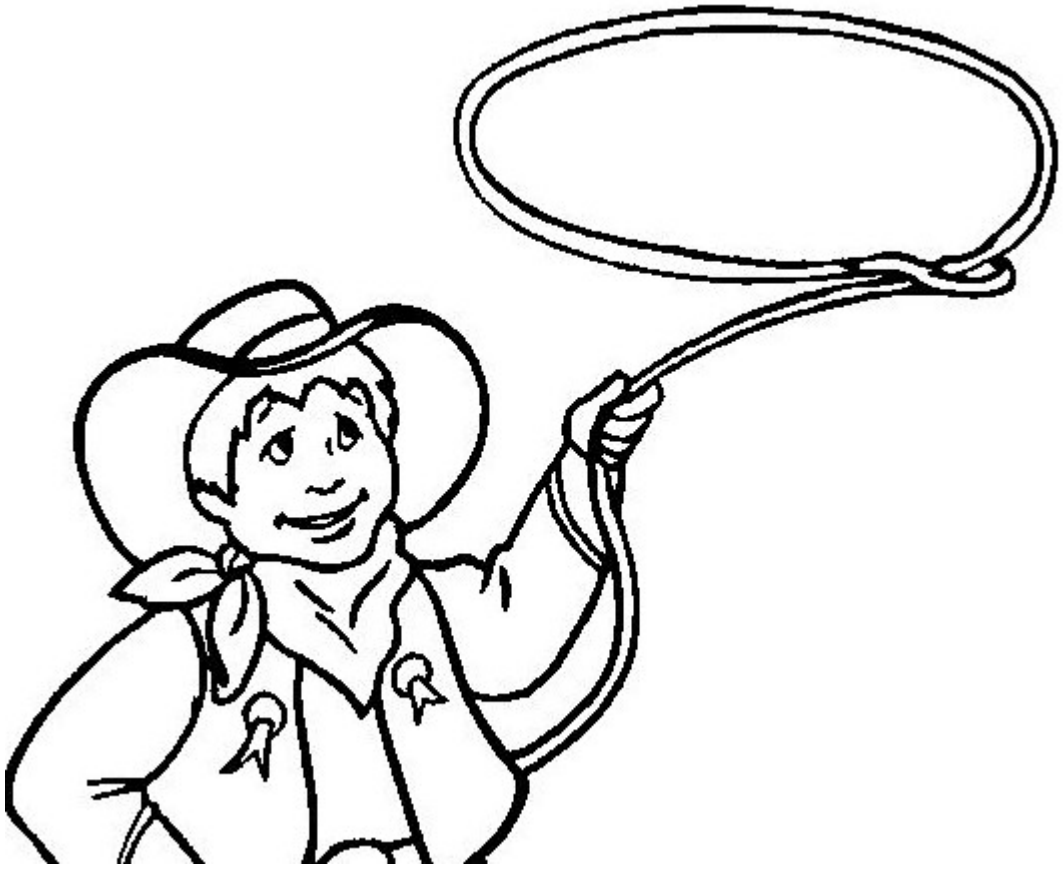
Processamento da Informação

Estruturas de repetição - Parte 2

Prof. Jesús P. Mena-Chalco
CMCC/UFABC

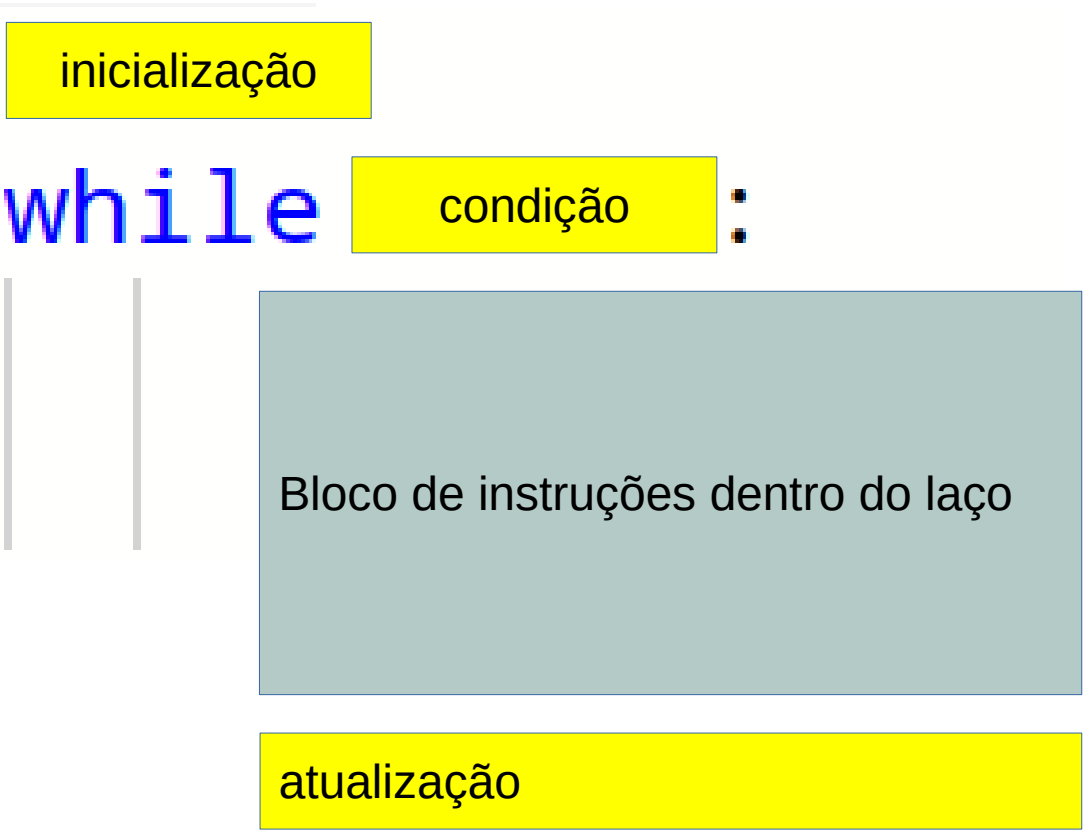
Q1/2020

Estrutura de repetição: laço



Fonte: http://www.kumamon.co/

Estrutura de repetição: laço while



```python
n = 1
while n<=10:
    print("Funciona!")
    n = n+1
```

```
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
```

```
n = 1
while n<=10:
    print("Funciona!")
    n = n+1
```

Ambas as versão
Imprimem 10 vezes “Funciona!”

```
for n in [1,2,3,4,5,6,7,8,9,10]:
  print("Funciona!")
```

Estrutura de repetição: laço for

```
for variável in Lista-de-elementos :
```

Bloco de instruções dentro do laço

A **variável** terá todos os valores definida na **Lista-de-elementos**

A função range

range

A função **range** permite criar um objeto que contém uma sequência de números inteiros.

Quando usado com apenas um único parâmetro, a sequência inicia em 0 e continua até o número inteiro antes do valor dado como parâmetro.

```
range(10)
```

Gera a sequência:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

range

```
range(5)
```

Gera a sequência:

```
[0, 1, 2, 3, 4]
```

```
range(1)
```

Gera a sequência:

```
[0]
```

```
range(0)
```

Gera a sequência:

```
[]
```

range

Podemos usar 2 parâmetros na função **range**:

```
range(5, 10)
```

Gera a sequência:

```
[5, 6, 7, 8, 9]
```

```
range(1, 7)
```

Gera a sequência:

```
[1, 2, 3, 4, 5, 6]
```

range

Podemos usar 3 parâmetros na função **range**:

```
range(5,10,2)
[5, 7, 9]

range(5,10,3)
[5, 8]

range(5,10,4)
[5, 9]

range(5,10,5)
[5]
```

O terceiro parâmetro
Indica o incremento (ou decremento)

range

O terceiro parâmetro
Indica o incremento (ou decremento)

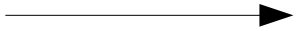
```
range(10,1,-1)
[10, 9, 8, 7, 6, 5, 4, 3, 2]

range(10,1,-2)
[10, 8, 6, 4, 2]

range(10,1,-3)
[10, 7, 4]
```

Exemplo

```
for n in range(1,11):
    print("Funciona!")
```



```
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
Funciona!
```

Exercícios com laço `for`

Exercício 01: Somatória de números inteiros

Dados dois inteiros, ***a*** e ***b***, com ***a<=b***, crie uma função que permita somar todos os números entre ***a*** e ***b***, eles incluídos.

```
def soma_intervalo(a: int, b: int) -> int:
```

Exercício 01: Somatória de números inteiros

```python
def soma_intervalo(a:int, b:int) -> int:
    soma = 0
    for n in range(a, b+1):
        soma = soma+n
    return soma
```

Exercício 02: Somatória de números ímpares

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

Assinatura:

```
def soma_impares(x:int, y:int) -> int:
```

Exercício 02: Somatória de números ímpares

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

Exercício 03: Somatória de termos

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

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

Assinatura:

```
def soma_termos(n:int) -> float:
```

Exercício 03: Somatória de termos

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

Exercício 04: Aproximação de π

Uma aproximação de π bem conhecida é aquela calculada a partir da **Equação de Leibniz**:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

Crie uma função que permita calcular essa aproximação de π usando os primeiros **1000** termos.

Assinatura: `def pi() -> float:`

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}$$

Exercício 04: Aproximação de π

```
def pi() -> float:
    soma = 0

    for i in range(0, 1000):
        soma = soma + (-1)**i/(2*i+1)

    return 4*soma
```

```
3.140592653839794
```

Exercício 05: Laços aninhados

Para **x=2** e **y=3** o que realiza a seguinte função?

```
def imprime_numeros(x:int, y:int):
    for i in range(1,x+1):
        for j in range(1,y+1):
            print("{} - {}".format(i, j))
```

Exercício 05: Laços aninhados

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

Atividade em aula

Questão 1 (a) e (b)

```python
def m1(n: int) -> int:
    soma = 0
    for i in range(1, n):
        soma = soma+i
    return soma
```

```python
def m2(n: int) -> int:
    soma = 0
    for i in range(n-1, 0, -1):
        soma = soma+i
    return soma
```

As funções devolvem a soma dos n-1 primeiros números naturais, para n>=1

Questão 1 (c)

```
def m3(num1: int, num2:int) -> int:
    soma = 0
    for p in range(num1):
        for r in range(num2):
            soma = soma+1
    return soma
```

A função devolve o produto entre num1 e num2

Questão 2

```python
def enigma(n: int) -> int:
    soma = 0
    for p in range(1, n+1):
        for r in range(p, n+1):
            soma = soma+1
    return soma
```

Resposta correta +2 pontos
Caso contrário -2 pontos

$$
\begin{array}{ll}
(a) & n^2/2 \\
(b) & (n^2+n)/2 \\
(c) & (n^2-n)/2 \\
(d) & 0
\end{array}
$$

Questão 3

```
def arctan1(x: float) -> float:
    soma = 0
    for i in range(100):
        coef  = 2*i+1
        soma  = soma + (-1)**(i) * (x**coef)/coef
    return soma
```

Questão 3

```
def arctan2(x: float) -> float:
    soma = 0
    sinal = -1
    for i in range(100):
        coef  = 2*i+1
        sinal = sinal*-1
        soma  = soma + sinal * (x**coef)/coef
    return soma
```