MAC2166 - Introdução à Computação

Prof. Dr. Helder Oliveira

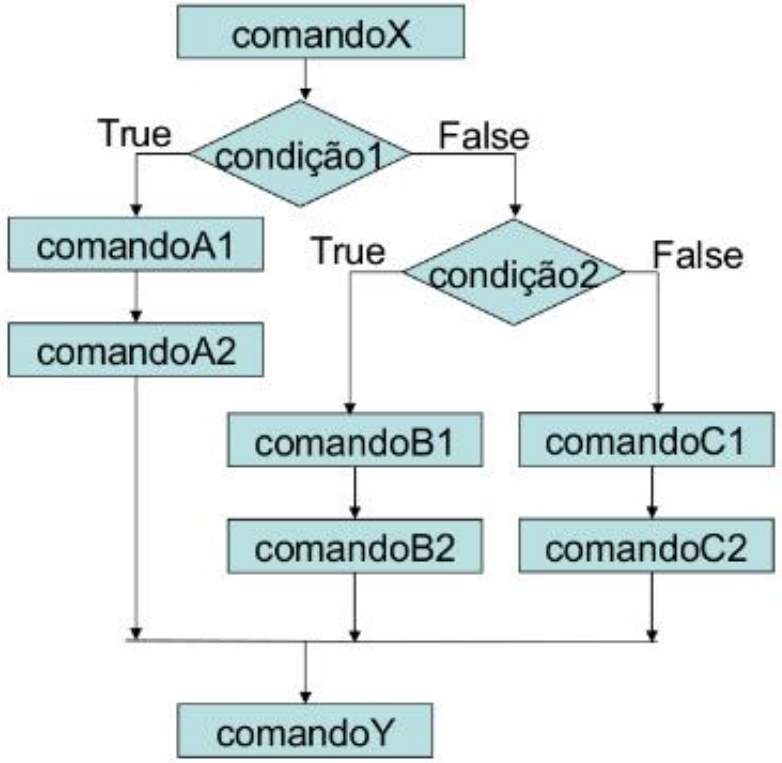
Agenda

- Operadores Aritméticos de Atribuição +=,-=,*=,/=,%=.
- Função de conversão float(), que converte de str para float.
- Diferença entre / e // e como % se comporta com float.
- Execução condicional encadeada if-elif-else.

Operadores Aritméticos de Atribuição

Operador	Exemplo	É equivalente a:
***=**	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5

Condicional encadeada




```
comandoX
if condição1 :
    # bloco de comandos.
    comandoA1
    comandoA2
elif condição2 :
    # bloco de comandos.
    comandoB1
    comandoB2
else :
    # bloco de comandos.
    comandoC1
    comandoC2
comandoY
```

Exercicio

- Dados dois inteiros x e y, indicar se eles são iguais ou qual é o maior entre eles.

```
1  if x < y :
2      print("x é menor do que y.")
3  else:
4      if x > y :
5          print("x é maior do que y.")
6      else:
7          print("x e y são iguais.")
```

```
1  if x < y :
2      print("x é menor do que y.")
3  elif x > y :
4      print("x é maior do que y.")
5  else:
6      print("x e y são iguais.")
```

Expressões Lógicas

- Expressões lógicas são aquelas que realizam uma operação lógica e retornam verdadeiro ou falso (como as expressões relacionais).
- Os operadores lógicos são:
  - **and**: operador E.
  - **or**: operador OU.
  - **not**: operador NÃO.

Operador Lógico and

- <expressão1> and <expressão2>: retorna verdadeiro quando ambas as expressões são verdadeiras.
- Sua tabela verdade é:

<expressão1>	<expressão2>	resultado
True	True	True
True	False	False
False	True	False
False	False	False

```
1 a = 5
2 b = 10
3 print((a > 0) and (b == 0))
4 # False
5 print((a > 0) and (b != 0))
6 # True
```

Operador Lógico or

- <expressão1> or <expressão2>: retorna verdadeiro quando pelo menos uma das expressões é verdadeira.
- Sua tabela verdade é:

<expressão1>	<expressão2>	resultado
True	True	True
True	False	True
False	True	True
False	False	False

```
1 a = 5
2 b = 10
3 print((a > 0) or (b == 0))
4 # True
5 print((a != 5) or (b == 0))
6 # False
```

Operador Lógico not

- not <expressão>: retorna verdadeiro quando a expressão é falsa e vice-versa.
- Sua tabela verdade é:

<expressão>	resultado
True	False
False	True
<expressão>	resultado

```
1a = 5
2b = 10
3print(not(a < b))
4# False
5print(not(a == b))
6# True
```

Operadores lógicos

A	B	A and B	A or B
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

A	not A
False	True
True	False

Expressões Equivalentes

- not(a == b) é equivalente a (a != b)
- not(a > b) é equivalente a (a <= b)
- not(a < b) é equivalente a (a >= b)

Exemplo

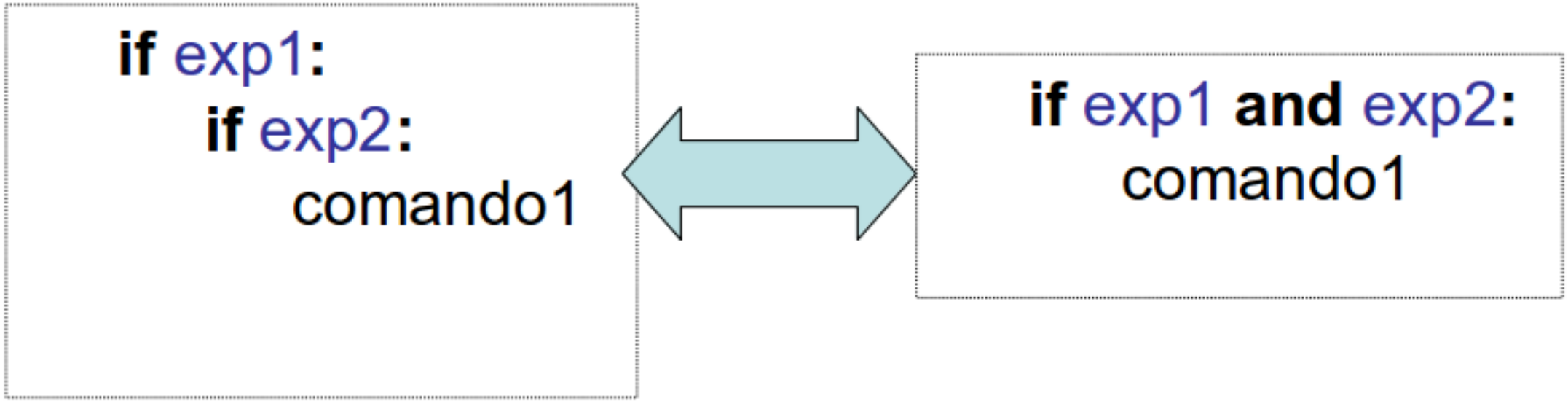
- O que será impresso pelo código a seguir?

```
a = True
b = False
print(not(a or b))
# False
print(not(a and b))
# True
print(not(a) and not(b))
# False
print(not(a) or not(b))
# True
```

Precedência de Operadores

Nível	Categoria	Operadores
7(alto)	exponenciação	**
6	multiplicação	*,/,//,%
5	adição	+,-
4	relacional	==,!=,<=,>=,>,<
3	lógico	not
2	lógico	and
1(baixo)	lógico	or

Equivalências: and



Equivalências: or

```
if exp1:
    comando1
elif exp2:
    comando1
```

```
if  exp1 or exp2:
    comando1
```

Exemplo: maior de 3

- Encontrar a variável de maior valor entre três variáveis inteiras **a**, **b** e **c** com valores distintos.

```
1  if a > b:
2      if a > c:
3          print("a é o maior")
4      else:
5          print("c é o maior")
6  else:
7      if b > c:
8          print("b é o maior")
9      else:
10         print("c é o maior")
```

```
1  if a > b and a > c :
2      print("a é o maior")
3  elif b > c :
4      print("b é o maior")
5  else:
6      print("c é o maior")
```

Álgebra Booleana: Propriedades Básicas

Propriedades Comutativas	A and B = B and A	A or B = B or A
Propriedades Distributivas	A and (B or C) = (A and B) or (A and C)	A or (B and C) = (A or B) and (A or C)
Propriedades Associativas	(A or B) or C = A or (B or C)	(A and B) and C = A and (B and C)
Propriedades Idempotentes	A and A = A	A or A = A
Dupla Negação	not not A = A	
Elementos Absorventes	A or True = True	A and False = False
Elementos Neutros	A or False = A	A and True = A
Leis de De Morgan	not (A or B) = (not A) and (not B)	not (A and B) = (not A) or (not B)

Álgebra Booleana: Exemplo

- Considere a expressão:

  (x < 3) or (y == 0)

- Podemos aplicar a dupla negação sem alterar seu resultado:

  not not ((x < 3) or (y == 0))

- Podemos agora trocar o operador or pelo operador and, aplicando a "Leis de De Morgan":

  not (not (x < 3) and not (y == 0))

- Trocando os operadores relacionais negados por operadores relacionais complementares:

  not (x >= 3 and y != 0)

Dúvidas

Exercicio

- Escreva um programa que determina a data **cronologicamente** maior entre duas datas fornecidas pelo usuário. Cada data deve ser fornecida por três valores inteiros onde o primeiro representa um **dia**, o segundo um **mês** e o terceiro um **ano**.

Exercicio

Sem o uso de operadores lógicos

```python
# Primeira data.
d1 = int(input("Dia: "))
m1 = int(input("Mês: "))
a1 = int(input("Ano: "))
# Segunda data.
d2 = int(input("Dia: "))
m2 = int(input("Mês: "))
a2 = int(input("Ano: "))

if a1 > a2:
    print("Data1 é maior!")
elif a1 == a2:
    if m1 > m2:
        print("Data1 é maior!")
    elif m1 == m2:
        if d1 > d2:
            print("Data1 é maior!")
        elif d1 == d2:
            print("Datas são iguais!")
        else:
            print("Data2 é maior!")
    else:
        print("Data2 é maior!")
else:
    print("Data2 é maior!")
```

Com uso de operadores lógicos

```python
# Primeira data.
d1 = int(input("Dia: "))
m1 = int(input("Mês: "))
a1 = int(input("Ano: "))
# Segunda data.
d2 = int(input("Dia: "))
m2 = int(input("Mês: "))
a2 = int(input("Ano: "))

if a1>a2 or (a1==a2 and m1>m2) or (a1==a2 and m1==m2 and d1>d2):
    print("Data1 é maior!")
elif a1==a2 and m1==m2 and d1==d2:
    print("Datas são iguais!")
else:
    print("Data2 é maior!")
```

Exercicio

- Segundo a propriedade distributiva da álgebra booleana temos:
  - expr1 and (expr2 or expr3) = (expr1 and expr2) or (expr1 and expr3)
- Logo, uma outra solução pode ser obtida colocando a condição a1==a2 em evidência:

```
1  # Primeira data.
2  d1 = int(input("Dia: "))
3  m1 = int(input("Mês: "))
4  a1 = int(input("Ano: "))
5  # Segunda data.
6  d2 = int(input("Dia: "))
7  m2 = int(input("Mês: "))
8  a2 = int(input("Ano: "))
9
10 if a1>a2 or (a1==a2 and (m1>m2 or (m1==m2 and d1>d2))):
11      print("Data1 é maior!")
12 elif a1==a2 and m1==m2 and d1==d2:
13      print("Datas são iguais!")
14 else:
15      print("Data2 é maior!")
```

Exercício

- Peça uma palavra ao usuário e verifique se ela tem mais de 5 letras **ou** se contém a letra 'a'.

```
palavra = input("Digite uma palavra: ")
print(len(palavra) > 5 or 'a' in palavra)
```

Exercicio

- Dados um número inteiro n>0 e as notas de n alunos, determinar quantos ficaram de recuperação. Um aluno está de recuperação se sua nota final for maior ou igual a 3 e menor do que 5.

Exercicio

- Dados um número inteiro n>0 e as notas de n alunos, determinar quantos ficaram de recuperação. Um aluno está de recuperação se sua nota final for maior ou igual a 3 e menor do que 5.

```
1  n = int(input("Digite n: "))
2  rec = 0
3  i = 1
4  while i <= n:
5      nota = float(input("Digite uma nota: "))
6      if 3.0 <= nota and nota < 5.0:
7          rec = rec + 1
8      i = i + 1
9
10 print(rec,"alunos ficaram de recuperação")
```

Exercicio

- Uma segunda solução sem usar operadores lógicos:

```
1  n = int(input("Digite n: "))
2  rec = 0
3  i = 1
4  while i <= n:
5      nota = float(input("Digite uma nota: "))
6      if nota >= 3.0:
7          if nota < 5.0:
8              rec = rec + 1
9      i = i + 1
10
11 print(rec,"alunos ficaram de recuperação")
```

Exercicio

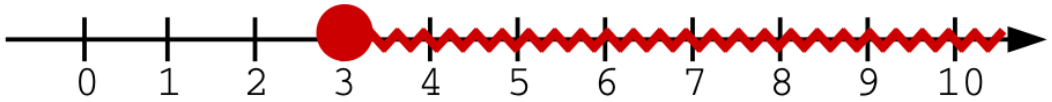
- O Python ainda permite uma terceira solução, na qual o intervalo numérico é especificado de modo mais similar a nossa linguagem, usando operadores de comparação de forma encadeada:

```
1  n = int(input("Digite n: "))
2  rec = 0
3  i = 1
4  while i <= n:
5      nota = float(input("Digite uma nota: "))
6      if 3.0 <= nota < 5.0:
7          rec = rec + 1
8      i = i + 1
9
10 print(rec,"alunos ficaram de recuperação")
```

And

- Do ponto de vista gráfico, usando a notação de conjuntos de intervalos na reta dos reais, temos:

```
3.0 <= nota
```




```
nota < 5.0
```


```
3.0 <= nota and nota < 5.0
```

Observe que graficamente o operador lógico "and" possui o efeito de intersecção dos conjuntos.

Exercicio

- Dados números inteiros n, i e j, todos maiores do que zero, imprimir em ordem crescente os n primeiros naturais que são múltiplos de i ou de j e ou de ambos. Por exemplo, para n=6, i=2 e j=3 a saída deverá ser:

Exercicio

- Testa os números 0, 1, 2, ... verificando e imprimindo quais são múltiplos de i ou j, até que n múltiplos sejam impressos.

```
# dados de entrada
print("Cálculo dos n primeiros múltiplos de i ou de j")
n = int(input("Digite n: "))
i = int(input("Digite i: "))
j = int(input("Digite j: "))

cont = 0 #conta quantos múltiplos foram impressos.
cm = 0 #candidato a múltiplo.
while cont < n:
    if cm%i == 0 or cm%j == 0:
        print(cm)
        cont += 1
    cm += 1
```

Exercicio

- Mais elaborada. Faz menos iterações que a anterior. A cada iteração imprime um múltiplo de i ou j.

```
# dados de entrada
print("Cálculo dos n primeiros múltiplos de i ou de j")
n = int(input("Digite n: "))
i = int(input("Digite i: "))
j = int(input("Digite j: "))

multi = 0 # múltiplos de i
multj = 0 # múltiplos de j
cont = 0 # conta quantos múltiplo foram impressos

while cont < n:
    if multi < multj:
        print(multi)
        multi += i
    elif multj < multi:
        print(multj)
        multj += j
    else: # multi == multj
        print(multj)
        multi += i
        multj += j
```