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 {} com erro {}'.format(c, abs(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 {} com erro {}'.format(c, abs(b-a)/2))
    sys.exit();

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

Inconveniente pois o código fica enorme, feio, deselegante

ICMC USP SÃO CARLOS SSC

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 {} com erro {}'.format(c, abs(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);
```

http://www.pythontutor.com/visualize.html#mode=edit

ICMC USP SÃO CARLOS SSC

Exemplo while

```
i = 1

while i < 5:
    print('O valor de i é: ', i)
    i = i + 1

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

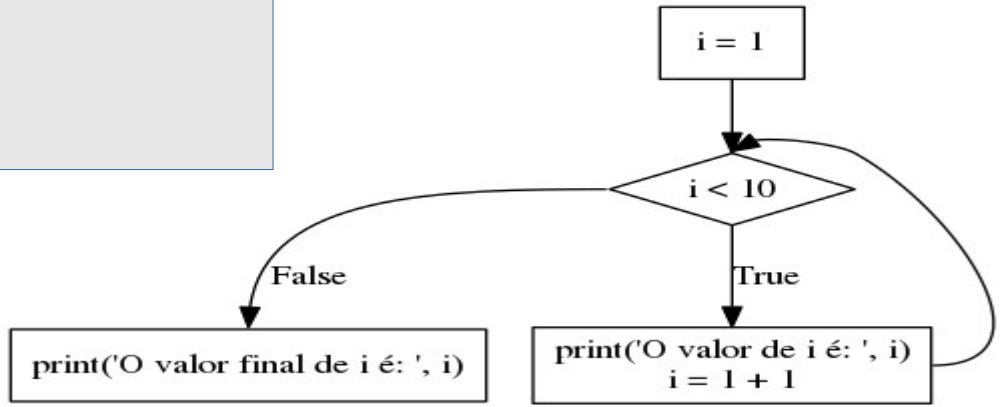
```
O valor de i é: 1
O valor de i é: 2
O valor de i é: 3
O valor de i é: 4
O valor final de i é: 5
```

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

- 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.

Praticando

```
n = int(input('Digite um número inteiro:'))
soma = 0
i = 1
while i < n:
    soma = soma + i
    i = i + 1
print('O valor da soma: {}'.format(soma))
```

Praticando

```
n = int(input('Digite um número inteiro:'))
i = 0
while i < n:
    if i % 9 == 0:
        print('{} é múltiplo de 9'.format(i))
    i = i + 1
```

ICMC USP SÃO CARLOS SSC

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 {} com erro {}'.format(c, abs(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 {} com erro {}'.format(c, abs(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 = 0
while i <= iteracoes:
c = (a+b)/2
if abs(( b - a ) / 2) < erro:
    print('Achou raiz {} com erro {}'.format(c, abs(b-a)/2))
    sys.exit()

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

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

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 = 0
while i <= iteracoes:
    c = (a+b)/2
    if abs(( b - a ) / 2) < erro:
        print('Achou raiz {} com erro {}'.format(c, abs(b-a)/2))
        sys.exit()

    if f(a) * f(c) < 0:
        b = c
    else:
        a = c
    i = i + 1
print('Valor calculado {} com erro {} '.format(c,(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 = 0
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 {} com erro {}.'.format(c,(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

ICMC USP SÃO CARLOS SSC

Comando break

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

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)
```

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 = 0
while i <= iteracoes:
    c = (a+b)/2
    if abs(( b - a ) / 2) < erro:
       break

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

ICMC USP SÃO CARLOS SSC

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.

ICMC USP SÃO CARLOS SSC

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: '))

while a <= b:
    if a % 11 == 0:
        a = a + 1
        continue
    if b % a == 0:
        break
     a = a + 1

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