Processamento da Informação
– Teoria –

Desvio Condicional

Semana 03

Prof. Jesús P. Mena-Chalco

08/05/2013

Operador módulo

O **operador módulo** trabalha com inteiros e produz o resto quando o primeiro operando é dividido pelo segundo.

Em Python, o operador módulo é um sinal de porcentagem (%). A sintaxe é a mesma que para os outros operadores

```
>>> divisao = 7 / 3
>>> print divisao
>>> 2

>>> resto = 7 % 3
>>> print resto
>>> 1
```

Operador módulo

O **operador módulo** trabalha com inteiros e produz o resto quando o primeiro operando é dividido pelo segundo.

Em Python, o operador módulo é um sinal de porcentagem (%). A sintaxe é a mesma que para os outros operadores

```
>>> divisao = 7 / 3
>>> print divisao
>>> 2

>>> resto = 7 % 3
>>> print resto
>>> 1
```

7 dividido por 3 é 2, com 1 sobrando (resto)

Operador módulo

```
>>> divisao = 13 / 2
>>> print divisao
>>> 6

>>> resto = 13 % 2
>>> print resto
>>> 1
```

13 dividido por 2 é 6, com 1 sobrando (resto)

Operador módulo

```
>>> divisao = 20 / 1
>>> print divisao
>>> 20

>>> resto = 20 % 1
>>> print resto
>>> 0
```

20 dividido por 1 é 20, com **0** sobrando (resto)

Operador módulo

```
>>> divisao = 2 / 13
>>> print divisao
>>> 0

>>> resto = 2 % 13
>>> print resto
>>> 2
```

2 dividido por 13 é 0, com 2 sobrando (resto)

Operador módulo

O operador módulo acaba sendo surpreendentemente útil.

Por exemplo, você pode verificar se um número é divisível por outro, se x%y é zero, então x é divisível por y

```
>>> 24 % 1    → 0
>>> 24 % 2    → 0
>>> 24 % 3    → 0
>>> 24 % 4    → 0
>>> 24 % 5    → 4
>>> 24 % 6    → 0
>>> 24 % 7    → 3
>>> 24 % 8    → 0
>>> 24 % 9    → 6
```

Operador módulo

Este operador pode ser utilizado para extrair o(s) digito(s) mais à direita de um número.

Por exemplo:

```
>>> 12345 / 10     → 1234
>>> 12345%10       → 5
```

Mantém o digito mais à direita

```
>>> 12345 / 100    → 123
>>> 12345%100      → 45
```

Mantém os 2 digitos mais à direita

Expressões booleanas

Uma expressão booleana é uma expressão que é ou **Verdadeira** ou **Falsa**.

Os seguintes exemplos usam o operador **“==”**, utilizado para comparar dois operandos e produzir **True** se eles forem iguais ou **False** em caso contrário.

```
>>> 5 == 5
True

>>> 5 == 6
False
```

Expressões booleanas

Uma expressão booleana é uma expressão que é ou **Verdadeira** ou **Falsa**.

Os seguintes exemplos usam o operador **"=="**, utilizado para comparar dois operandos e produzir **True** se eles forem iguais ou **False** em caso contrário.

```
>>> 5 == 5
True
```


```
>>> 5 == 6
False
```

Expressões booleanas

```
>>> w = 40
>>> p = 60

>>> w == 5*8
True

>>> w == 5*8+1
False

>>> w+p == 100
True
```

Expressões booleanas

```
>>> w = 40
>>> p = 60

>>> w == 5*8
True

>>> w == 5*8+1
False

>>> w+p == 100
True
```

```
>>> w+p = 100
```

Erro comum

Expressões booleanas

True e False são valores especiais que pertencem ao tipo de dado **bool** (eles não são *strings*).

```
>>> type(True)
<type 'bool'>
```

```
>>> type(False)
<type 'bool'>
```

Expressões booleanas

O operador "==" é um dos operadores relacionais, os outros são:

```
x !=  y       # x não é igual a y
x  >  y       # x é maior que y
x  <  y       # x é menor que y
x >= y        # x é maior ou igual a y
x <= y        # x é menor ou iqual a y
```

Expressões booleanas

O operador "==" é um dos operadores relacionais, os outros são:

```
x !=  y       # x não é igual a y
x  >  y       # x é maior que y
x  <  y       # x é menor que y
x >= y        # x é maior ou igual a y
x <= y        # x é menor ou iqual a y
```

Um erro comum é usar "=" no lugar de "==".

Expressões booleanas

O operador "==" é um dos operadores relacionais, os outros são:

```
x !=  y       # x não é igual a y
x  >  y       # x é maior que y
x  <  y       # x é menor que y
x >=  y       # x é maior ou igual a y
x <=  y       # x é menor ou iqual a y
```

Um erro comum é usar "=" no lugar de "==".

Operador de atribuição

Operador relacional

Expressões booleanas

O operador "==" é um dos operadores relacionais, os outros são:

```
x !=  y      # x não é igual a y
x  >  y      # x é maior que y
x  <  y      # x é menor que y
x >=  y      # x é maior ou igual a y
x <=  y      # x é menor ou iqual a y
```

Um erro comum é usar "=" no lugar de "==".

Não existem os operadores **=<** ou **=>**.

Operadores lógicos

Existem 3 operadores lógicos: **and**, **or**, e **not**.

A semântica (significado) destes operadores é similar ao seu significado em Inglês/Português.

Por exemplo a expressão: `x > 0 and x < 10` é verdadeira somente se `x` é maior a zero **e** menor do que dez.

Operadores lógicos

```
n%2==0 or n%3==0
```

Operadores lógicos

```
n%2==0 or n%3==0
```

A expressão é verdadeira se uma das condições for verdadeira, isto é, se `n` for divisível por `2` **ou** `3`.

Operadores lógicos

Finalmente, o operador **not** nega uma expressão booleana, assim

```
not (x > y)
```

é verdadeira se **x>y** for falso

isto é, se **x** é menor ou igual a **y**.

Operadores lógicos

A rigor, os operandos de operadores lógicos deveriam ser expressões booleanas, mas Python não é muito rigoroso.

Qualquer número diferente de zero é interpretado como "True", assim:

```
>>> 17 and True
True
```

Essa flexibilidade pode ser útil mas existem algumas sutilezas que isso poderia ser confuso. Você pode querer evitá-lo (a menos que você saiba o que está fazendo)

Execução condicional

Para escrever programas úteis, quase sempre precisamos da possibilidade de verificar condições e mudar o comportamento do programa.

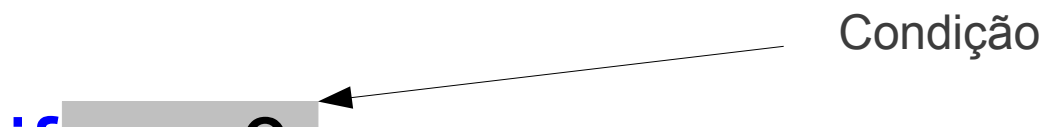
Instruções condicionais nos dão essa habilidade. A forma mais simples é o **if**:

```
if x > 0 :
    print 'x é positivo'
```

Execução condicional

Para escrever programas úteis, quase sempre precisamos da possibilidade de verificar condições e mudar o comportamento do programa.

Instruções condicionais nos dão essa habilidade. A forma mais simples é o **if**:

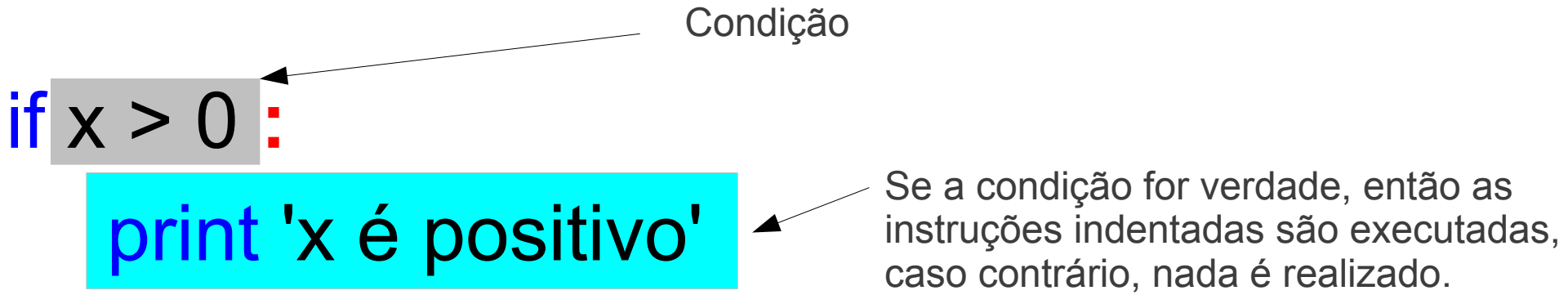



```
if x > 0 :
    print 'x é positivo'
```

Execução condicional

Para escrever programas úteis, quase sempre precisamos da possibilidade de verificar condições e mudar o comportamento do programa.

Instruções condicionais nos dão essa habilidade. A forma mais simples é o **if**:



Execução condicional

```
if x > 0 :
    print 'x é positivo'
```

```
if eh_primo(x)  :
    print 'x é primo'
```

if tem a mesma estrutura que as definições de função: um cabeçalho seguido por um corpo indentado.

Instruções como esta são chamadas de **declarações compostas**.

Não há limites para o número de instruções dentro do corpo. Entretanto, deve existir, pelo menos UM.

Execução alternativa

A segunda forma da instrução **if** é a execução alternativa, na qual existem duas possibilidades e a condição determina qual delas será executada.

A sintaxe parece assim

```
if x%2 == 0:
    print 'x é par'
else:
    print 'x é ímpar'
```

Execução alternativa

A segunda forma da instrução **if** é a execução alternativa, na qual existem duas possibilidades e a condição determina qual delas será executada.

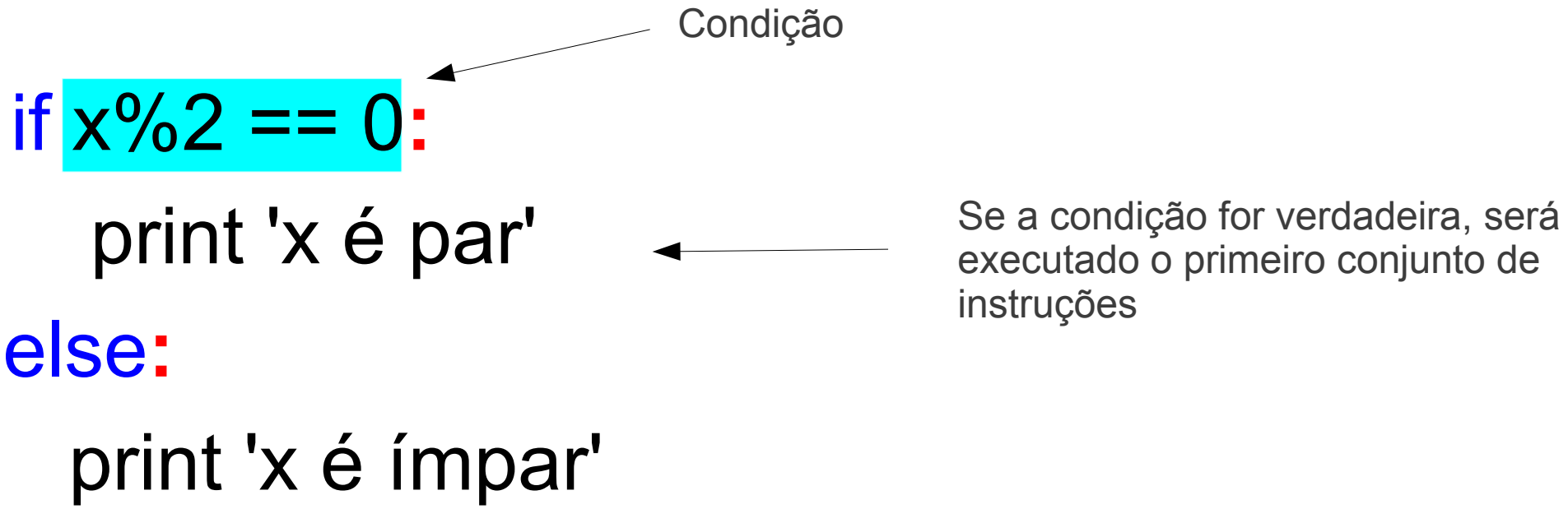
A sintaxe parece assim


```
if x%2 == 0:
    print 'x é par'
else:
    print 'x é ímpar'
```

Execução alternativa

A segunda forma da instrução **if** é a execução alternativa, na qual existem duas possibilidades e a condição determina qual delas será executada.

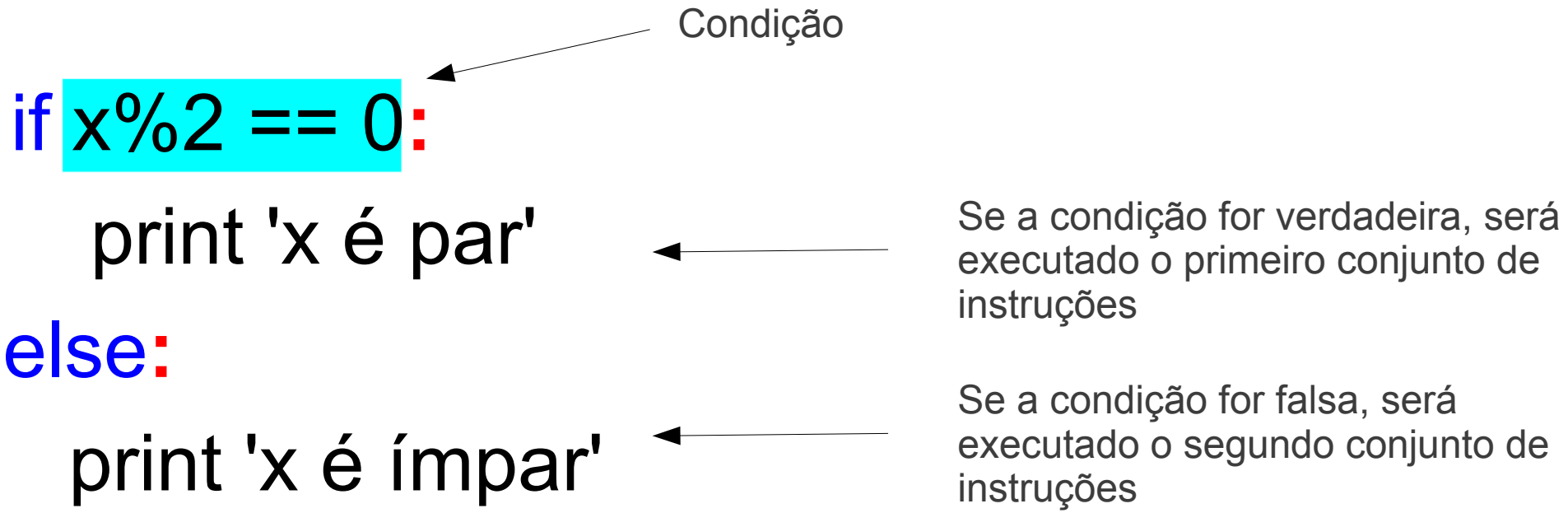
A sintaxe parece assim



Execução alternativa

A segunda forma da instrução **if** é a execução alternativa, na qual existem duas possibilidades e a condição determina qual delas será executada.

A sintaxe parece assim



Execução alternativa

```
if x%2 == 0:
    print 'x é par'
else:
    print 'x é ímpar'
```

Desde que a condição deve ser verdadeira ou falsa, exatamente uma das alternativas será executada.

As alternativas são chamadas ramos, porque eles são ramos no fluxo de execução.

Execução alternativa

```
if x%2 == 0:
    print 'x é par'
else:
    print 'x é ímpar'
```

Desvios condicionais encadeados

Às vezes, há mais de duas possibilidades e precisamos de mais do que dois ramos.

Uma maneira de expressar uma computação como essa é uma condicional encadeada:

```
if x < y:
    print 'x é menor que y'
elif x > y:
    print 'x é maior que y'
else:
    print 'x e y são iguais'
```

Desvios condicionais encadeados

Às vezes, há mais de duas possibilidades e precisamos de mais do que dois ramos.

Uma maneira de expressar uma computação como essa é uma condicional encadeada:

```
if x < y:
    print 'x é menor que y'
elif x > y:
    print 'x é maior que y'
else:
    print 'x e y são iguais'
```

elif é a abreviação de **else if**

Desvios condicionais encadeados

Às vezes, há mais de duas possibilidades e precisamos de mais do que dois ramos.

Uma maneira de expressar uma computação como essa é uma condicional encadeada:

```
if x < y:
    print 'x é menor que y'
elif x > y:
    print 'x é maior que y'
else:
    print 'x e y são iguais'
```

elif é a abreviação de **else if**

Apenas um ramo será executado

Desvios condicionais encadeados

Não há limite no número de instruções **elif**.

```
if x == 'a':
    print 'x contém a letra a'
elif x == 'b':
    print 'x contém a letra b'
elif x == 'c':
    print 'x contém a letra c'
else:
    print 'x não contem a, b ou c'
```

Desvios condicionais encadeados

Não há limite no número de instruções **elif**.

```
if x == 'a':
    print 'x contém a letra a'
elif x == 'b':
    print 'x contém a letra b'
elif x == 'c':
    print 'x contém a letra c'
else:
    print 'x não contem a, b ou c'
```

A instrução **else** não é obrigatória nesse caso

Desvios condicionais encadeados

```
if x > 0:
    print 'x é positivo'
elif x%2==0:
    print 'x é par'
elif x%3==0:
    print 'x é múltiplo de 3'
```

Considere x=12:

- 12 >0
- 12%2=0
- 12%3=0

Qual seria o resultado?

Desvios condicionais encadeados

```
if x > 0:
    print 'x é positivo'
elif x%2==0:
    print 'x é par'
elif x%3==0:
    print 'x é múltiplo de 3'
```

Considere x=12:

- 12 >0
- 12%2=0
- 12%3=0

Qual seria o resultado?

Desvios condicionais encadeados

```
if x > 0:
    print 'x é positivo'
elif x%2==0:
    print 'x é par'
elif x%3==0:
    print 'x é múltiplo de 3'
```

Considere x=12:

- 12 >0
- 12%2=0
- 12%3=0

Qual seria o resultado?

Se a primeira condição for Falsa, então a segunda é verificada. Se a segunda for Falsa, então a terceira condição é executada.

Apenas a primeira condição verdadeira é executada.

Desvios condicionais encadeados

```
if x > 0:
    print 'x é positivo'
elif x%2==0:
    print 'x é par'
elif x%3==0:
    print 'x é múltiplo de 3'
```

Considere x=12:

- 12 >0
- 12%2=0
- 12%3=0

Qual seria o resultado?

Resposta: x é positivo

Desvios condicionais aninhados

Uma condicional também pode ser aninhada dentro de outra. Poderíamos ter escrito o exemplo de tricotomia como esta:

```
if x == y:
    print 'x e y são iguais'
else:
    if x < y:
        print 'x é menor que y'
    else:
        print 'x é maior que y'
```

Desvios condicionais aninhados

Uma condicional também pode ser aninhada dentro de outra. Poderíamos ter escrito o exemplo de tricotomia como esta:

```
if x == y:
    print 'x e y são iguais'
else:
    if x < y:
        print 'x é menor que y'
    else:
        print 'x é maior que y'
```

Verifique a indentação dos blocos de instruções

Desvios condicionais aninhados

Uma condicional também pode ser aninhada dentro de outra. Poderíamos ter escrito o exemplo de tricotomia como esta:

```
if x == y:
    print 'x e y são iguais'
else:
    if x < y:
        print 'x é menor que y'
    else:
        print 'x é maior que y'
```

Verifique a indentação dos blocos de instruções

Desvios condicionais aninhados

Os **operadores lógicos** muitas vezes fornecem uma maneira de simplificar instruções condicionais aninhadas.

```
if 0<x:
    if x<10:
        print 'x é um número positivo de 1 digito'
```

A mensagem é mostrada quando as duas condições forem válidas. Pode ser reescrita usando UMA única condicional:

```
if 0<x and x<10:
        print 'x é um número positivo de 1 digito'
```

Atividade em aula

Questão 1: Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
if x>=8.5:
    print "Conceito A"
if x>=7.5:
    print "Conceito B"
if x>=5.5:
    print "Conceito C"
if x>=5:
    print "Conceito D"
```

Atividade em aula

Questão 2: Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
if x>=8.5:
    print "Conceito A"
elif x>=7.5:
    print "Conceito B"
elif x>=5.5:
    print "Conceito C"
elif x>=5:
    print "Conceito D"
```

Atividade em aula

Questão 3: Indique a mensagem que apresentará a execução das seguintes instruções:

```
aluno = "Joao Carlo"
disciplina = "PI"
if aluno=="Joao Carlos" and disciplina=="PI":
    print "Conceito A"
else:
    print "Aluno não cadastrado"
```

Atividade em aula

Questão 4: Indique a mensagem que apresentará a execução das seguintes instruções:

```
aluno = "Joao Carlo"
disciplina = "PI"
if aluno=="Joao Carlos" or disciplina=="PI":
    print "Conceito A"
else:
    print "Aluno não cadastrado"
```

Atividade em aula

Questão 5: Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
y=5
z=13
if x>=1 and x<=31:
    if y>0 and y<13:
        if x+y!=z:
            print "A data de hoje é 8/5/13"
        else:
            print "A data de hoje não é 8/5/13"
```

Atividade em aula

Avaliação

Cada questão vale 2 pontos. Considere respostas exatas.

Questão 1: 2 pontos
Questão 2: 2 pontos
Questão 3: 2 pontos
Questão 4: 2 pontos
Questão 5: 2 pontos

Atividade em aula

Questão 1: Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
if x>=8.5:
    print "Conceito A"
if x>=7.5:
    print "Conceito B"
if x>=5.5:
    print "Conceito C"
if x>=5:
    print "Conceito D"
```

Resposta:

Conceito B
Conceito C
Conceito D

Atividade em aula

Questão 2: Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
if x>=8.5:
    print "Conceito A"
elif x>=7.5:
    print "Conceito B"
elif x>=5.5:
    print "Conceito C"
elif x>=5:
    print "Conceito D"
```

Resposta:

Conceito B

Atividade em aula

Questão 3: Indique a mensagem que apresentará a execução das seguintes instruções:

```
aluno = "Joao Carlo"
disciplina = "PI"
if aluno=="Joao Carlos" and disciplina=="PI":
    print "Conceito A"
else:
    print "Aluno não cadastrado"
```

Resposta:
Aluno não cadastrado

Atividade em aula

Questão 4: Indique a mensagem que apresentará a execução das seguintes instruções:

```
aluno = "Joao Carlo"
disciplina = "PI"
if aluno=="Joao Carlos" or disciplina=="PI":
    print "Conceito A"
else:
    print "Aluno não cadastrado"
```

Resposta:
Conceito A

Atividade em aula

Questão 5: Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
y=5
z=13
if x>=1 and x<=31:
    if y>0 and y<13:
        if x+y!=z:
            print "A data de hoje é 8/5/13"
        else:
            print "A data de hoje não é 8/5/13"
```

Resposta:
A data de hoje não é 8/5/13