# Processamento da informação

Estruturas de repetição - parte 2

Profa. Debora Medeiros

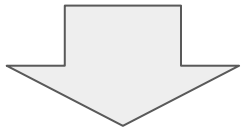
# Recaptulando

```
n = 1
while n <= 10:
    print("A capital de Montana não é Hannah")
    n += 1
```

```
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
```

# Versão com *for*

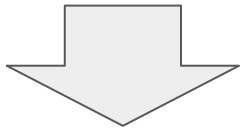
```
n = 1
while n <= 10:
    print("A capital de Montana não é Hannah")
    n += 1
```



```
for n in [1,2,3,4,5,6,7,8,9,10]:
    print("A capital de Montana não é Hannah")
```

# Versão com *for*

```
n = 1
while n <= 10:
    print("A capital de Montana não é Hannah")
    n += 1
```



```
for n in [1,2,3,4,5,6,7,8,9,10]:
    print("A capital de Montana não é Hannah")
```

```
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
```

# 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**

# Função *range*

- Cria um objeto que contém uma sequência de números inteiros.
  - 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]

# Função *range*

- range(5)
  - Gera a sequência: [0, 1, 2, 3, 4]
- range(1)
  - Gera a sequência: [0]
- range(0)
  - Gera a sequência: []

# Função *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]

# Função *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]
  - 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]

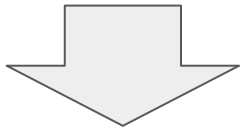
O terceiro parâmetro
Indica o incremento
(ou decremento)

# Versão com *range*

```
for n in [1,2,3,4,5,6,7,8,9,10]:
    print("A capital de Montana não é Hannah")
```

# Versão com *range*

```
for n in [1,2,3,4,5,6,7,8,9,10]:
    print("A capital de Montana não é Hannah")
```



```
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
A capital de Montana não é Hannah
```

```
for n in range(1,11):
    print("A capital de Montana não é Hannah")
```

# Exercício

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

# Exercício

- 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, b):
    total = 0
    for i in range(a, b + 1):
        total += i
    return total
```

# Exercício

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

# Exercício

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

```
def soma_impares(a, b):
    total = 0
    for i in range(a, b + 1):
        if i % 2 == 1:
            total += i
    return total
```

# Exercício

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

```
def soma_impares(a, b):
    total = 0
    for i in range(a, b + 1):
        if i % 2 == 1:
            total += i
    return total
```

```
def soma_impares2(a, b):
    total = 0
    if a % 2 == 0:
        a += 1
    for i in range(a, b + 1, 2):
        total += i
    return total
```

# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
> comb(2, 3)
0 - 0
0 - 1
0 - 2
1 - 0
1 - 1
1 - 2
```

# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
> comb(2, 3)
0 - 0
```

| i | |
|---|---|
| 0 | 1 |

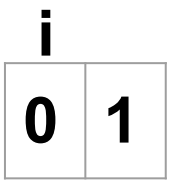
| j | | |
|---|---|---|
| 0 | 1 | 2 |

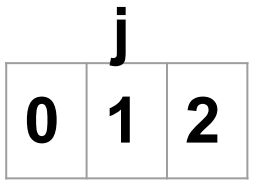
# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
comb(2, 3)
0 - 0
0 - 1
```



# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
comb(2, 3)
0 - 0
0 - 1
0 - 2
```

**i**

| 0 | 1 |
|---|---|

**j**

| 0 | 1 | 2 |
|---|---|---|

# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
comb(2, 3)
0 - 0
0 - 1
0 - 2
1 - 0
```

| i | |
|---|---|
| 0 | 1 |

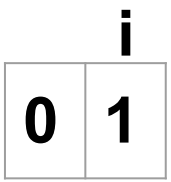
| j | | |
|---|---|---|
| 0 | 1 | 2 |

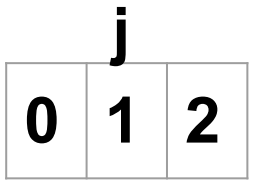
# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
comb(2, 3)
0 - 0
0 - 1
0 - 2
1 - 0
1 - 1
```



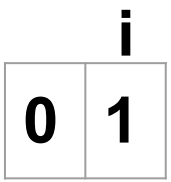



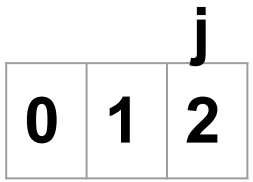
# Laços aninhados

- O que realiza a seguinte função?

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
comb(2, 3)
0 - 0
0 - 1
0 - 2
1 - 0
1 - 1
1 - 2
```



# Outros tipos de dados

```
escolha = ["pedra", "papel", "tesoura"]
print("Quem ganha?\nJogador 1 - Jogador 2")
for i in escolha:
    for j in escolha:
        print("{} - {}".format(i, j))
```

```
Quem ganha?
Jogador 1 - Jogador 2
pedra - pedra
pedra - papel
pedra - tesoura
papel - pedra
papel - papel
papel - tesoura
tesoura - pedra
tesoura - papel
tesoura - tesoura
```

# Combinações sem repetição

```
def comb(a, b):
    for i in range(a):
        for j in range(b):
            print("{} - {}".format(i, j))
```

```
comb(2, 3)
0 - 0
0 - 1
0 - 2
1 - 0
1 - 1
1 - 2
```

# Combinações sem repetição

```
def comb2(a, b):
  for i in range(a):
    for j in range(i, b):
      print("{} - {}".format(i, j))
```

```
In [65]: comb2(2, 3)
0 - 0
0 - 1
0 - 2
1 - 1
1 - 2
```

# Referências


- Material do prof. Jesús P Mena-Chalco (UFABC)
- Material do prof. Thiago Covões (UFABC)