Processamento da Informação
– Teoria –

Laços

Semana 02

Prof. Jesús P. Mena-Chalco

04/05/2013

Laços (estruturas de repetição)

Usado em situações em que é necessário repetir um determinado trecho de um programa, um determinado número de vezes.

Duas formas:

- Escrever o trecho quantas vezes for necessário, ou
- Utilizar o conceito de Laços.

Laços (for)

Da primeira aula: Mostrar uma sequência de números de 1 até n:

```
def imprimir_sequencia(n):
    for i in range(1,n+1):
        print i
```

Laços (for)

Da primeira aula: Mostrar uma sequência de números de 1 até n:

```
def imprimir_sequencia(n):
    for i in range(1,n+1):
        print i
```

```
>>> imprimir_sequencia(3)
1
2
3
```

Laços (for)

```
def imprimir_sequencia(n):
    for i in range(1,n+1):
        print i
```

```
>>> range(1,4)
[1, 2, 3]

>>> range(0,4)
[0, 1, 2, 3]

>>> range(10,20)
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

Laços (for)

Crie uma função que permita mostrar a sequência de números, no intervalo [x,y], considere x<y:

```
def imprimir_intervalo(x,y):
    for i in range(x,y+1):
        print i
```

```
>>> imprimir_intervalo(1,3)
1
2
3
```

Laços (for)

Crie uma função que permita mostrar a sequência de números, no intervalo [x,y], considere x<y:

```
def imprimir_intervalo(x,y):
    for i in range(x,y+1):
        print i
```

```
>>> imprimir_intervalo(-3,3)
```



```
-3
-2
-1
 0
 1
 2
 3
```

Laços (for)

Crie uma função que permita somar a sequência de números inteiros, no intervalo [x,y], considere x<y:

```
def somar_intervalo(x,y):
    soma = 0
    for i in range(x,y+1):
        soma = soma+i
    return soma
```

Laços (for)

Crie uma função que permita somar a sequência de números inteiros, no intervalo [x,y], considere x<y:

```
def somar_intervalo(x,y):
    soma = 0
    for i in range(x,y+1):
        soma = soma+i
    return soma
```

```
>>> somar_intervalo(0,4)
10
>>> somar_intervalo(-4,4)
0
>>> somar_intervalo(10,12)
33
```

Laços (for)

Crie uma função (com nome somaP), em que dado um inteiro n>0, permita somar a seguinte sequência:

$$1^2 + 2^2 + 3^2 + \cdots + n^2$$

Laços (for)

$$1^2 + 2^2 + 3^2 + \cdots + n^2$$

```
def somaP(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma + i**2
    return soma
```

Laços (for)

$$1^2 + 2^2 + 3^2 + \cdots + n^2$$

```
def somaP(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma + i**2
    return soma
```

```
>>> somaP(1)
1
>>> somaP(2)
5
>>> somaP(3)
14
>>> somaP(4)
30
>>> somaP(5)
55
```

Laços (for)

Modifique a função **somaP** em que, além do número **n**, seja utilizado um outro número **k**, de tal forma que o calculo da seguinte somatória seja realizada:

$$1^k + 2^k + 3^k + \cdots + n^k$$

Laços (for)

Modifique a função **somaP** em que, além do número **n**, seja utilizado um outro número **k**, de tal forma que o calculo da seguinte somatória seja realizada:

$$1^k + 2^k + 3^k + \cdots + n^k$$

```
def somaP(n, k):
    soma = 0
    for i in range(1,n+1):
        soma = soma + i**k
    return soma
```

Laços (for)

Modifique a função **somaP** em que, além do número **n**, seja utilizado um outro número **k**, de tal forma que o calculo da seguinte somatória seja realizada:

$$k^1 + k^2 + k^3 + \cdots + k^n$$

Laços (for)

Modifique a função **somaP** em que, além do número **n**, seja utilizado um outro número **k**, de tal forma que o calculo da seguinte somatória seja realizada:

$$k^1 + k^2 + k^3 + \cdots + k^n$$

```
def somaP(n, k):
    soma = 0
    for i in range(1,n+1):
        soma = soma + k**i
    return soma
```

Laços (for e while)

Mostrar a palavra SPAM **n** vezes:

```
def imprimir_spam(n):
    for i in range(1,n+1):
        print "SPAM"
```

```
def imprimir_spam(n):
    i = 1
    while i <= n:
        print "SPAM"
        i = i+1
```

```
>>> imprimir_spam(7)
SPAM
SPAM
SPAM
SPAM
SPAM
SPAM
SPAM
```

Laços (for e while)

Mostrar uma sequência de números de 1 até n:

```
def imprimir_sequencia(n):
    for i in range(1,n+1):
        print i
```

```
def imprimir_sequencia(n):
    i = 1
    while i <= n:
        print i
        i = i+1
```

Laços (for e while)

Mostrar uma sequência de números de 1 até n:

```
def imprimir_sequencia(n):
    for i in range(1,n+1):
        print i
```

Cada iteração será executada para a sequência de valores de i = [1,n]

```
def imprimir_sequencia(n):
    i = 1
    while i <= n:
        print i
        i = i+1
```

Cada iteração será executada para valores de i<=n
(Aqui usa-se uma condição)

Laços (for e while)

Mostrar uma sequência de números **ímpares** de 1 até n:

```
def imprimir_sequencia(n):
    i = 1
    while i <= n:
        print i
        i = i+2
```

```
>>> imprimir_sequencia(7)
1
3
5
7
```

Laços (for e while)... do exercício anterior

$$k^1 + k^2 + k^3 + \cdots + k^n$$

```
def somaP(n, k):
    soma = 0
    for i in range(1,n+1):
        soma = soma + k**i
    return soma
```

```
def somaP(n, k):
    soma = 0
    i = 1
    while i <=n:
        soma = soma + k**i
        i = i+1
    return soma
```

Laços (for e while)

Crie uma função em que, dado um inteiro não-negativo **n**, seja possível determinar **n!**

Laços (for e while)

Crie uma função em que, dado um inteiro não-negativo **n**, seja possível determinar **n!**

```
def fatorial1(n):
    mult = 1
    for p in range(1,n+1):
        mult = mult*p
    return mult
```

```
def fatorial2(n):
    mult = 1
    p = 1
    while p<=n:
        mult = mult*p
        p = p+1
    return mult
```

Laços: Atividade em aula 01

Dizemos que um número natural é *triangular* se ele é produto de três número naturais consecutivos.

Exemplo: 120 é triangular, pois 4*5*6 = 120.

2730 é triangular, pois 13*14*15 = 2730.

Dado um inteiro não negativo **n**, verificar se **n** é triangular.

Cabeçalho da função: def numeroTriangular(n):
Devolve “True” se o número for triangular, caso contrário “False”.

Laços: Atividade em aula 01

```
def numeroTriangular(n):
    i=3
    while i<=n:
        if (i-2)*(i-1)*(i)==n:
            return True
        i=i+1
    return False
```

Laços: Atividade em aula 01

```
def numeroTriangular(n):
    i=3
    while i<=n:
        if (i-2)*(i-1)*(i)==n:
            return True
        i=i+1
    return False
```

```
def numeroTriangular(n):
    for i in range(3,n+1):
        if (i-2)*(i-1)*(i)==n:
            return True
    return False
```

Laços: Atividade em aula 02

Faça uma função que calcula a soma:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots + \frac{1}{9999} - \frac{1}{10000}$$

Pelas seguintes maneiras:

- Adição de termos da esquerda para a direita;
- Adição de termos da direita para a esquerda;
- Adição separada dos termos positivos e dos termos negativos da esquerda para a direita;
- Adição separada dos termos positivos e dos termos negativos da direita para a esquerda.

Compare e discuta os resultados obtidos no computador.

Laços: Atividade em aula 02

```
def equacaoE():
    soma = 0
    i=1
    while i<=1000:
        soma = soma + (1.0/i)*((-1)**(i+1))
        i=i+1
    return soma
```

Adição de termos da esquerda para a direita

Laços: Atividade em aula 02

```
def equacaoE():
    soma = 0
    i=1000
    while i>=1:
        soma = soma + (1.0/i)*((-1)**(i+1))
        i=i-1
    return soma
```

Adição de termos da direita para a esquerda

Lista 02: Funções e laços

Questão 1:

Crie uma função que permita somar apenas os números impares da sequência de inteiros contida no intervalos [x,y], para x<y. O cabeçalho da função deve ser o seguinte:

def soma_impares(x,y):

Questão 2:

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

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

Cabeçalho: def soma():

Lista 02: Funções e laços

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:

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

Cabeçalho: def arctan(x):

Obs.: Considere somente os 100 primeiros termos da série.

Lista 02: Funções e laços

A entrega da Lista 02 deverá ser realizada através do Tidia-ae.

Seção Atividades/lista-02. Até 10/05 (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-02.pdf

O gabarito será apresentado em sala de aula 11/05.