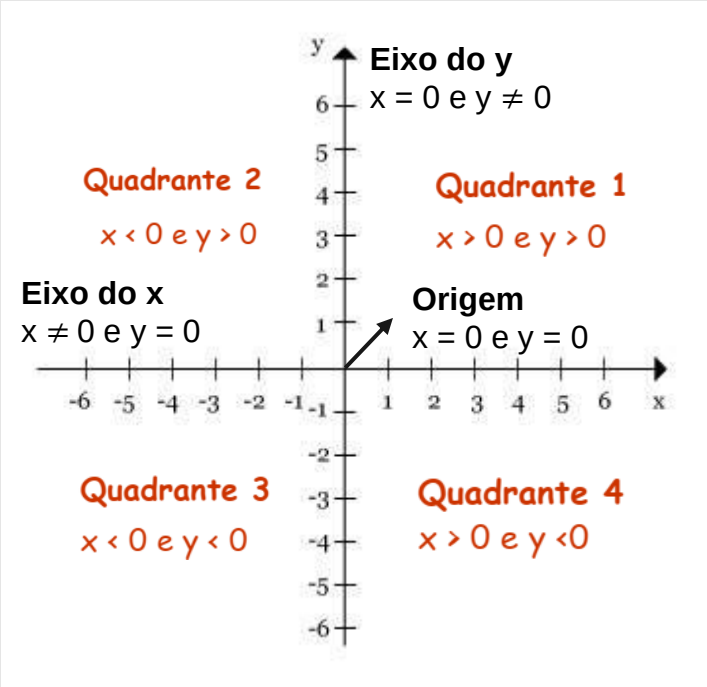
```
if <condição_1>:
    if <condição_2>:
        if <condição_3>:
            <comando 1>;
        else:
            <comando 2>;
    else:
        <comando 3>;
```

Que comandos são executados se condição_1 é verdadeira, condição_2 é falsa e condição_3 é verdadeira?

E se condição_1 é verdadeira, condição_2 é verdadeira e condição_3 é falsa?

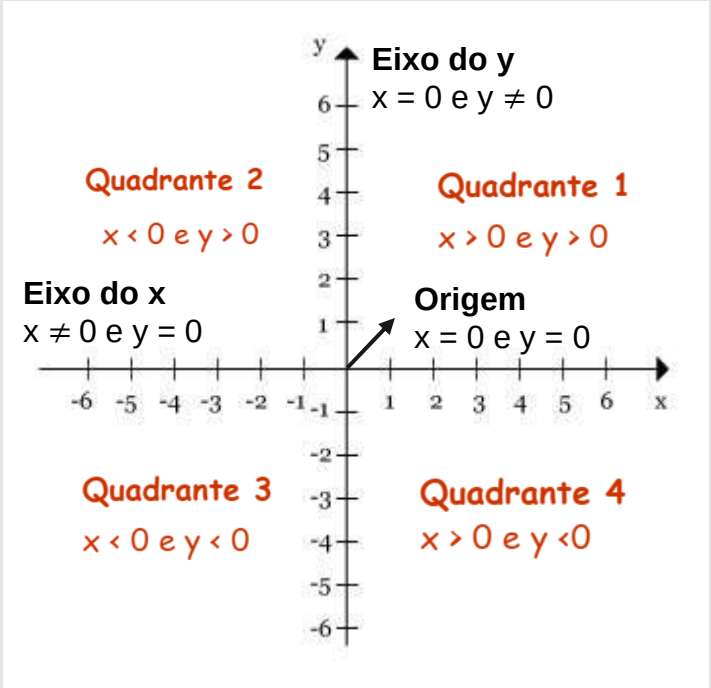
Retornando ao problema do Plano Cartesiano

```
x, y = input("Informe as coordenada do ponto: ").split(" ")
x, y = int(x), int(y)

if x == 0 and y == 0:
    print("Ponto na origem")
if x > 0 and y > 0:
    print("Quadrante 1")
if x < 0 and y > 0:
    print("Quadrante 2")
if x < 0 and y < 0:
    print("Quadrante 3")
if x > 0 and y < 0:
    print("Quadrante 4")
if x != 0 and y == 0:
    print("Ponto sobre o Eixo x")
if x == 0 and y != 0:
    print("Ponto sobre o Eixo y")
```

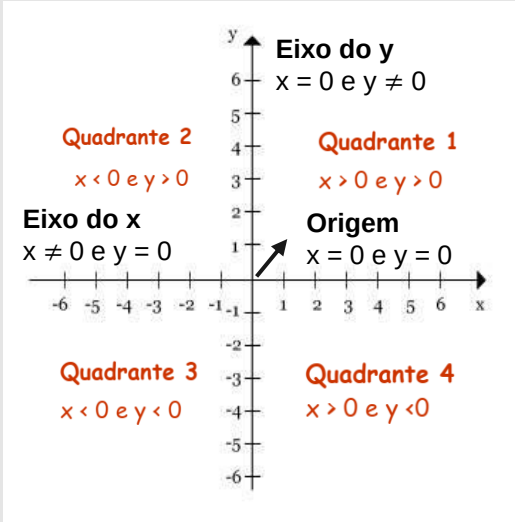


Retornando ao problema do Plano Cartesiano

```
x, y = input("Informe as coordenada do ponto: ").split(" ")
x, y = int(x), int(y)

if x == 0 and y == 0:
    print("Ponto na origem")
elif x > 0 and y > 0:
    print("Quadrante 1")
elif x < 0 and y > 0:
    print("Quadrante 2")
elif x < 0 and y < 0:
    print("Quadrante 3")
elif x > 0 and y < 0:
    print("Quadrante 4")
elif x != 0 and y == 0:
    print("Ponto sobre o Eixo x")
elif x == 0 and y != 0:
    print("Ponto sobre o Eixo y")
```



```python
x, y = input("Informe as coordenada do ponto: ").split(" ")
x, y = int(x), int(y)

# Verifica se o ponto está na origem
if x == 0:
    if y == 0: # Como x é 0, então o ponto está na origem ou no eixo y
        print("Ponto na origem")
    else:   # O ponto só pode estar no eixo y
        print("Ponto sobre o Eixo y")
else:   # Se entrar aqui, temos certeza que x != 0
    if y == 0: # O ponto está no eixo x
        print("Ponto sobre o Eixo x")
    else: # só sobraram os quadrantes!
        if x > 0: # quadrantes 1 ou 4, dependendo de y
            if y > 0:  # quadrante 1
                print("Quadrante 1")
            else: # quadrante 4, sem precisar mais testes
                print("Quadrante 4")
        else: #  x é < 0: sobraram quadrantes 2 e 3, dependendo de y
            if y > 0:
                print("Quadrante 2")
            else: # // sobrou x < 0 e y < 0: não precisa testar
                print("Quadrante 3")
```



Exercício

Fazer o programa para calcular e informar as raízes de uma equação do 2º grau. Os valores das variáveis a, b e c devem ser fornecidos via teclado.