Comandos de repetição

Prof. Marcio Delamaro

SSC0301

Método da bisseção

```
import sys

f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))

#iteração 1
c = (a+b)/2
if abs(( b - a ) / 2) < erro:
    print('Achou raiz ', c, ' com erro ', (b-a)/2)
    sys.exit();

if f(a) * f(c) < 0:
    b = c
else:
    a = c
```

Método da bisseção

```
import sys

f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))

#iteração 1
c = (a+b)/2
if abs(( b - a ) / 2) < erro:
    print('Achou raiz ', c, ' com erro ', (b-a)/2)
    sys.exit();

if f(a) * f(c) < 0:
    b = c
else:
    a = c
```

Inconveniente pois o código fica enorme, feio, deselegante

Método da bisseção

```
import sys

f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))

#iteração 1
c = (a+b)/2
if abs(( b - a ) / 2) < erro:
    print('Achou raiz ', c, ' com erro ', (b-a)/2)
    sys.exit();

if f(a) * f(c) < 0:
    b = c
else:
    a = c
```

Inconveniente pois o código fica enorme, feio, deselegante

Inflexível, pois sempre temos o mesmo número máximo de iterações.

Comandos de repetição

- Permitem controlar quantas vezes um comando (ou vários) é executado
- comando `while`

Comandos de repetição

- Permitem controlar quantas vezes um comando (ou vários) é executado
- comando `while`

```
while < expressão booleana > :
    comando executado se expressão for verdadeira
    comando executado se expressão for verdadeira
    comando executado se expressão for verdadeira
```

Exemplo while

```
i = 1

while i < 10:
    print('O valor de i é: ', i)
    i = i + 1

print('O valor final de i é: ', i);
```

Exemplo while

```
i = 1

while i < 10:
    print('O valor de i é: ', i)
    i = i + 1

print('O valor final de i é: ', i);
```

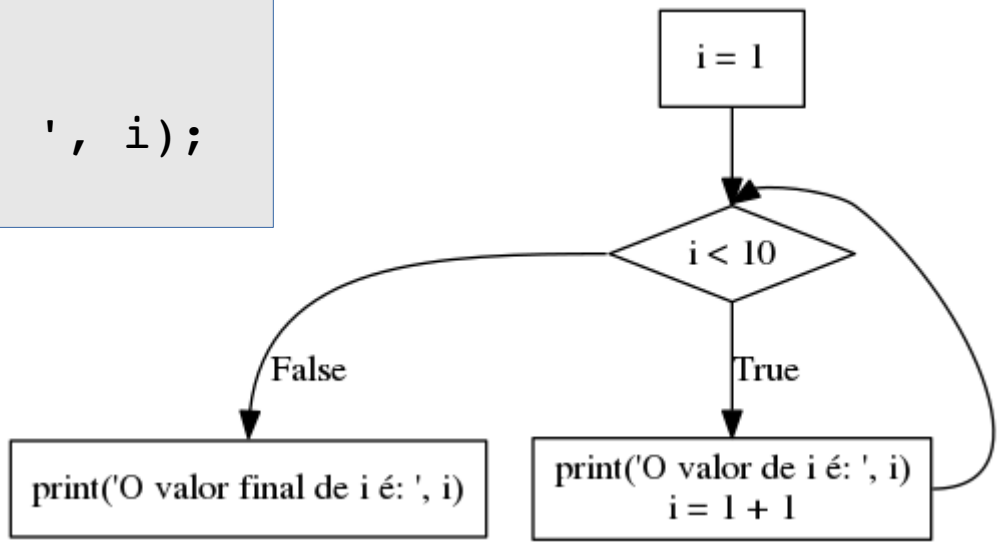
```
O valor de i é: 1
O valor de i é: 2
O valor de i é: 3
O valor de i é: 4
O valor de i é: 5
O valor de i é: 6
O valor de i é: 7
O valor de i é: 8
O valor de i é: 9
O valor final de i é: 10
```

Exemplo while

```
i = 1

while i < 10:
    print('O valor de i é: ', i)
    i = i + 1

print('O valor final de i é: ', i);
```



Praticando

- O que acontece nos programas abaixo?

```
i = 1
while True:
    print('O valor de i é: ', i)
    i = i + 1

print('O valor final de i é: ', i);
```

```
i = 1
while False:
    print('O valor de i é: ', i)
    i = i + 1

print('O valor final de i é: ', i);
```

ICMC USP SÃO CARLOS SSC

Praticando

- Escreva um programa que leia um número inteiro positivo e mostre como resutado a soma de todos os números de 0 até o número lido.
- Escreva um programa que leia um número inteiro positivo e mostre todos os números múltiplos de 9 entre 0 e o número lido.

Voltando à bisseção

- Agora que sabemos usar o while, vamos usá-lo para implementar o método da bisseção
- Cada iteração é repetida enquanto não tivermos o número desejado de iterações e o erro for maior do que a tolerância

Voltando à bisseção

```
import sys

f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))

#iteração 1
c = (a+b)/2
if abs(( b - a ) / 2) < erro:
    print('Achou raiz ', c, ' com erro ', (b-a)/2)
    sys.exit();

if f(a) * f(c) < 0:
    b = c
else:
    a = c
```

Voltando à bisseção

```
import sys

f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))
iteracoes = int(input('Número máximo de iterações: '))

#iteração 1
c = (a+b)/2
if abs(( b - a ) / 2) < erro:
    print('Achou raiz ', c, ' com erro ', (b-a)/2)
    sys.exit();

if f(a) * f(c) < 0:
    b = c
else:
    a = c
```

Voltando à bisseção

```
import sys

f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))
iteracoes = int(input('Número máximo de iterações: '))

i = 1
while i <= iteracoes
c = (a+b)/2
if abs(( b - a ) / 2) < erro:
    print('Achou raiz ', c, ' com erro ', (b-a)/2)
    sys.exit();

if f(a) * f(c) < 0:
    b = c
else:
    a = c
```

Voltando à bisseção

```
import sys

f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))
iteracoes = int(input('Número máximo de iterações: '))

i = 1
while i <= iteracoes
    c = (a+b)/2
    if abs(( b - a ) / 2) < erro:
       print('Achou raiz ', c, ' com erro ', (b-a)/2)
       sys.exit();

    if f(a) * f(c) < 0:
       b = c
    else:
       a = c
    i = i + 1
```

Voltando à bisseção

```
f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))
iteracoes = int(input('Número máximo de iterações: '))

i = 1
while i <= iteracoes
    c = (a+b)/2
    if abs(( b - a ) / 2) < erro:
       print('Achou raiz ', c, ' com erro ', (b-a)/2)
       sys.exit();

    if f(a) * f(c) < 0:
       b = c
    else:
       a = c
    i = i + 1
print('Valor calculado ', c, ' com erro ', (b-a)/2)
```

Ou ainda...

```
f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))
iteracoes = int(input('Número máximo de iterações: '))

i = 1
c = (a+b)/2

while i <= iteracoes and abs(( b - a ) / 2) >= erro :
    if f(a) * f(c) < 0:
        b = c
    else:
        a = c
    i = i + 1
    c = (a+b)/2

print('Valor calculado ', c, ' com erro ', (b-a)/2)
```

Ou ainda...

```
f = lambda x: x ** 3 - x ** 2 - 13 * x + 8
a = float(input('Forneça o valor inicial de a: '))
b = float(input('Forneça o valor inicial de b: '))
erro = float(input('Qual o valor da tolerância? '))
iteracoes = int(input('Número máximo de iterações: '))

i = 1
c = (a+b)/2

while i <= iteracoes and abs(( b - a ) / 2) >= erro :
    if f(a) * f(c) < 0:
        b = c
    else:
        a = c
    i = i + 1
    c = (a+b)/2

print('Valor calculado ', c, ' com erro ', (b-a)/2)
```

Vai pensando...

- E o método de Newton-Raphson? Como implementar? (x3 – x2 – 13x + 8)
- Algoritmo
  - definir quem é a função f
  - definir sua 1a. derivada f'
  - definir o chute inicial x_0
  - calcular $x_{i+1} = x_i – (f(x_i)/f'(x_i))$ enquanto $x_{i+1} – x_i$ for maior do que o erro desejado ou até que se alcance a um número máximo de iterações

Comando for

- Variável de controle
- Objeto com várias “partes”
- A cada iteração uma dessas partes é atribuída à variável de controle
- Os comandos do for são executados

Exemplo: string

- Um string é formado por diversos caracteres
- Podemos “percorrer” cada elemento do string

Exemplo: string

- Um string é formado por diversos caracteres
- Podemos “percorrer” cada elemento do string

```
s = 'Python'
for c in s:
    print(c)
```

A cada iteração a variável c recebe um caractere do string em s

Exemplo: string

- Um string é formado por diversos caracteres
- Podemos “percorrer” cada elemento do string

```
s = 'Python'
for c in s:
    print(c)
```

A cada iteração a variável c recebe um caractere do string em s

```
P
y
t
h
o
n
```

Range

- É um tipo de objeto que serve para criar um contador
- range(n) – vai gerar um contador de 0 até n-1

```
for j in range(5):
    print(j)
```

Range

- É um tipo de objeto que serve para criar um contador
- range(n) – vai gerar um contador de 0 até n-1

```
for j in range(5):
    print(j)
```

```
s = 0
for j in range(5):
    s += j
print(j)
```

Range

- range(p,n) – gera um contador de p até n-1
- range(p,n,s) – gera um contador de p até n-1, com incremento de s

```
for j in range(-5, 10, 3):
    print(j)
```

```
for j in range(5, -10, -3):
    print(j)
```

Exemplos

- Verificar se um número inteiro é primo.

Exemplos

- Verificar se um número inteiro é primo.

```
n = int(input('Número a verificar: '))

cont = 0

for i in range(2,n):
    if n % i == 0:
        cont += 1

if cont > 0 :
    print('Não é primo')
else:
    print('É primo')
```

Exemplos

- Calcular a média de m notas de n alunos

Exemplos

- Calcular a média de m notas de n alunos

```
n = int(input('Qtos alunos? '))
m = int(input('Qtas notas? '))
```

Exemplos

- Calcular a média de m notas de n alunos

```
n = int(input('Qtos alunos? '))
m = int(input('Qtas notas? '))

for i in range(1,n+1):
    print('Entre com as notas do aluno {}'.format(i))
    soma = 0.0
    for j in range(1,m+1):
        nota = float(input('Nota {}: '.format(j)))
        soma += nota
    print('Média é {:.2f}'.format(soma/m))
```

Comando break

- Serve para sair de um comando de repetição
- Qdo o break é executado, a eecução vai direto para o próximo comando depois do `while` ou `for`

Comando break

- Por exemplo, vamos supor que temos dois números inteiros, armazenados nas variáveis a e b e a < b. Queremos achar o menor valor entre esses dois que seja um divisor de b. Para isso, vamos incrementando o valor de a até acharmos um divisor ou até que seu valor chegue em b.

Comando break

```
a = int(input('Digite o valor de a: '))
b = int(input('Digite o valor de b: '))
while a < b:
    if b % a == 0: # verifica se a divide b
        break
    a += 1
print('O valor do divisor é: ', a)
```

Comando continue

- O comando `continue` faz com que a a execução do laço seja interrompida mas não abandonada.
- A execução volta para o início do comando de repetição
- A condição vai ser testada novamente e, se for verdadeira, uma nova iteração do comando acontece.
- Se for falsa, o comando de repetição termina normalmente.
- Em uma execução de um comando de repetição o `continue`, ao contrário do `break`, pode ser executado várias vezes.

Comando continue

- Queremos achar o menor divisor de b, entre a e b
- Mas ele não pode ser múltiplo de 11
- Então, cada vez que um múltiplo de 11 aparecer nosso programa vai fazer a execução voltar ao comando de repetição
- Ou seja, não vamos verificar se ele é ou não divisor de b

Comando continue

```
a = int(input('Digite o valor de a: '))
b = int(input('Digite o valor de b: '))

for a in range(a,b+1):
    if a % 11 == 0:
        continue
    if b % a == 0:
        Break

print('O valor do divisor é: ', a)
```