

# **COMPUTAÇÃO 1 – PYTHON**

## **AULA 7 TEÓRICA**

**SILVIA BENZA**

**SILVIABENZA@COS.UFRJ.BR**

# **ESTRUTURAS DE REPETIÇÃO**

**VAMOS A APRENDER COMO TRABALHAR  
COM LOOPS!**

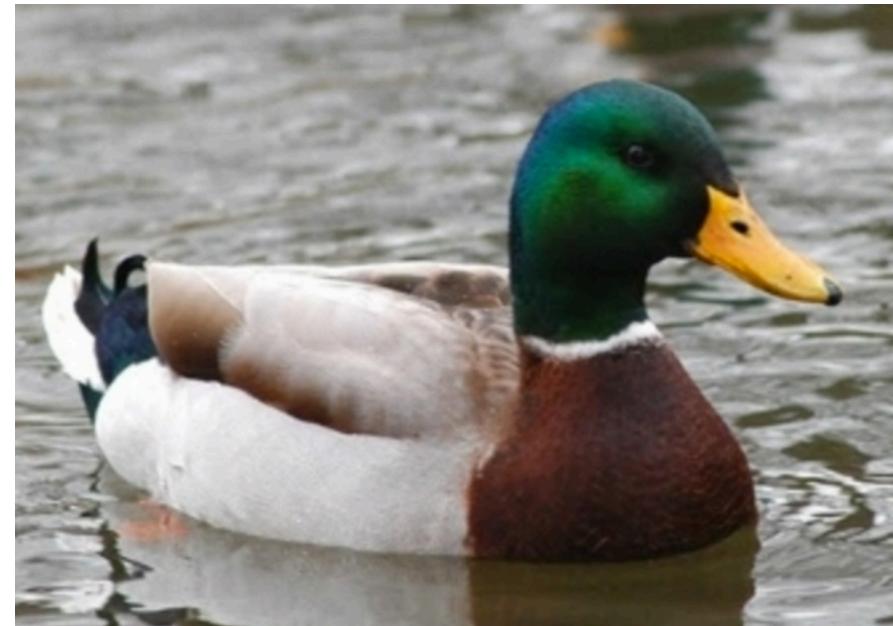
# WHILE

Permite que o programador especifique que o programa deve repetir um conjunto de comandos enquanto uma dada condição for verdadeira.

```
while condição:  
    conjunto de comandos
```

# WHILE

Permite que o programador especifique que o programa deve repetir um conjunto de comandos **enquanto uma dada condição for verdadeira.**



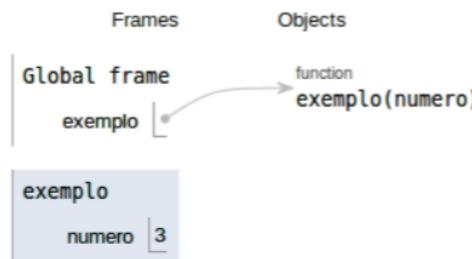
# WHILE

Permite que o programador especifique que o programa deve repetir um conjunto de comandos enquanto uma dada condição for verdadeira.

```
# Função que reduz em 1 o valor do numero passado como
# parâmetro até chegar a zero.
# int → str
def exemplo(numero):
    while numero > 0:
        numero = numero - 1
    return "boom!"
```

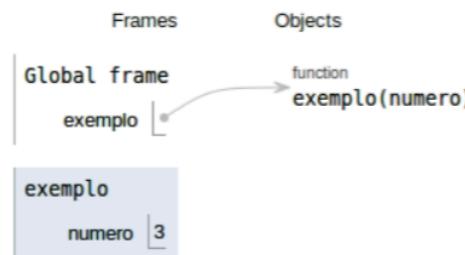
# WHILE

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero > 0:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(3)
```



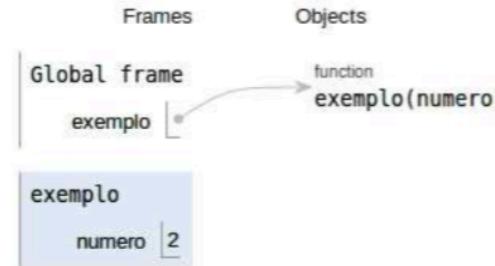
# WHILE

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero > 0:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(3)
```



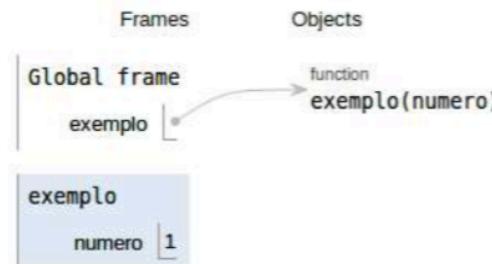
# WHILE

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero > 0:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(3)
```



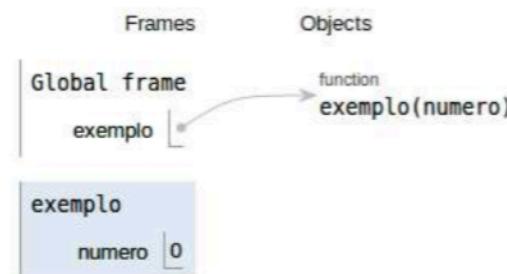
# WHILE

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero > 0:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(3)
```



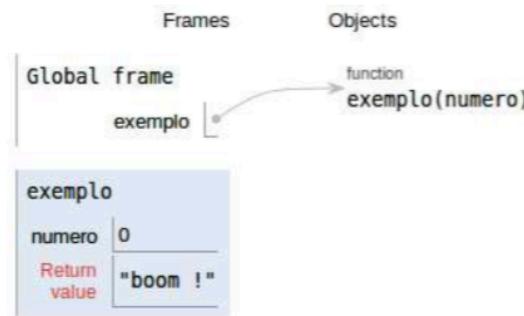
# WHILE

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero > 0:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(3)
```



# WHILE

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero > 0:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(3)
```



# WHILE

A condição é uma expressão ou dado do tipo **booleano** (True ou False), tal como os testes usados com o **comando IF**.

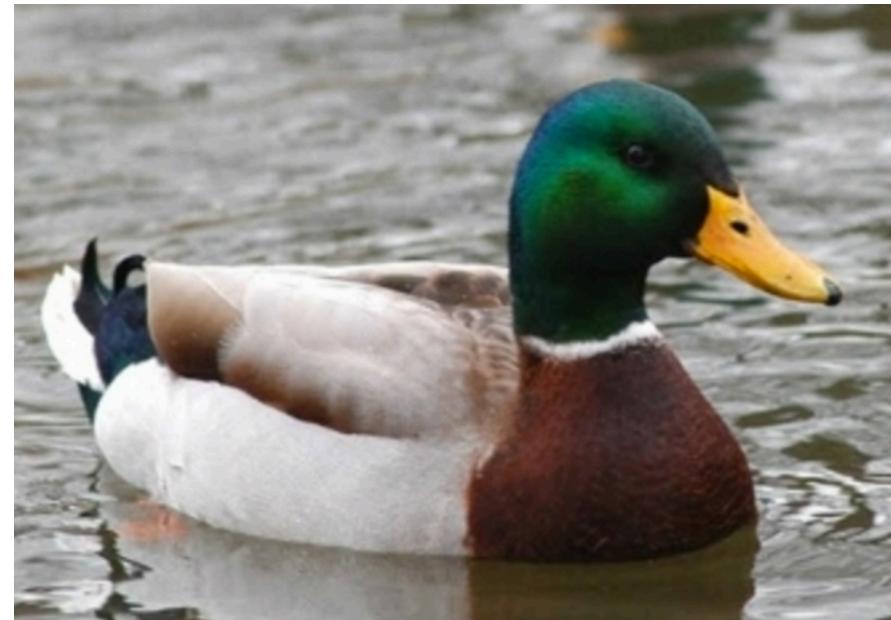
Estrutura também conhecida como **laço de repetição** ou “loop”: o bloco de comandos é sequencialmente repetido **tantas vezes** quanto o teste da condição for **verdadeiro**.

Somente quando a condição se torna falsa a próxima instrução após o bloco de comandos associado ao while é executada (fim do laço).

# WHILE

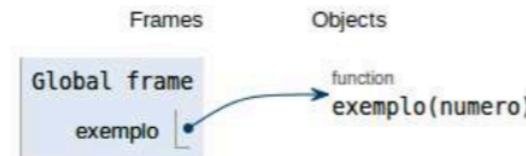
Se a condição da estrutura while já for **falsa** desde o início, o bloco de comandos associado a ela **nunca é executado**.

Deve haver algum processo dentro do bloco de comandos que torne a condição falsa e a repetição seja encerrada, ou um erro GRAVE ocorrerá: sua função **ficará rodando para sempre!!**



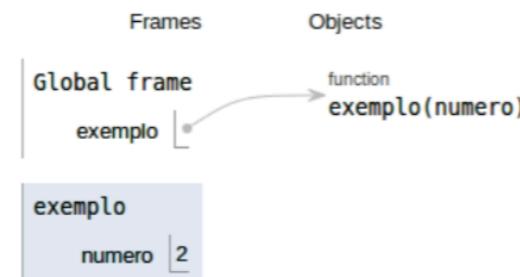
# TESTANDO DE NOVO!

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero < 5:
7         numero = numero - 1
8     return "boom !"
9
→ 10 exemplo(2)
```



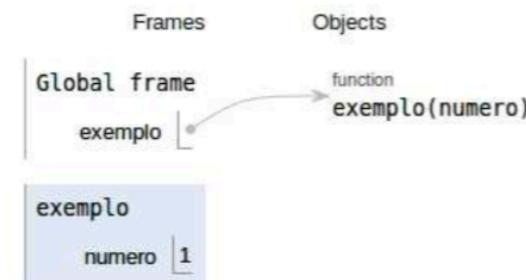
# TÁ TRANQUILO

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero < 5:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(2)
```



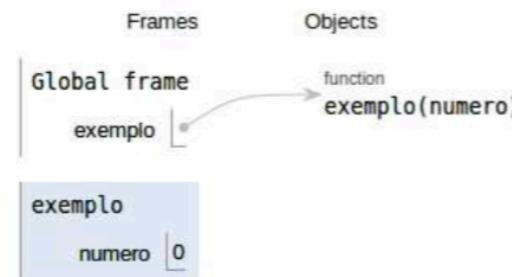
# TÁ FAVORÁVEL

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
6     while numero < 5:
7         numero = numero - 1
8     return "boom !"
9
10 exemplo(2)
```



# NÃO, PERA!

```
1 # função que reduz em 1 o valor do numero passado
2 # parâmetro até chegar a zero.
3 # int -> str
4
5 def exemplo(numero):
→ 6     while numero < 5:
→ 7         numero = numero - 1
8     return "boom !"
9
10 exemplo(2)
```



# OUTRO EXEMPLO

```
# Função que conta quantas vezes se pode reduzir em 1 o valor do número
# passado como parâmetro até chegar a zero.
# int → str
def exemplo1(numero):
    contador = 0 # variável contadora
    while numero > 0:
        numero = numero - 1
        contador = contador + 1
    return "O programa rodou " + str(contador) + "vezes."
```

# **EXERCÍCIO**

**Faça uma função que determina a soma de todos os números pares desde 100 até 200.**

# EXERCÍCIO

Faça uma função que determina a soma de todos os números pares desde 100 até 200.

```
# Função que calcula a soma dos números pares de 100 a 200
# sem entrada → int
def somaPares():
    soma = 0 # variável acumuladora
    contador = 100 # o contador não precisa começar de zero
    while contador < 200:
        soma = soma + contador
        contador = contador + 2 # o contador não precisa ir de 1 em 1
    return soma
```

# QUAL O RESULTADO DESDE AQUI?

```
# sem entrada → int
def exemplo3():
    x = 10
    while x > 8:
        x = x+ 2
    return x
```



# LOOP INFINITO!



Sendo X igual a 10, o teste  $X > 8$  inicialmente verdadeiro.

Enquanto a condição for verdadeira, apenas o comando  
 $X = X + 2$  será executado. Porém incrementar a variável X não  
altera a validade da condição  $X > 8$ .

Logo, a repetição segue **indefidamente!** (Loop infinito)

# E ESSA AQUI?

```
# int → int
def soma(numero):
    soma = 0
    contador = 0
    while contador < numero:
        soma = soma + contador
        contador = contador + 1
    return soma
```

# **EXERCÍCIO**

**Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.**

**Use a função randint(início,fim) do módulo random para gerar um número aleatório, onde os valores de (início,fim) representam o intervalo desejado para os números a serem gerados.**

Exemplo: `randint(1,10)` → gera um número aleatório entre 1 e 10, inclusive.

# EXERCÍCIO

Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.

```
from random import randint
# sem entrada → int

def somaAleatoria():
    soma = 0
    numero = randint(1,10)
    while numero != 5:
        soma = soma + numero
        numero = randint(1,10)
    return soma
```

# **OUTRO MAIS**

**Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5.**

# OUTRO MAIS

Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5.

```
from random import randint
# sem entrada → int

def soma10():
    soma = 0
    contador = 0
    while contador < 10:
        numero = randint(1,5)
        soma = soma + numero
        contador = contador + 1
    return soma
```

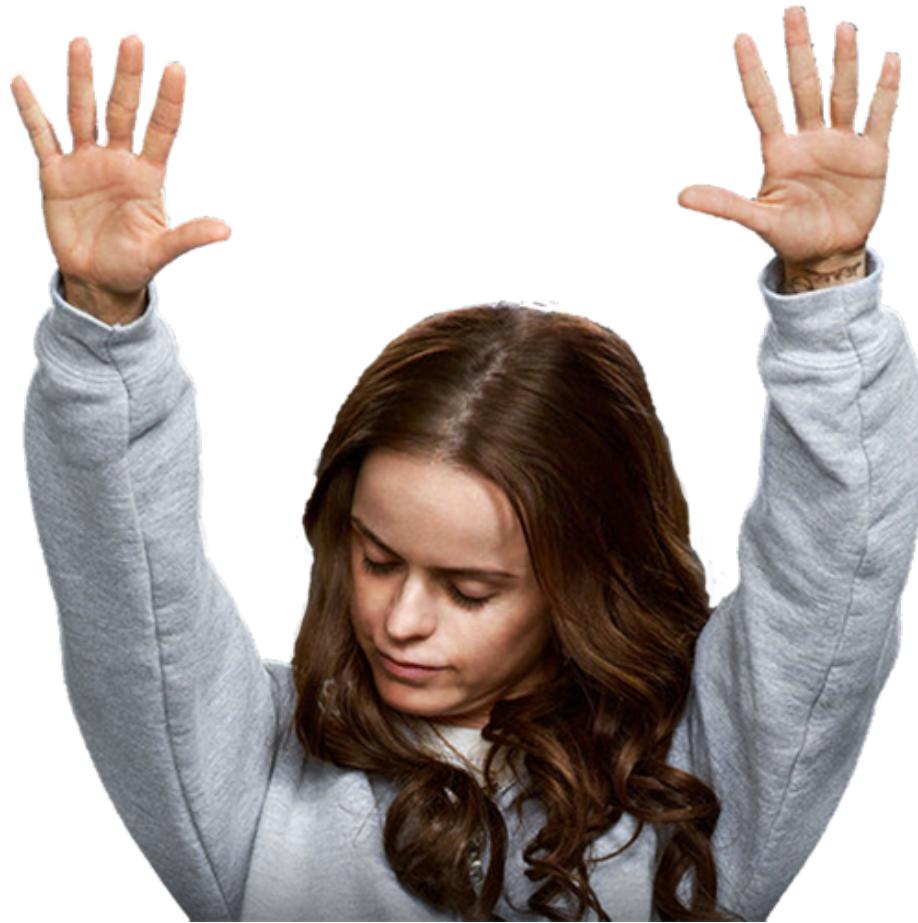
# E OUTROS

Para cada um dos itens abaixo, faça uma tabela mostrando os valores que i, j e n assumem depois de cada execução do laço while.

```
def ...
    i = 0
    j = 10
    n = 0
    while i < j:
        i = i + 1
        j = j - 1
        n = n + 1
```

```
def ...
    i = 0
    j = 0
    n = 0
    while i < 10:
        i = i + 1
        n = n + i + j
        j = j + 1
```

# **LIBERADOS!**



# **COMPUTAÇÃO 1 – PYTHON**

## **AULA 7 TEÓRICA**

**SLIDES BASEADOS NOS TRABALHOS:**  
**AULAS TEÓRICAS DO DCC UFRJ**  
**AULA DO CLAUDIO ESPERANÇA DO PESC**