Processamento da Informação

Revisão p1

Profa. Debora Medeiros

Construção de programas

- Conjunto de instruções
  - Ordem sequencial lógica
  - Algoritmo

Linguagem de programação

- É um conjunto limitado de:
  - **Símbolos** (comandos, identificadores, caracteres, etc)
  - **Regras de sintaxe** (descrevem de forma precisa ações)

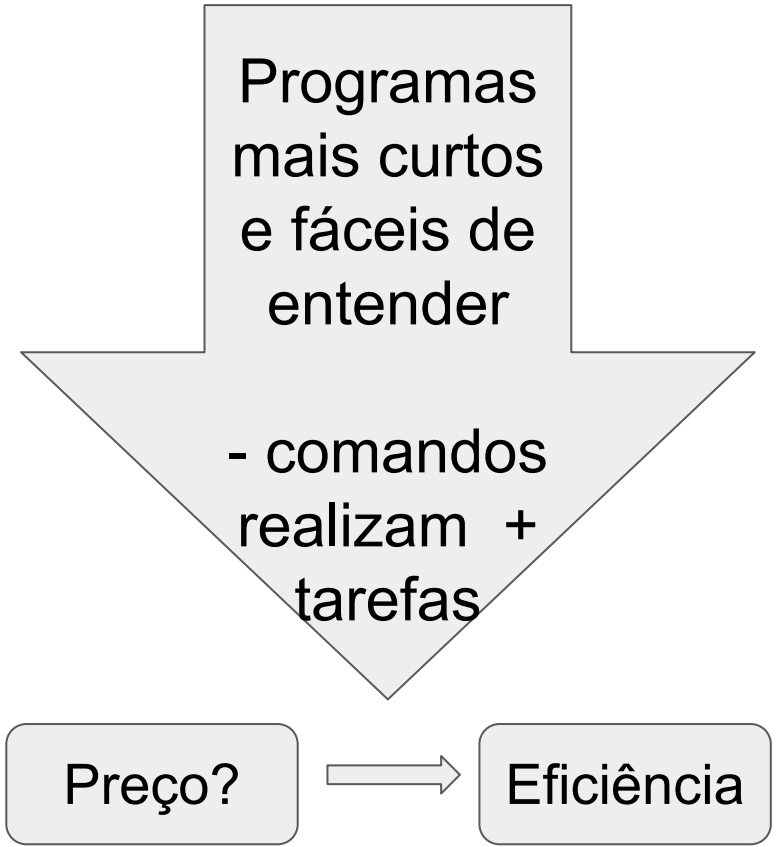
- Exemplo
  - **print** seguido de parênteses

Linguagem de programação

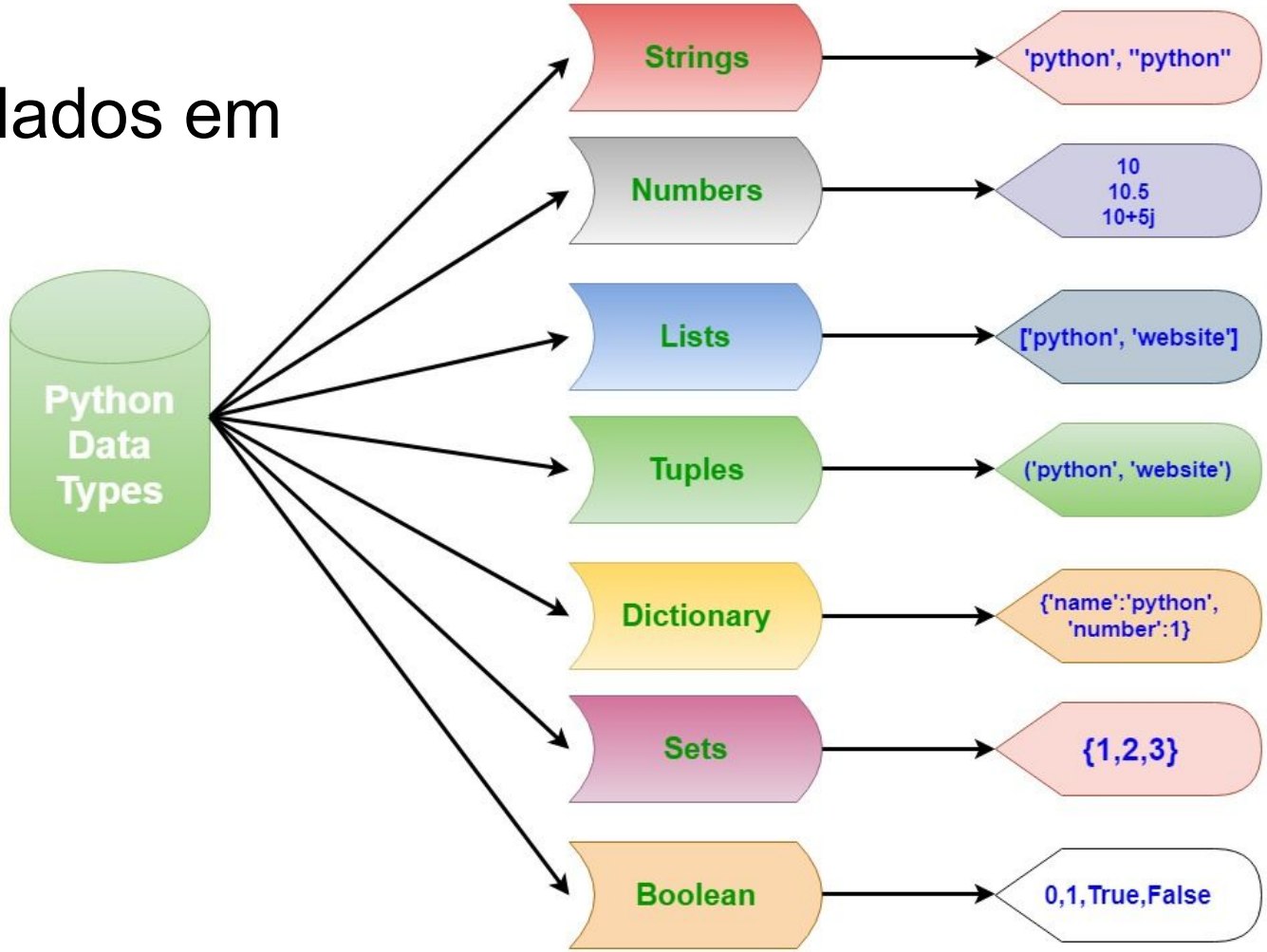
Linguagem de máquina	Compreendida pelo computador. Dependente da arquitetura do computador
Linguagem de baixo nível	Utiliza mnemonicos para a representação de ações elementares Ex. Assembler
Linguagem de alto nível	Utiliza instruções próximas da linguagem humana Ex. C, Java, Python, PHP

Linguagem de programação

Linguagem de máquina	Compreendida pelo computador. Dependente da arquitetura do computador
Linguagem de baixo nível	Utiliza mnemonicos para a representação de ações elementares Ex. Assembler
Linguagem de alto nível	Utiliza instruções próximas da linguagem humana Ex. C, Java, Python, PHP



Tipos de dados em Python



Tipos de dados em Python

Tipos de dados primitivos:

- Inteiros (int)
  - 3, 8, 3000
- Ponto flutuante (float)
  - 2.5, 3.33, 20.0
- String (str)
  - Sequência de caracteres, ex.: “olá”, “Maria”, “200”
- Listas (list)
  - [“may”, 4, “2021”]
- Booleano (bool)
  - Valor lógico: **True** ou **False**

Precedência de operadores

Qual seria o resultado da execução das seguintes instruções?

- 4 + 5 + 6 / 3 = 11
- 3 ** 2 + 2 = 11
- -2 ** 4 = -16

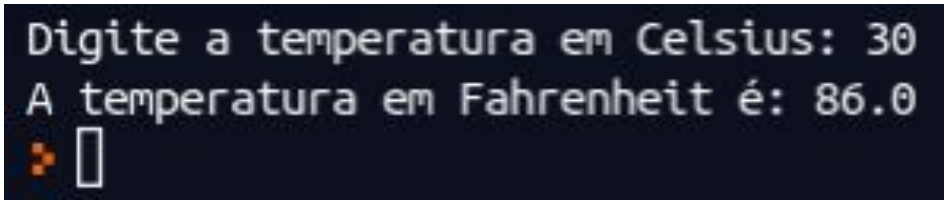
Precedência de operadores

Operador	Operação
**	Exponenciação
-	Negação
* / // %	Multiplicação Divisão Divisão inteira Módulo
+ -	Soma subtração

Um programinha com entrada e saída

Conversor de temperatura

```
C = int(input("Digite a temperatura em Celsius: "))
# a expressão a seguir converte uma temperatura de
Celsius para Fahrenheit
F = C * 9/5 + 32
print("A temperatura em Fahrenheit é: " + str(F));
```



Aula passada

- Quais dos comando abaixo vocês acham que daria erro:
  - 6 * ----------8
  - **8 = alunos**
  - ((((4 ** 3))))
  - (-(-(-(-5))))
  - **4 += 7 / 2**

- Problema: números recebendo valores
- += está ok

Modularização

- Emprego de módulos/funções/métodos
  - Ferramentas com entrada, saída e um procedimento específico
- Nas aulas anteriores chegamos a usar algumas funções:
  - min(5, 8, 2) → 2
  - round(2.567, 2) → 2.57

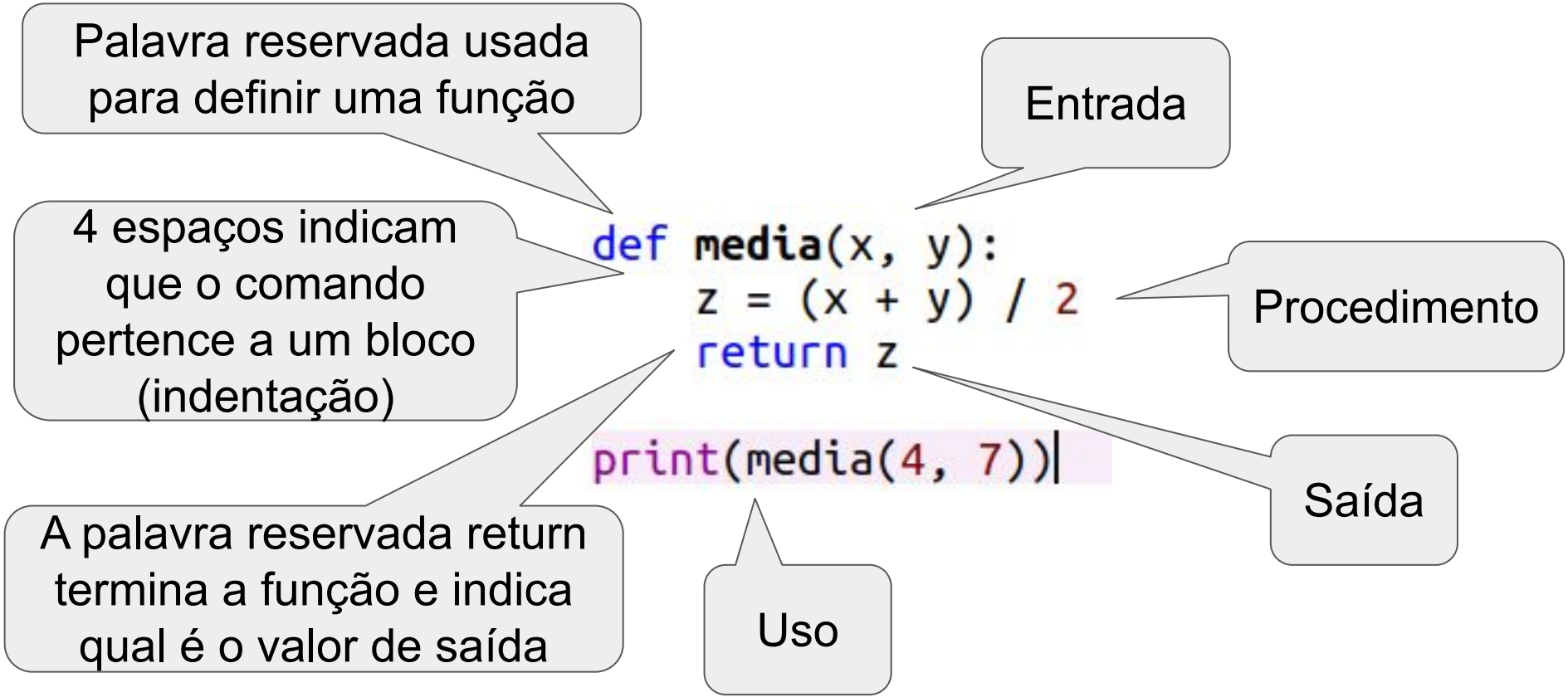
```
In [9]: help(round)
Help on built-in function round in module builtins:

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.  Otherwise
    the return value has the same type as the number.  ndigits may be negative.
```

- Agora vamos aprender a fazer nossas próprias funções

Definição de uma função



Importante

Cada função cria um ambiente específico para suas variáveis

```
def media(x, y):
    z = (x + y) / 2
    return z

print(media(4, 7)) #5.5
print(z)
```

```
Traceback (most recent call last):

  File "<ipython-input-67-893fa9747e51>", line 7, in <module>
    print(z)

NameError: name 'z' is not defined
```

Tuplas

- Conjunto pré-determinado de elementos

```
x = 3.987
print(type(x)) #<class 'float'>
y = (x, x)
print(type(y)) #<class 'tuple'>
z = (1.2, 0)
print(type(z)) #<class 'tuple'>
print(z[0]) #1.2
w = 1, "um"
print(type(w)) #<class 'tuple'>
a, b = z
print(a) #1.2
print(b) #0
```

Continuando Bhaskara

```
def bhaskara(a, b, c):
    delta = b**2 - 4*a*c
    xl = (-b - delta**0.5)/(2*a)
    xll = (-b + delta**0.5)/(2*a)
    return xl, xll

x1, x2 = bhaskara(1, 0, -1)
print(x1) #-1.0
print(x2) #1.0
```

Determinar se um número é par ou não

- Pensando de maneira informal
  1. Calcular o resto da divisão por 2
  2. Se o resto for 0, é par
  3. Senão, é ímpar
- Para isso, utilizamos o **if**

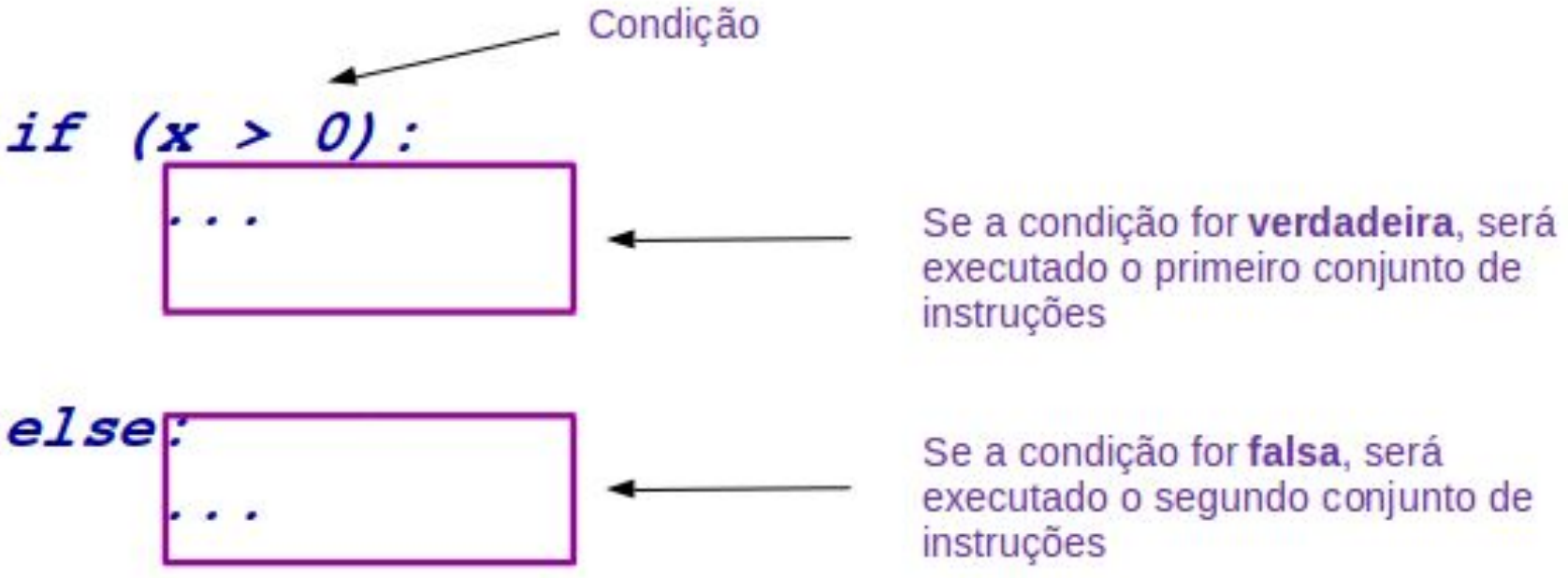
```
if (x>0):
  Instrucao1
  Instrucao2
  ...
```

Se a condição for verdade, então as instruções no bloco de instruções são executadas.

O bloco deve conter a mesma indentação.

Execução condicional

- Neste caso existem 2 possibilidades



Voltando ao exemplo dos números pares

```
def ehPar(x):
    if (x % 2 > 0):
        return False
    else:
        return True
```

Desafio

O que a seguinte função realiza?

```
def que(n, m, i):
    if(n < m):
        print(n)
        que(n + i, m, i)
```

Tente descobrir sem utilizar o compilador

Expressões booleanas

- O operador “==” é um dos operadores relacionais, os outros são:
  - x != y # x não é igual a y
  - x > y # x é maior que y
  - x < y # x é menor que y
  - x >= y # x é maior ou igual a y
  - x <= y # x é menor ou igual a y
- Um erro comum é usar “=” no lugar de “==”.

Operadores lógicos

- Em Python existem 3 operadores lógicos:
  - *and*
  - *or*
  - *not*
- A semântica destes operadores é similar ao seu significado em Inglês/Português.
- Exemplo: x > 0 and x < 10
  - É verdadeira somente se x é maior a 0 e menor do que 10

Exercício: soma

- Crie uma função em que, dados 3 números como parâmetros, permita verificar se a soma de quaisquer par de números gera a soma do terceiro número.
  - Tente não usar condicionais
  - Sua função deve devolver True ou False:

```
def verificaSoma(a, b, c):
    return a + b == c or a + c == b or b + c == a
```

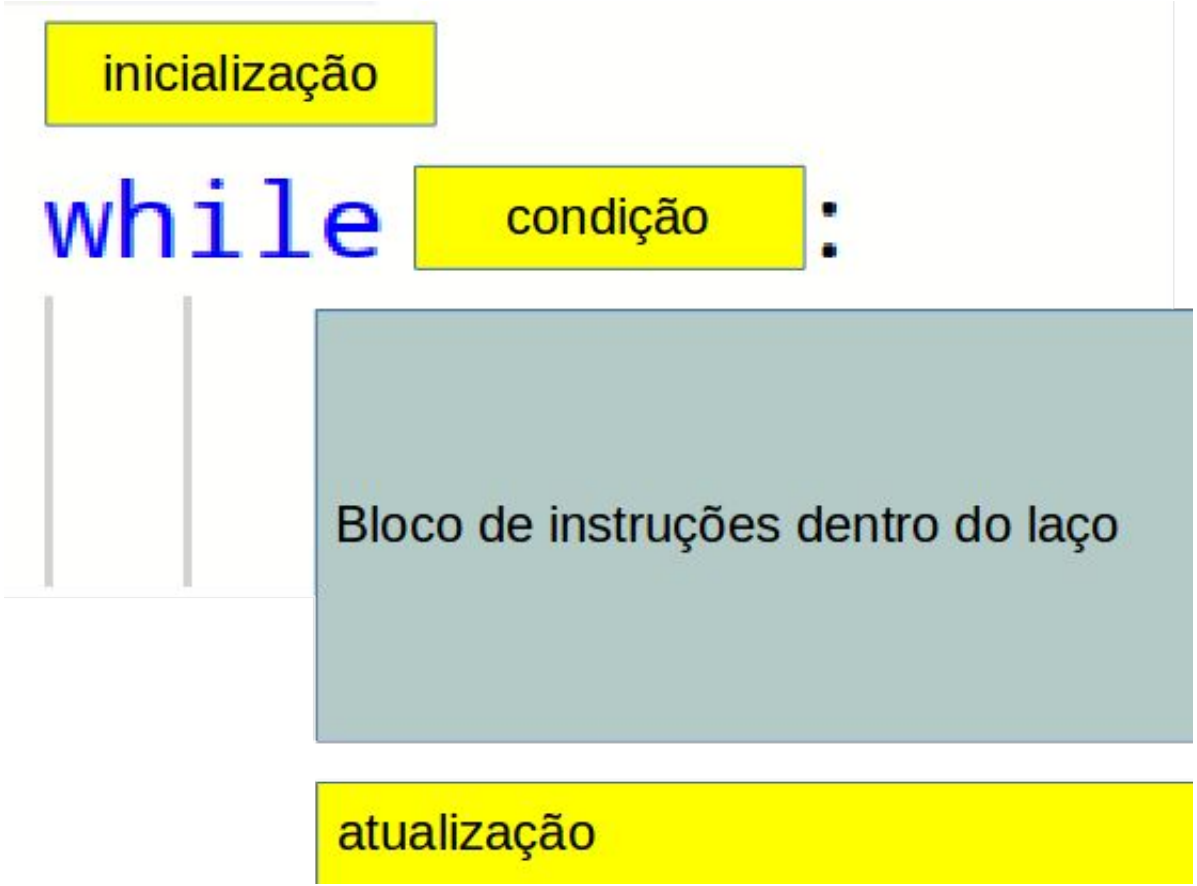
Organização

- Algumas coisas já comentamos
  - Organizar o código em funções
  - Dar nomes intuitivos a variáveis e métodos
    - anoNascimento
    - ano_nascimento

Organização

- Uso do método *main*
  - Para evitar instruções jogadas no meio do código
  - Concentrar o procedimento principal no método *main*
  - A única instrução que deve ficar fora de algum método é a chamada para o método *main*

Estrutura de repetição: laço

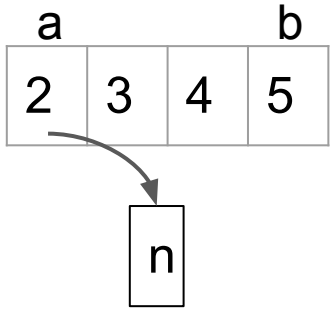


Exercício: somatório

- 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):
    soma = 0
    n = a
    while n <= b:
        soma += n
        n += 1
    return soma
```

```
soma_intervalo(2,5)
14
```

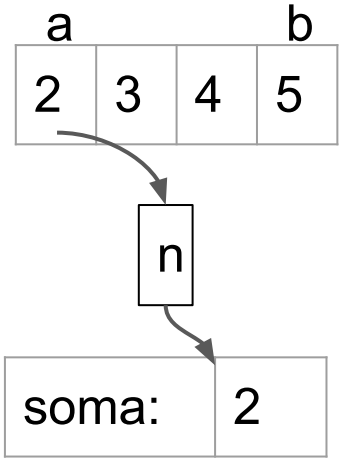


Exercício: somatório

- 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):
    soma = 0
    n = a
    while n <= b:
        soma += n
        n += 1
    return soma
```

```
soma_intervalo(2,5)
14
```

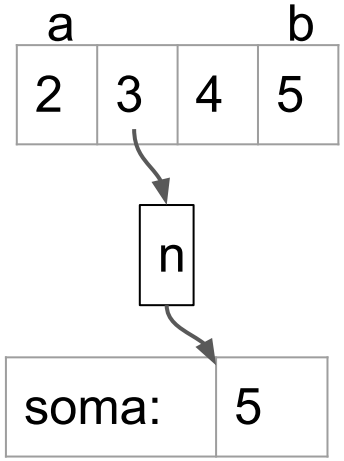


Exercício: somatório

- 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):
    soma = 0
    n = a
    while n <= b:
        soma += n
        n += 1
    return soma
```

```
soma_intervalo(2,5)
14
```

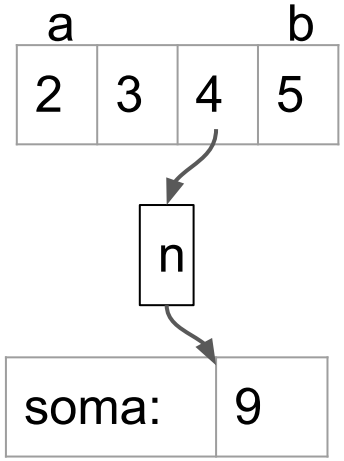


Exercício: somatório

- 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):
    soma = 0
    n = a
    while n <= b:
        soma += n
        n += 1
    return soma
```

```
soma_intervalo(2,5)
14
```

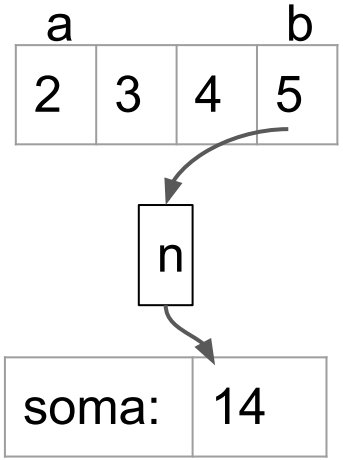


Exercício: somatório

- 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):
    soma = 0
    n = a
    while n <= b:
        soma += n
        n += 1
    return soma
```

```
soma_intervalo(2,5)
14
```



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**

Exemplo: contagem regressiva

```
n = 5
while n > 0:
    print(n)
    n -= 1
print("Feliz ano novo!")
```

```
for i in [5, 4, 3, 2, 1]:
    print(i)
print('Feliz ano novo!')
```

```
5
4
3
2
1
Feliz ano novo!
```

Função *range*

- range(5)
  - Gera a sequência: [0, 1, 2, 3, 4]
- range(5, 10)
  - Gera a sequência: [5, 6, 7, 8, 9]
- range(5,10,2)
  - Gera a sequência: [5, 7, 9]
- range(10,1,-2)
  - Gera a sequência: [10, 8, 6, 4, 2]

```
n = 5
while n > 0:
    print(n)
    n -= 1
print("Feliz ano novo!")
```

Exemplo:
contagem regressiva

```
5
4
3
2
1
Feliz ano novo!
```

```
for i in [5, 4, 3, 2, 1]:
    print(i)
print('Feliz ano novo!')
```

```
for i in range(5, 0, -1):
    print(i)
print('Feliz ano novo!')
```

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))
```

```
> 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

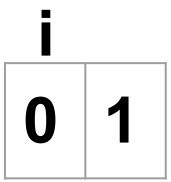
j		
0	1	2

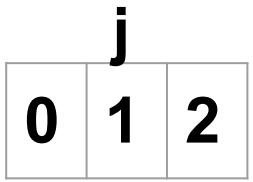
Laços aninhados

- 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
```



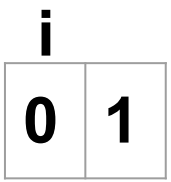


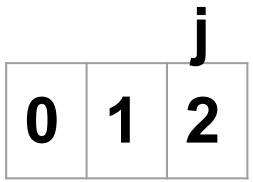
Laços aninhados

- 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
```





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

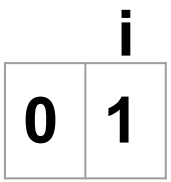
0	1	2

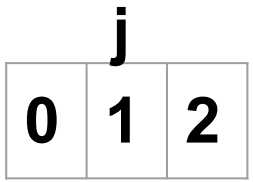
Laços aninhados

- 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
```



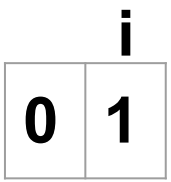


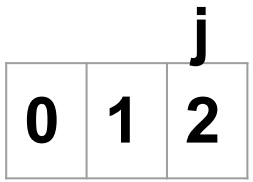
Laços aninhados

- 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
```





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)