Processamento da Informação
– Teoria –

Laços aninhados

Semana 03

Prof. Jesús P. Mena-Chalco

10/05/2013

Uma possível solução da lista 02...

Questão 1: Crie uma função que permita somar apenas os números impares da sequência de inteiros contida no intervalos [x,y], para x<y.

```
def soma_impares(x,y):
    soma = 0
    for i in range(x,y+1):
        if i%2==1:
            soma = soma+i
    return soma
```

Uma possível solução da lista 02...

Questão 2: Dado um inteiro positivo n, crie uma função para calcular a seguinte soma:

$$\frac{1}{n} + \frac{2}{n-1} + \frac{3}{n-2} + \ldots + \frac{n}{1}$$

```
def soma(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma + float(i)/(n-(i-1))
    return soma
```

Uma possível solução da lista 02...

Questão 3: Faça uma função arctan que recebe o número real $x \in [0,1]$ e devolve uma aproximação do arco tangente de x (em radianos) através da série:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

```
def arctan(x):
    x = float(x)
    soma = 0
    sinal = -1
    for i in range(1,100+1):
        coef  = 2*i-1
        sinal = sinal*-1
        soma  = soma + sinal * (x**coef)/coef
    return soma
```

Processamento da Informação
– Teoria –

Laços aninhados

Semana 03

Prof. Jesús P. Mena-Chalco

10/05/2013

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n
        n = n-1
    print 'Boom!'
```

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n
        n = n-1
    print 'Boom!'
```

Evaluar a condição, produzindo True ou False

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n
        n = n-1
    print 'Boom!'
```

Se a condição for falsa, sai do laço while e continua a execução da seguinte instrução

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n
        n = n-1
    print 'Boom!'
```

Se a condição for verdadeira, executa as instruções do bloco do laço (cor cinza), e volta ao início do laço.

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n
        n = n-1
    print 'Boom!'
```

Se a condição for verdadeira, executa as instruções do bloco do laço (cor cinza), e volta ao início do laço.

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n
        n = n-1
    print 'Boom!'
```

```
>>> contagem_regressiva(3)
3
2
1
Boom
```

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n

    print 'Boom!'
```

Laços (while)

Contagem regressiva, considerando **n** como parâmetro de entrada:

```
def contagem_regressiva(n):
    while n > 0:
        print n

    print 'Boom!'
```

```
>>> contagem_regressiva(3)
3
3
3
3
3
3
3
.
.
```

Laço sem fim (infinito)

Laços (while)

As vezes nem sempre é fácil determinar se o laço irá terminar. Um exemplo:

```
def sequencia(n):
    while n != 1:
        print n
        if n%2 == 0:
            n = n/2
        else:
        n = n*3+1
    print 'finalizou'
```

Laços (while)

As vezes nem sempre é fácil determinar se o laço irá terminar. Um exemplo:

```
def sequencia(n):
    while n != 1:
        print n
        if n%2 == 0:
            n = n/2
        else:
        n = n*3+1
    print 'finalizou'
```

```
>>> sequencia(3)
 3
10
5
16
8
4
2
finalizou
```

Laços (while)

Se n incrementa ou decrementa de valor, então não é fácil provar que atingirá o valor de 1, ou que o laço termine.

Vimos que para n=3, o laço termina. Para n = potências de 2 também terminará.

Até agora ninguém conseguiu provar de forma completa se o laço terminará: Problema 3n+1 (Conjectura Collatz)

Teste em casa para n=27. Quantas vezes o laço é executado? (O maior valor atingido é de 9232).

Instruções de interrupção: Break

As vezes é necessário terminar/interromper a execução de instruções dentro de um laço:

```
def funcaoB():
    i=1
    while i>0:
        print i
        if i==2013:
            break
        i=i+1
    print 'finalizou'
```

Instruções de interrupção: Break

As vezes é necessário terminar/interromper a execução de instruções dentro de um laço:

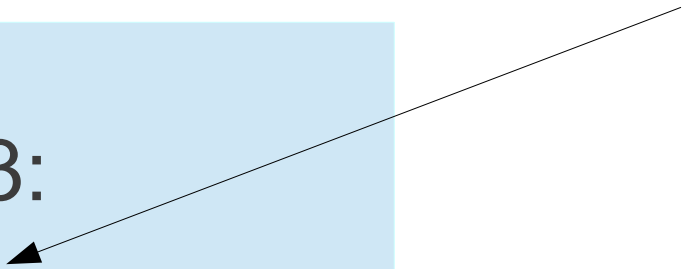
```
def funcaoB():
    i=1
    while i>0:
        print i
        if i==2013:
            break
        i=i+1
    print 'finalizou'
```

```
>>> funcaoB(3)
1
2
3
...
2011
2012
2013
finalizou
```

Instruções de interrupção: Break

As vezes é necessário terminar/interromper a execução de instruções dentro de um laço:

```
def funcaoB():
    i=1
    while i>0:
        print i
        if i==2013:
            break
        i=i+1
    print 'finalizou'
```



A instrução **break** permite interromper (imediatamente) o laço.

Instruções de interrupção: Break

As vezes é necessário terminar/interromper a execução de instruções dentro de um laço:

```
def funcaoB():
    i=1
    while i>0:
        print i
        if i==2013:
            break
        i=i+1
    print 'finalizou'
```

A instrução **break** permite interromper (imediatamente) o laço.

A instrução **i=1+1** não será executada caso i=2013.

Instruções de interrupção: Break

Questão: Faça uma função arctan que recebe o número real $x \in [0,1]$ e devolve uma aproximação do arco tangente de x (em radianos) através da série:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

Incluindo todos os termos da série até:

$$\left|\frac{x^k}{k}\right| < 0.0001$$

Instruções de interrupção: Break

Lembrando a solução da lista 02:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

```
def arctan(x):
    x = float(x)
    soma = 0
    sinal = -1
    for i in range(1,100+1):
        coef  = 2*i-1
        sinal = sinal*-1
        soma  = soma + sinal * (x**coef)/coef
    return soma
```

Instruções de interrupção: Break

```
def arctan2(x):
    x = float(x)
    soma  = 0
    sinal = -1
    i = 1
    while True:
        coef  = 2*i-1
        sinal = sinal*-1
        termo = (x**coef)/coef
        soma  = soma + sinal * termo
        if abs(termo)<0.0001:
            break
        i=i+1
    return soma
```

Instruções de interrupção: Break

```
def arctan2(x):
    x = float(x)
    soma  = 0
    sinal = -1
    i = 1
    while True:
        coef  = 2*i-1
        sinal = sinal*-1
        termo = (x**coef)/coef
        soma  = soma + sinal * termo
        if abs(termo)<0.0001:
            break
        i=i+1
    return soma
```

Considerando 100 primeiros termos:

```
>>> arctan(0.95)
0.7597626673838579
```

Considerando o módulo do i-éssimo termino:

```
>>> arctan2(0.95)
0.7598055115958504
```

Laços aninhados

Similar as outras estruturas condicionais, podemos ter laços dentro do escopo de outro laço, i.e., laços aninhado.

```
def tabuada(a,b):
    for i in range(a,b+1):
        print "\nTabuada: "+str(i)
        for j in range(1,11):
            print str(i)+"x"+str(j)+"="+str(i*j)
```

Laços aninhados

Similar as outras estruturas condicionais, podemos ter laços dentro do escopo de outro laço, i.e., laços aninhado.

```
def tabuada(a,b):
    for i in range(a,b+1):
        print "\nTabuada: "+str(i)
        for j in range(1,11):
            print str(i)+"x"+str(j)+"="+str(i*j)
```

```
>>> tabuada(7,8)

Tabuada: 7
7x1=7
7x2=14
7x3=21
7x4=28
7x5=35
7x6=42
7x7=49
7x8=56
7x9=63
7x10=70

Tabuada: 8
8x1=8
8x2=16
8x3=24
8x4=32
8x5=40
8x6=48
8x7=56
8x8=64
8x9=72
8x10=80
```

Laços aninhados

Dados os números reais **x** e ϵ>0, calcular a aproximação para e^x usando a seguinte expansão da Série de Taylor:

$$1+\frac{x^1}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\frac{x^5}{5!}+\cdots$$

Incluindo todos os termos da série até:

$$\left|\frac{x^k}{k!}\right| < \epsilon$$

Laços aninhados

```
def exp2(x, epsilon):
    x = float(x)
    soma  = 1
    i = 1
    while True:
        fact=1
        for i in range(1,i+1):
            fact = fact*i
        termo = (x**i)/fact
        soma  = soma + termo
        if abs(termo)<epsilon:
            break
        i=i+1
    return soma
```

Laços aninhados

```
def exp2(x, epsilon):
    x = float(x)
    soma  = 1
    i = 1
    while True:
        fact=1
        for i in range(1,i+1):
            fact = fact*i
        termo = (x**i)/fact
        soma  = soma + termo
        if abs(termo)<epsilon:
            break
        i=i+1
    return soma
```

Comparação entre Funções:

```
>>> exp(1)
2.718281828459045

>>> exp2(1, 0.01)
2.7166666666666663
>>>
```

Atividade em aula

Questão1. Dada a seguinte função

```
def funcaoEnigma1(p, q):
    soma = 0
    for i in range(p,q+1):
        for j in range(1,i+1):
            soma = soma + i
        if soma==0:
            break
    return soma
```

(a) Indique algebricamente (de forma concisa) a somatória que a seguinte função realiza.

(b) Qual é o resultado para o chamado a função com os parâmetros p=3, e q=4.

Atividade em aula

Questão2. Escreva uma função **`encaixa`** que, recebendo dois números inteiros **a** e **b** como parâmetros, verifica se **b** corresponde a os últimos dígitos de **a**.

Ex.:

a	b		
567890	890	$\Rightarrow$	encaixa
1243	1243	$\Rightarrow$	encaixa
2457	245	$\Rightarrow$	não encaixa
457	2457	$\Rightarrow$	não encaixa

Atividade em aula

Avaliação:

Questão 1: 4 pontos

Questão 2: 6 pontos

Atividade em aula

Questão1. Dada a seguinte função

```
def funcaoEnigma1(p, q):
    soma = 0
    for i in range(p,q+1):
        for j in range(1,i+1):
            soma = soma + i
        if soma==0:
            break
    return soma
```

Resposta:

(a) $= \sum_{i=p}^{q} \sum_{j=1}^{i} i$

(1 ponto)

$= \sum_{i=p}^{q} i^2$

(2 pontos)

(b) 25 (2 pontos)

(a) Indique algebricamente (de forma concisa) a somatória que a seguinte função realiza.

(b) Qual é o resultado para o chamado a função com os parâmetros p=3, e q=4.

Atividade em aula

Questão2: Uma solução

```
def encaixa(a, b):
    while True:
        if a%10==b%10:
            a = a/10
            b = b/10
            if b==0:
                return "encaixa"
        else:
            return "nao encaixa"
```

(6 pontos)

Lista 03

Questão única. O matemático Srinivasa Ramanujan encontrou uma série infinita, que pode ser usada para gerar uma aproximação numérica de pi:

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{(k!)^4 396^{4k}}$$

Escreva uma função chamada `estimar_pi` que usa esta fórmula para calcular e retornar uma estimativa de π. Deve usar um laço while para calcular os termos de soma até o último termo ser menor que 1e-15 (que é a notação Python para 10^{-15}).

Compare o resultado com a constante já implementada em Python, isto é, mostre o resultado para a seguinte expressão:

```
>>> pi - estimar_pi()
```

Lista 03

A entrega da Lista 03 deverá ser realizada através do Tidia-ae.

Seção Atividades/lista-03. Até 17/05 (23h50) – Sexta-feira.

Apenas deve ser enviado um arquivo PDF contendo a solução das questões. O documento deve ter o seguinte nome: RA-SeuNomeCompleto-Lista-03.pdf