

# ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

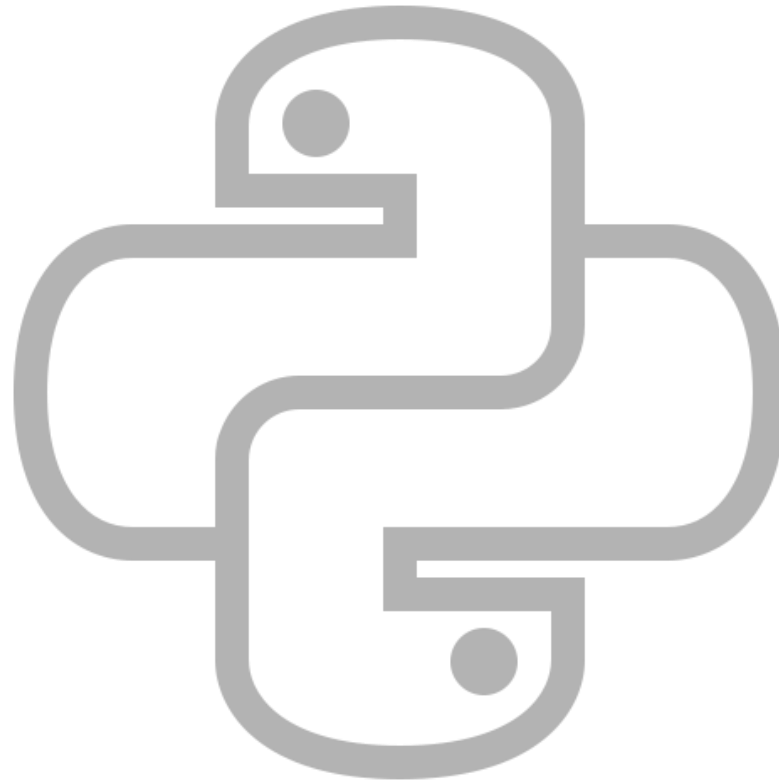
## AULA 7: ESTRUTURAS DE REPETIÇÃO ANINHADAS

# DO QUE VAMOS FALAR

- **1.** Estruturas de repetição aninhadas
- **2.** Exercícios extras



# 1. ESTRUTURAS DE REPETIÇÃO ANINHADAS

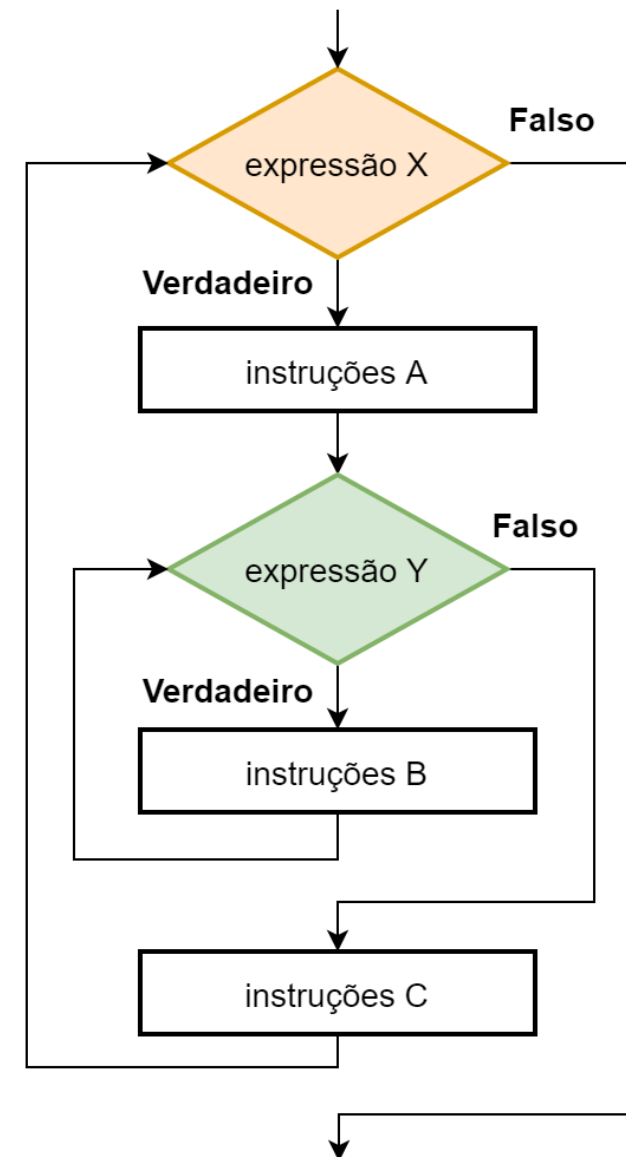


# ESTRUTURAS DE REPETIÇÃO ANINHADAS

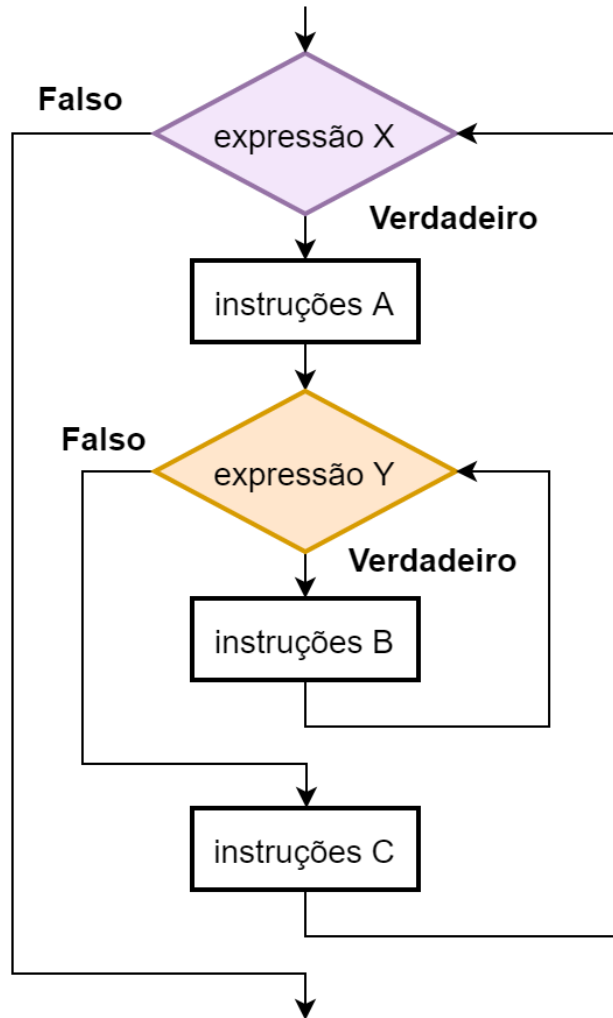
É possível que dentro de uma estrutura de repetição exista outra estrutura de repetição, nestes casos dizemos que a estrutura de repetição interna está **aninhada** em relação à externa.

Na figura, o *loop verde* está aninhado no *loop laranja*.

```
...  
while expressão X:  
    instruções A  
    while expressão Y:  
        instruções B  
    instruções C  
...
```



# ESTRUTURAS DE REPETIÇÃO ANINHADAS



A cada rodada do *loop* mais externo (*expressão X*) o *loop* mais interno (*expressão Y*) executa todas suas rodadas. Dizemos que o *loop* mais interno está aninhado em relação ao mais externo.

```
...  
while expressão X:  
    instruções A  
    while expressão Y:  
        instruções B  
    instruções C  
...
```

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

Uma consequência lógica do aninhamento de estruturas de repetição é que cada rodada do *loop* externo implica em uma execução completa do *loop* interno.

Isso é útil quando precisamos que uma **repetição de instruções** seja executada **várias vezes**.

Um exemplo de repetição de repetições é um relógio onde o ponteiro de segundos executa o mesmo movimento de deslocamento 60 vezes e esse ciclo se repete a cada 1 minuto. O próprio ponteiro de minutos repete seu deslocamento 60 vezes a cada 1 hora.



# ESTRUTURAS DE REPETIÇÃO ANINHADAS

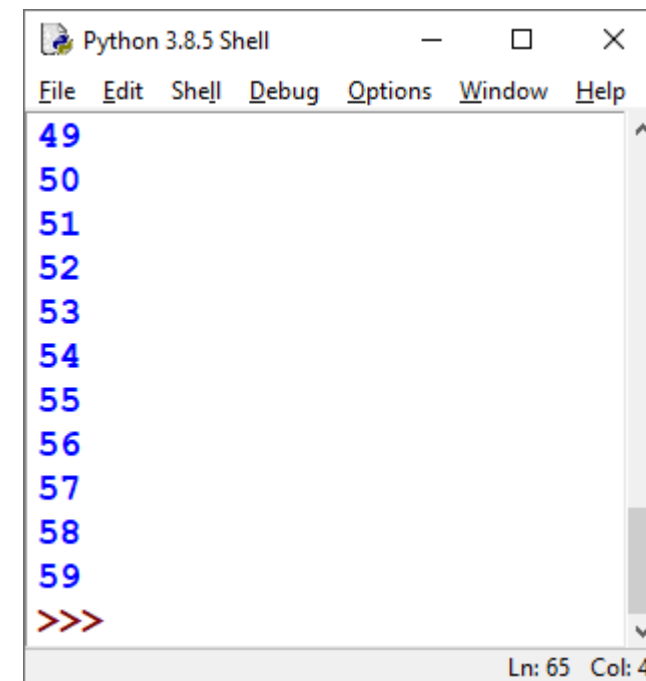
**[EXEMPLO 1]** Crie um programa que exiba todos os segundos de um minuto. O primeiro segundo exibido será 0 e o último 59.

```
s = 0
while s < 60:
    print(s)
    s += 1
```

VERSÃO WHILE

```
for s in range(60):
    print(s)
```

VERSÃO FOR



The screenshot shows a terminal window titled "Python 3.8.5 Shell". The output consists of the numbers 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, and 59, each on a new line. At the bottom of the window, the prompt ">>>" is visible. The status bar at the bottom right indicates "Ln: 65 Col: 4".

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

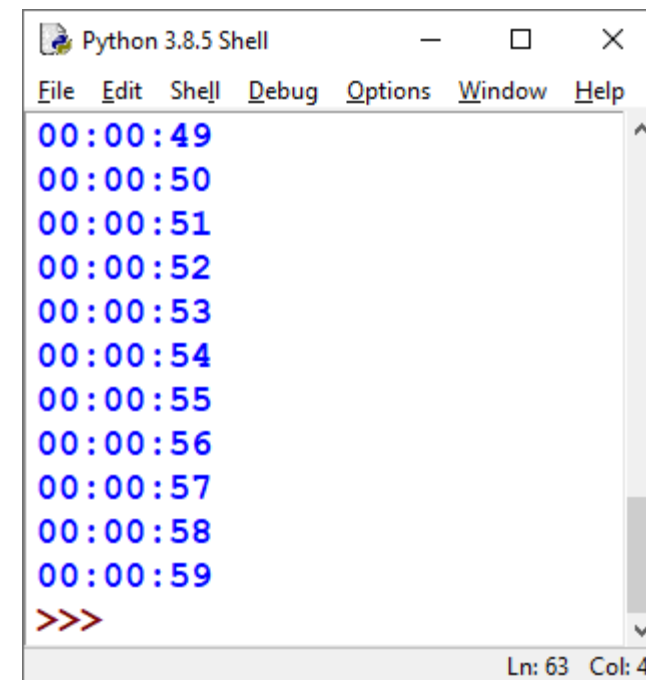
**[EXEMPLO 2]** Altere o programa do Exemplo 1 de modo que os segundos sejam exibidos no formato de um relógio digital, ou seja, hh:mm:ss.

```
s = 0
while s < 60:
    print('00:00:%02d' % s)
    s += 1
```

VERSÃO WHILE

```
for s in range(60):
    print('00:00:%02d' % s)
```

VERSÃO FOR



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
00:00:49
00:00:50
00:00:51
00:00:52
00:00:53
00:00:54
00:00:55
00:00:56
00:00:57
00:00:58
00:00:59
>>>
Ln: 63 Col: 4
```



# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 3]** Altere o programa do Exemplo 2 para que use a função `sleep()` da biblioteca `time` e aguarde 1 segundo entre as exibições.

```
from time import sleep

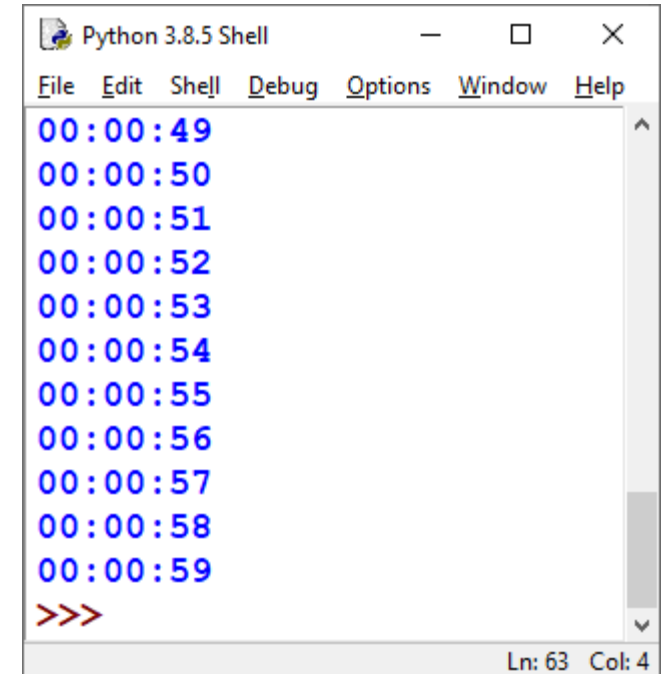
s = 0
while s < 60:
    print('00:00:%02d' % s)
    s += 1
    sleep(1)
```

VERSÃO WHILE

```
from time import sleep

for s in range(60):
    print('00:00:%02d' % s)
    sleep(1)
```

VERSÃO FOR



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
00:00:49
00:00:50
00:00:51
00:00:52
00:00:53
00:00:54
00:00:55
00:00:56
00:00:57
00:00:58
00:00:59
>>>
Ln: 63 Col: 4
```

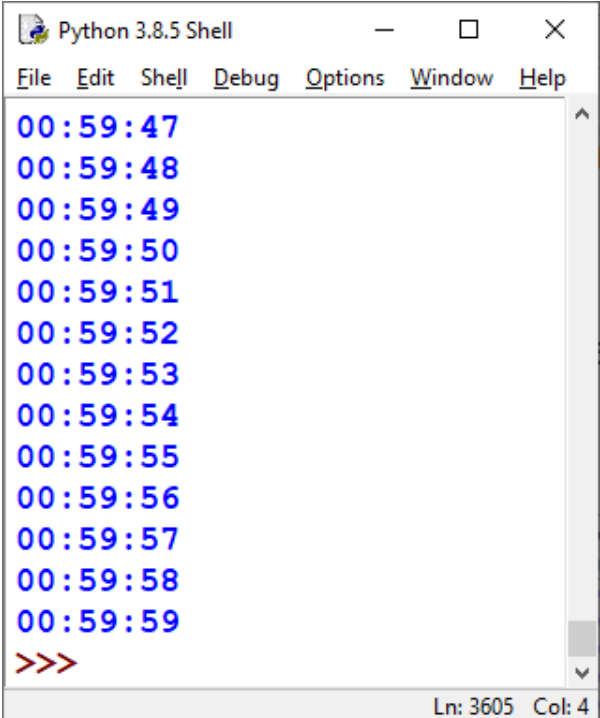
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 4]** Altere o programa do Exemplo 3 de modo que a cada 60 segundos o mostrador de minutos seja incrementado e o de segundos zerado. Repita o procedimento até completar 00:59:59.

```
from time import sleep

m = 0
while m < 60:
    s = 0
    while s < 60:
        print('00:%02d:%02d' % (m, s))
        s += 1
        sleep(1)
    m += 1
```

VERSÃO WHILE



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
00:59:47
00:59:48
00:59:49
00:59:50
00:59:51
00:59:52
00:59:53
00:59:54
00:59:55
00:59:56
00:59:57
00:59:58
00:59:59
>>>
Ln: 3605 Col: 4
```

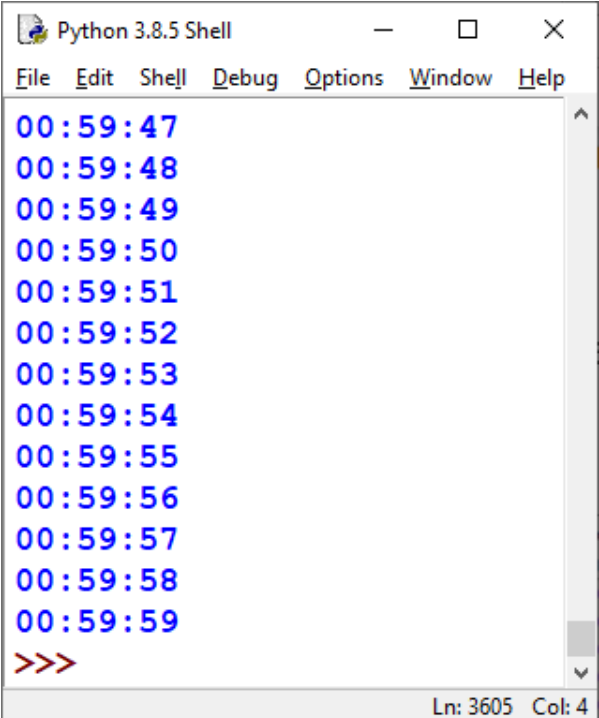
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 4]** Altere o programa do Exemplo 3 de modo que a cada 60 segundos o mostrador de minutos seja incrementado e o de segundos zerado. Repita o procedimento até completar 00:59:59.

```
from time import sleep

for m in range(60):
    for s in range(60):
        print('00:%02d:%02d' % (m, s))
        sleep(1)
```

VERSÃO FOR



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
00:59:47
00:59:48
00:59:49
00:59:50
00:59:51
00:59:52
00:59:53
00:59:54
00:59:55
00:59:56
00:59:57
00:59:58
00:59:59
>>>
Ln: 3605 Col: 4
```

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 5]** Altere o programa do Exemplo 4 de modo que a cada 60 minutos o mostrador de horas seja incrementado e o de minutos zerado. Repita o procedimento até completar 23:59:59.

```
from time import sleep
```

```
h = 0
```

```
while h < 24:
```

```
    m = 0
```

```
    while m < 60:
```

```
        s = 0
```

```
        while s < 60:
```

```
            print('%02d:%02d:%02d' % (h, m, s))
```

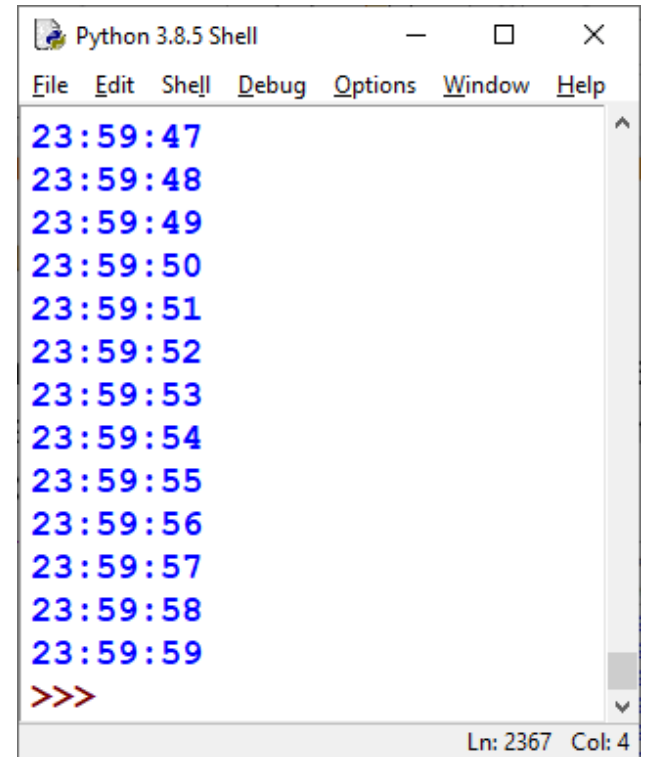
```
            s += 1
```

```
            sleep(1)
```

```
        m += 1
```

```
    h += 1
```

VERSÃO WHILE



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
23:59:47
23:59:48
23:59:49
23:59:50
23:59:51
23:59:52
23:59:53
23:59:54
23:59:55
23:59:56
23:59:57
23:59:58
23:59:59
>>>
Ln: 2367 Col: 4
```

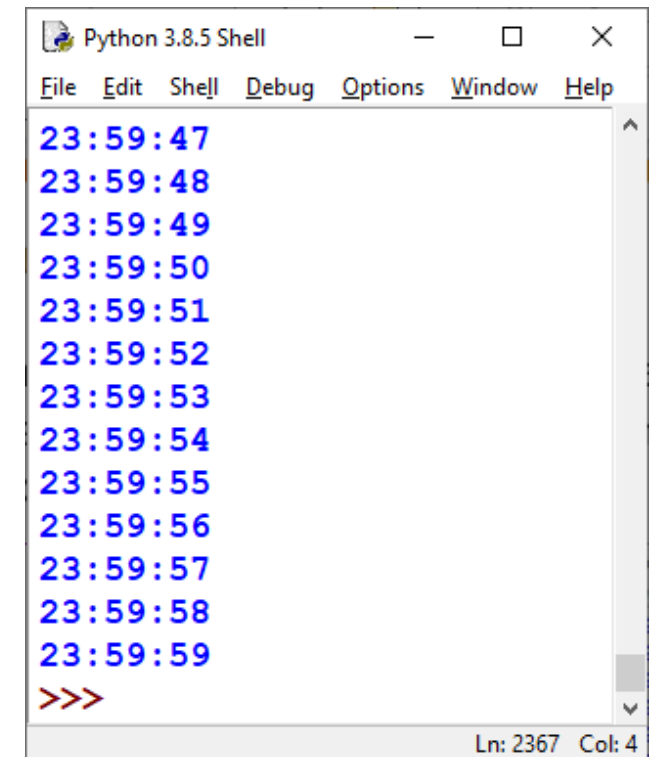
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 5]** Altere o programa do Exemplo 4 de modo que a cada 60 minutos o mostrador de horas seja incrementado e o de minutos zerado. Repita o procedimento até completar 23:59:59.

```
from time import sleep

for h in range(24):
    for m in range(60):
        for s in range(60):
            print('%02d:%02d:%02d' % (h, m, s))
            sleep(1)
```

VERSÃO FOR



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
23:59:47
23:59:48
23:59:49
23:59:50
23:59:51
23:59:52
23:59:53
23:59:54
23:59:55
23:59:56
23:59:57
23:59:58
23:59:59
>>>
Ln: 2367 Col: 4
```

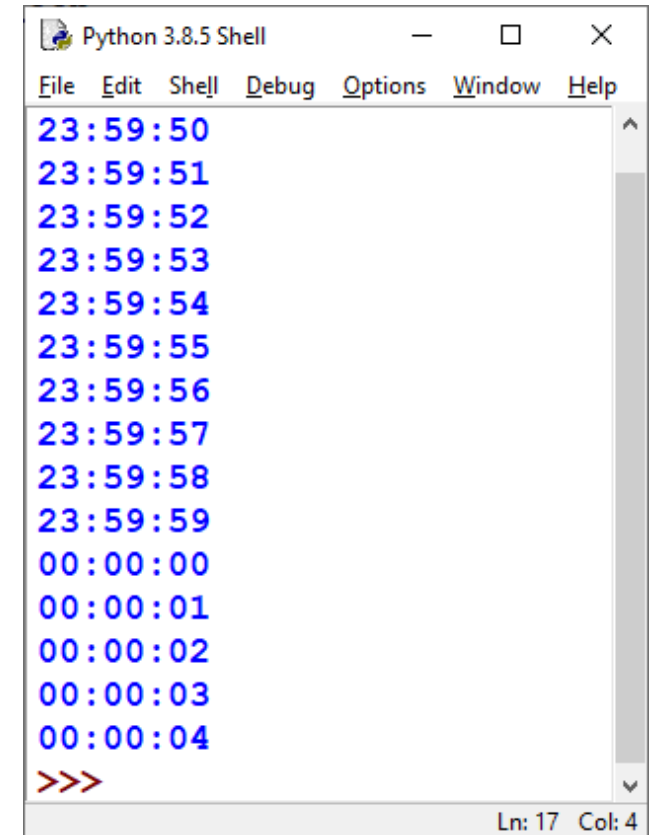
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 6]** Altere o programa do Exemplo 5 de modo que a cada 24 horas os três mostradores sejam zerados e o relógio reinicie a contagem.

```
from time import sleep

while True:
    h = 0
    while h < 24:
        m = 0
        while m < 60:
            s = 0
            while s < 60:
                print('%02d:%02d:%02d' % (h, m, s))
                s += 1
                sleep(1)
            m += 1
        h += 1
```

VERSÃO WHILE



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
23:59:50
23:59:51
23:59:52
23:59:53
23:59:54
23:59:55
23:59:56
23:59:57
23:59:58
23:59:59
00:00:00
00:00:01
00:00:02
00:00:03
00:00:04
>>>
Ln: 17 Col: 4
```

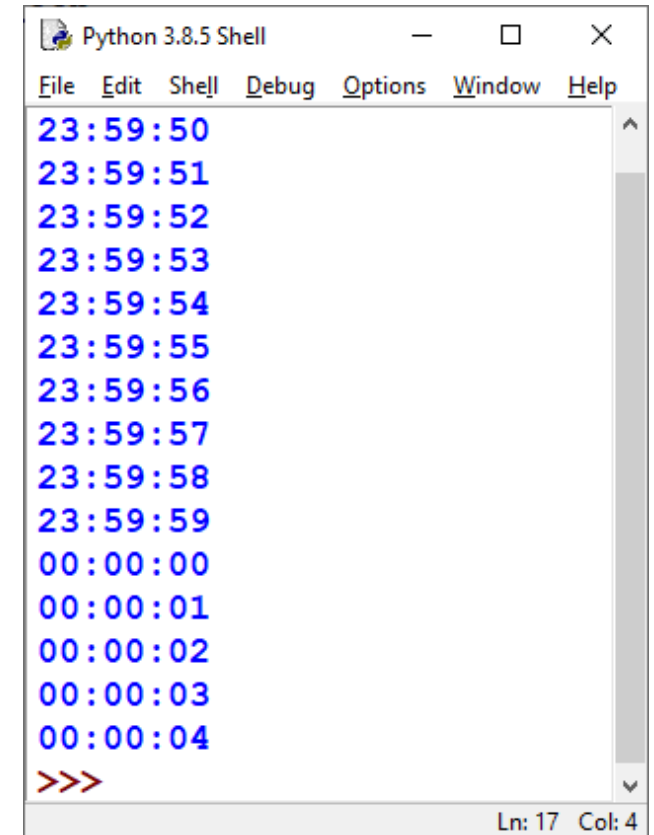
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 6]** Altere o programa do Exemplo 5 de modo que a cada 24 horas os três mostradores sejam zerados e o relógio reinicie a contagem.

```
from time import sleep

while True:
    for h in range(24):
        for m in range(60):
            for s in range(60):
                print('%02d:%02d:%02d' % (h, m, s))
                sleep(1)
```

VERSÃO FOR



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
23:59:50
23:59:51
23:59:52
23:59:53
23:59:54
23:59:55
23:59:56
23:59:57
23:59:58
23:59:59
00:00:00
00:00:01
00:00:02
00:00:03
00:00:04
>>>
```

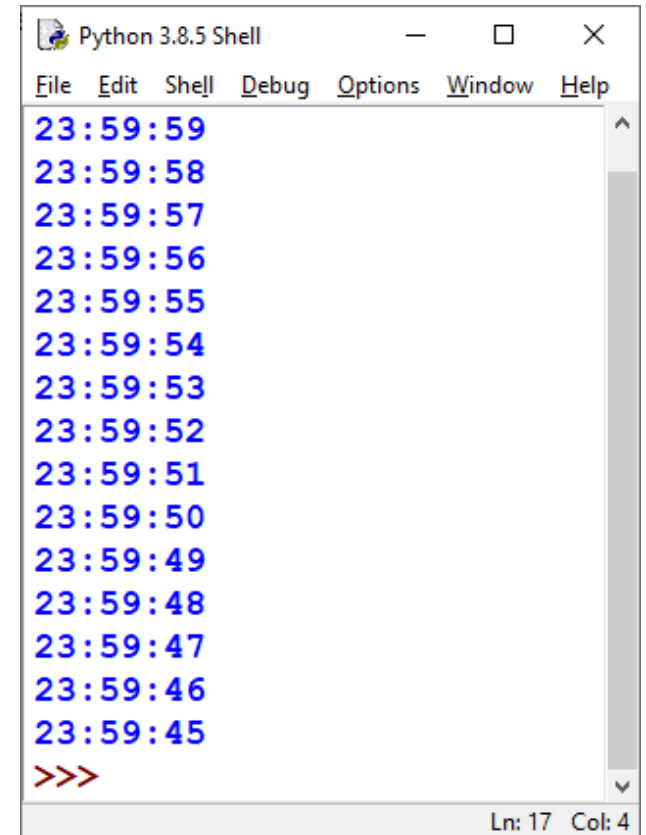
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 7]** Altere o programa do Exemplo 6 de modo que o mostrador inicie em 23:59:59 e a contagem de tempo seja decrescente.

```
from time import sleep

while True:
    h = 23
    while h >= 0:
        m = 59
        while m >= 0:
            s = 59
            while s >= 0:
                print('%02d:%02d:%02d' % (h, m, s))
                s -= 1
                sleep(1)
            m -= 1
        h -= 1
```

VERSÃO WHILE



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
23:59:59
23:59:58
23:59:57
23:59:56
23:59:55
23:59:54
23:59:53
23:59:52
23:59:51
23:59:50
23:59:49
23:59:48
23:59:47
23:59:46
23:59:45
>>>
Ln: 17 Col: 4
```



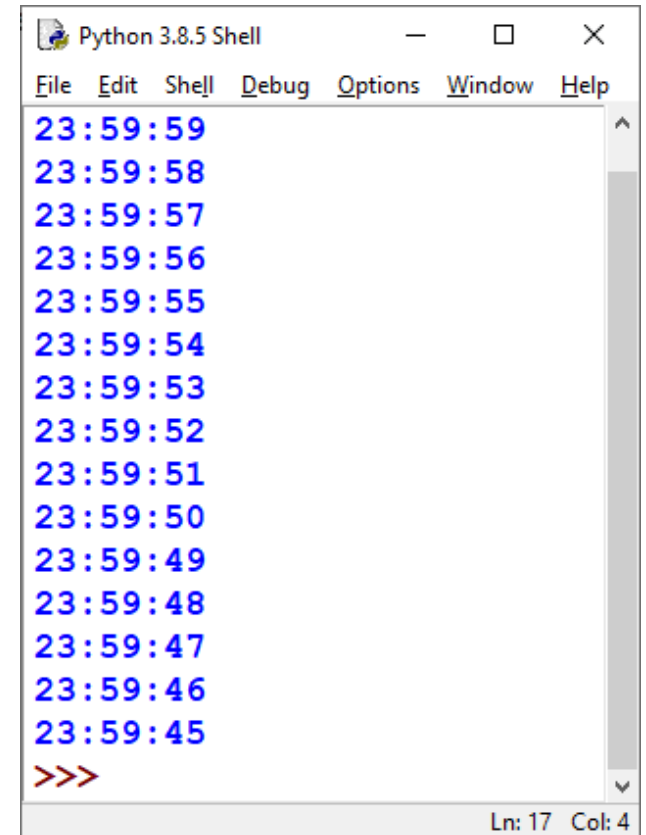
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 7]** Altere o programa do Exemplo 6 de modo que o mostrador inicie em 23:59:59 e a contagem de tempo seja decrescente.

```
from time import sleep

while True:
    for h in range(23, -1, -1):
        for m in range(59, -1, -1):
            for s in range(59, -1, -1):
                print('%02d:%02d:%02d' % (h, m, s))
                sleep(1)
```

VERSÃO FOR

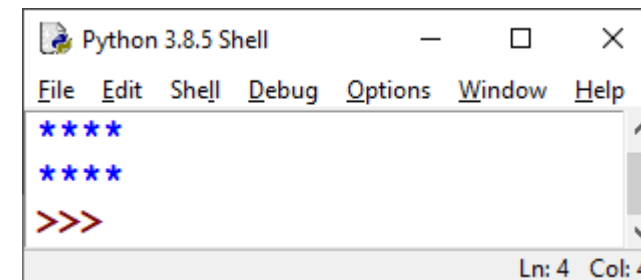


```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
23:59:59
23:59:58
23:59:57
23:59:56
23:59:55
23:59:54
23:59:53
23:59:52
23:59:51
23:59:50
23:59:49
23:59:48
23:59:47
23:59:46
23:59:45
>>>
Ln: 17 Col: 4
```

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXEMPLO 8]** Crie um programa que exiba 2 linhas com 4 colunas do caractere ' \* '.

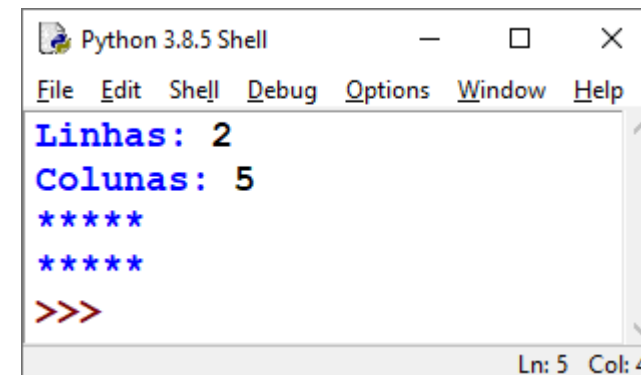
```
for i in range(2):  
    for j in range(4):  
        print(' * ', end=' ')  
    print()
```



```
Python 3.8.5 Shell  
File Edit Shell Debug Options Window Help  
****  
****  
>>>  
Ln: 4 Col: 4
```

**[EXERCÍCIO 1]** Crie um programa que leia dois naturais  $L$  e  $C$  e exiba  $L$  linhas com  $C$  colunas do caractere ' \* '.

```
L = int(input('Linhas: '))  
C = int(input('Colunas: '))  
for i in range(L):  
    for j in range(C):  
        print(' * ', end=' ')  
    print()
```

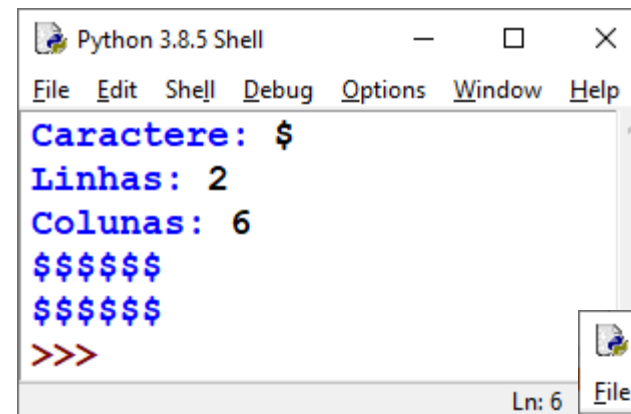


```
Python 3.8.5 Shell  
File Edit Shell Debug Options Window Help  
Linhas: 2  
Colunas: 5  
*****  
*****  
>>>  
Ln: 5 Col: 4
```

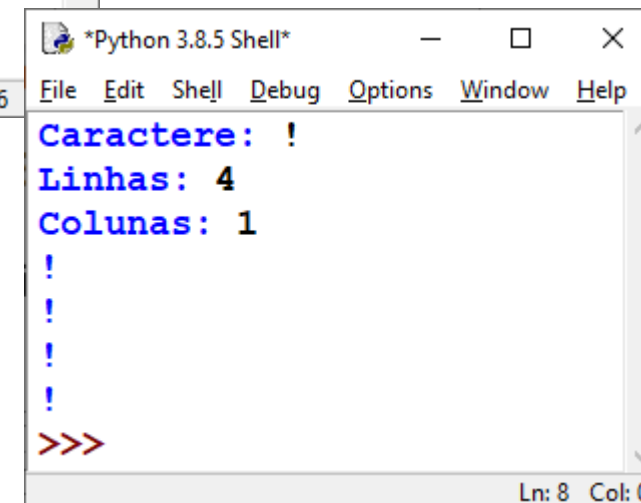
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXERCÍCIO 2]** Crie um programa que leia um caractere  $x$  e dois naturais  $L$  e  $C$ , o programa deve exibir  $L$  linhas com  $C$  colunas do caractere  $x$ .

```
x = input('Caractere: ')
L = int(input('Linhas: '))
C = int(input('Colunas: '))
for i in range(L):
    for j in range(C):
        print(x, end=' ')
    print()
```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Caractere: $
Linhas: 2
Colunas: 6
$$$$$$
$$$$$$
>>>
```

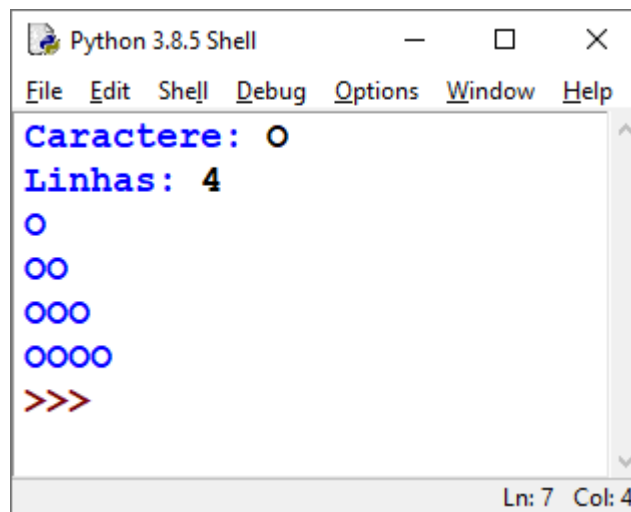


```
*Python 3.8.5 Shell*
File Edit Shell Debug Options Window Help
Caractere: !
Linhas: 4
Colunas: 1
!
!
!
!
>>>
```

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXERCÍCIO 3]** Crie um programa que leia um caractere  $x$  e um natural  $L$ , o programa deve exibir  $L$  linhas do caractere  $x$ . A quantidade de caracteres  $x$  por linha será crescente de 1 à  $L$ .

```
x = input('Caractere: ')
L = int(input('Linhas: '))
for i in range(L):
    for j in range(i+1):
        print(x, end=' ')
    print()
```



Python 3.8.5 Shell

File Edit Shell Debug Options Window Help

Caractere: o

Linhas: 4

o

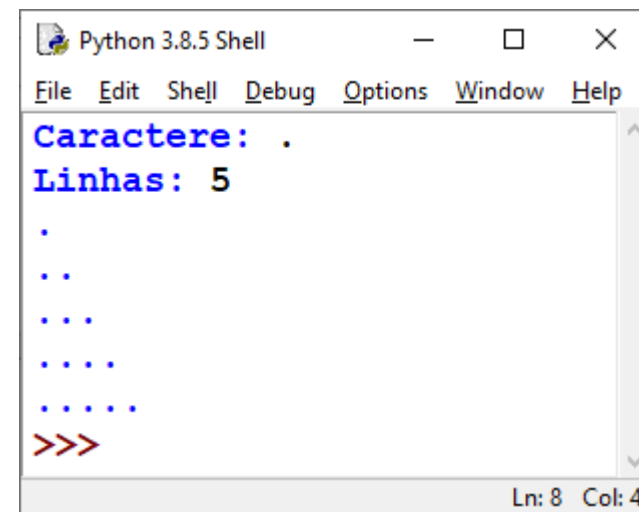
oo

ooo

oooo

>>>

Ln: 7 Col: 4



Python 3.8.5 Shell

File Edit Shell Debug Options Window Help

Caractere: .

Linhas: 5

.

..

...

....

.....

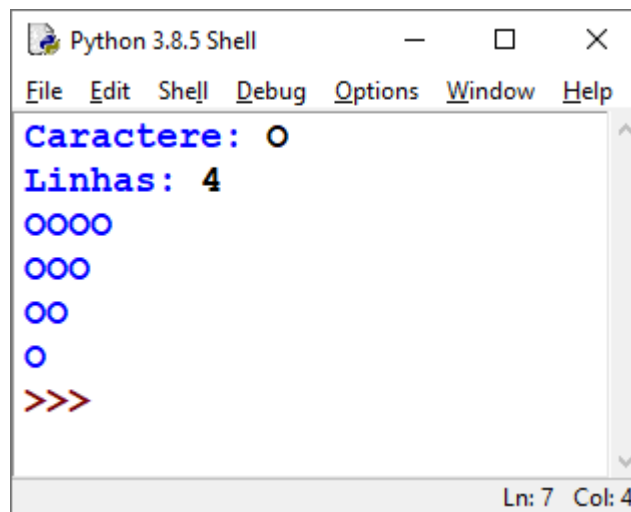
>>>

Ln: 8 Col: 4

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXERCÍCIO 4]** Crie um programa que leia um caractere  $x$  e um natural  $L$ , o programa deve exibir  $L$  linhas do caractere  $x$ . A quantidade de caracteres  $x$  por linha será decrescente de  $L$  à 1.

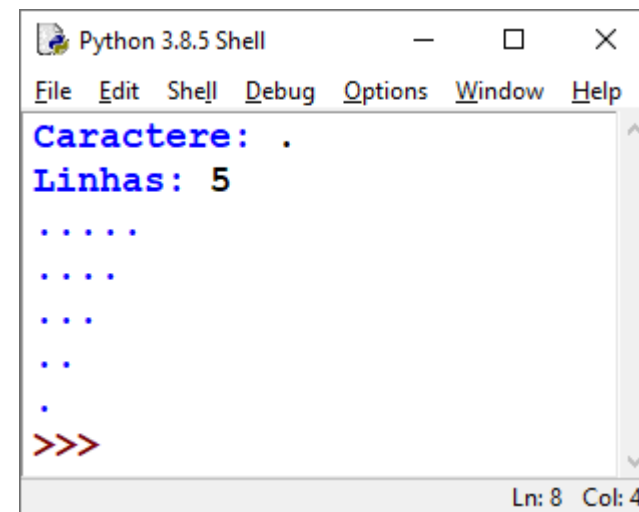
```
x = input('Caractere: ')
L = int(input('Linhas: '))
for i in range(L):
    for j in range(L-i):
        print(x, end=' ')
    print()
```



Python 3.8.5 Shell

```
File Edit Shell Debug Options Window Help
Caractere: o
Linhas: 4
oooo
ooo
oo
o
>>>
```

Ln: 7 Col: 4



Python 3.8.5 Shell

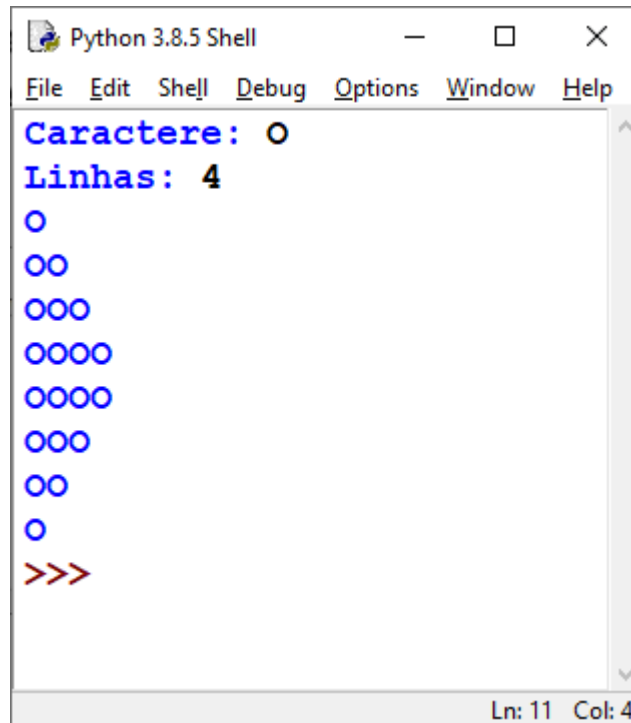
```
File Edit Shell Debug Options Window Help
Caractere: .
Linhas: 5
.....
....
...
..
.
>>>
```

Ln: 8 Col: 4

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

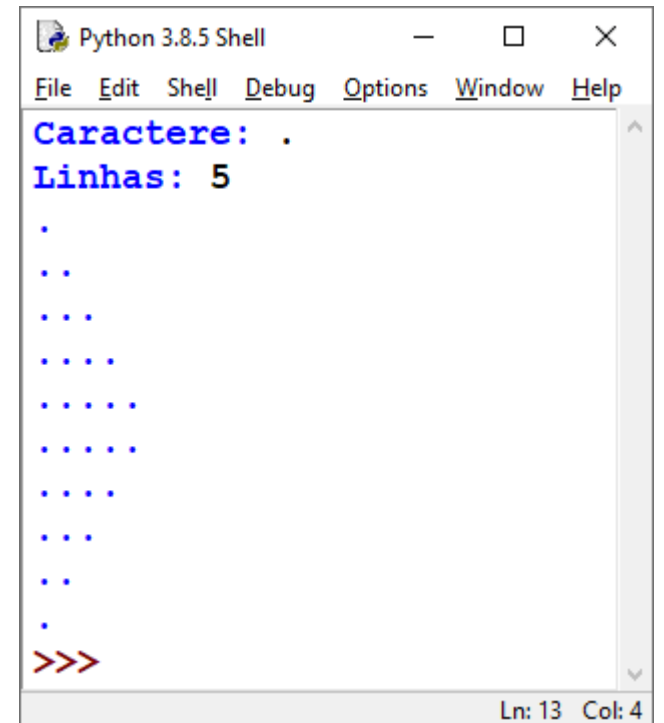
**[EXERCÍCIO 5]** Crie um programa que leia um caractere  $x$  e um natural  $L$ , o programa deve exibir  $2 * L$  linhas do caractere  $x$ . A quantidade de caracteres  $x$  por linha será crescente de 1 à  $L$  e, em seguida, decrescente de  $L$  à 1.

```
x = input('Caractere: ')
L = int(input('Linhas: '))
for i in range(L):
    for j in range(i+1):
        print(x, end='')
    print()
for i in range(L):
    for j in range(L-i):
        print(x, end='')
    print()
```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Caractere: o
Linhas: 4
o
oo
ooo
oooo
oooo
ooo
oo
o
>>>
```

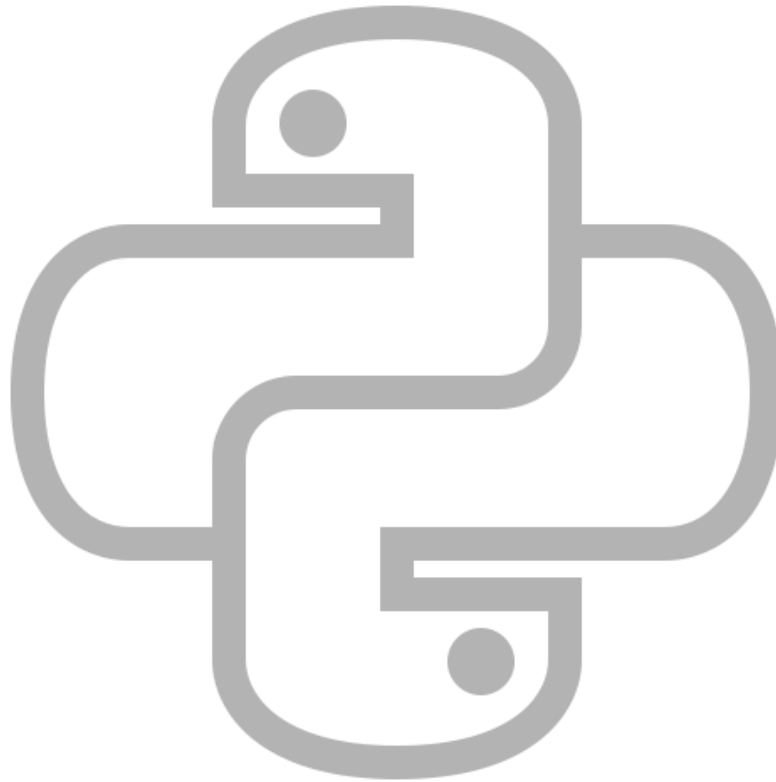
Ln: 11 Col: 4



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Caractere: .
Linhas: 5
.
..
...
....
.....
.....
....
...
..
.
>>>
```

Ln: 13 Col: 4

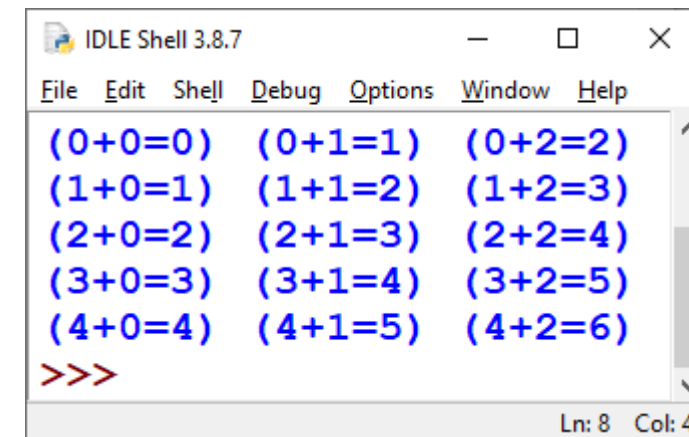
## 2. EXERCÍCIOS EXTRAS



# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXTRA 1]** Crie um programa que exiba 5 linhas com 3 colunas, como se fosse uma matriz em que os itens são a soma de suas próprias coordenadas (linha,coluna).

```
for i in range(5):  
    for j in range(3):  
        print(' (%d+%d=%d) ' % (i,j,i+j), end=' ' )  
    print()
```



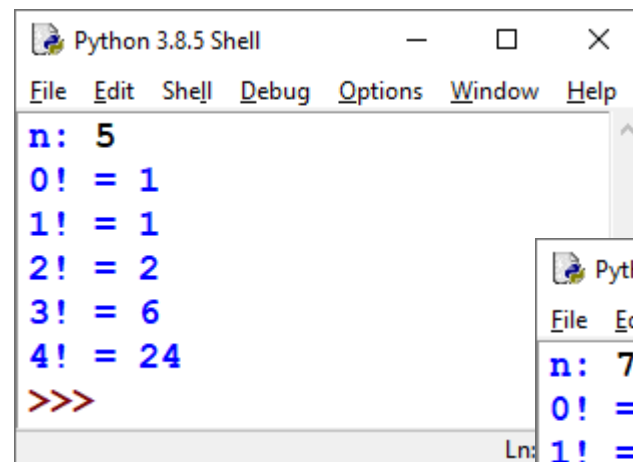
```
IDLE Shell 3.8.7  
File Edit Shell Debug Options Window Help  
(0+0=0) (0+1=1) (0+2=2)  
(1+0=1) (1+1=2) (1+2=3)  
(2+0=2) (2+1=3) (2+2=4)  
(3+0=3) (3+1=4) (3+2=5)  
(4+0=4) (4+1=5) (4+2=6)  
>>>  
Ln: 8 Col: 4
```



# ESTRUTURAS DE REPETIÇÃO ANINHADAS

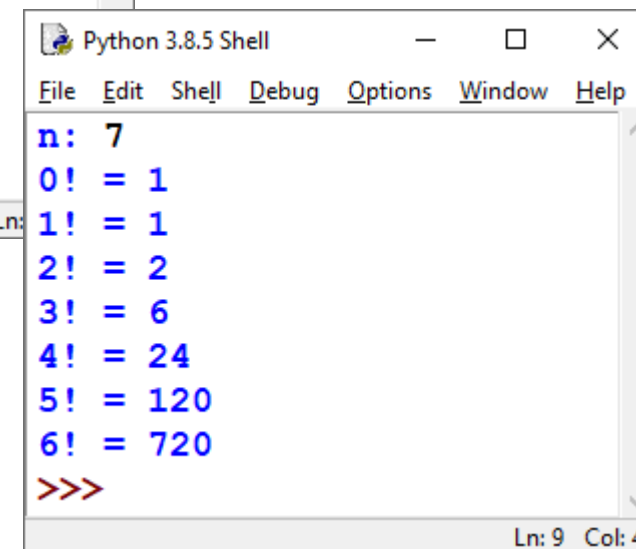
**[EXTRA 2]** Crie um programa que leia um natural  $n$  e exiba os fatoriais dos  $n$  primeiros naturais.

```
n = int(input('n: '))
for x in range(n):
    f = 1
    for i in range(1, x+1):
        f = f * i
    print('%d! = %d' % (x, f))
```



A screenshot of a Python 3.8.5 Shell window. The window title is "Python 3.8.5 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The output shows the program running for n=5, displaying factorials from 0! to 4!. The prompt ">>>" is visible at the bottom.

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
n: 5
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
>>>
```



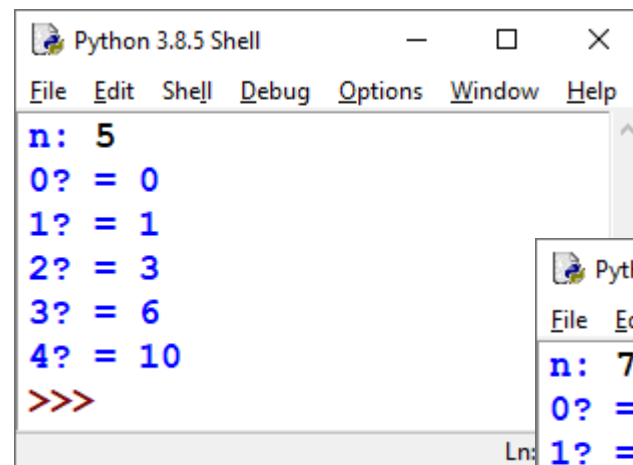
A screenshot of a Python 3.8.5 Shell window. The window title is "Python 3.8.5 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The output shows the program running for n=7, displaying factorials from 0! to 6!. The prompt ">>>" is visible at the bottom. The status bar at the bottom right shows "Ln: 9 Col: 4".

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
n: 7
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
>>>
Ln: 9 Col: 4
```

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

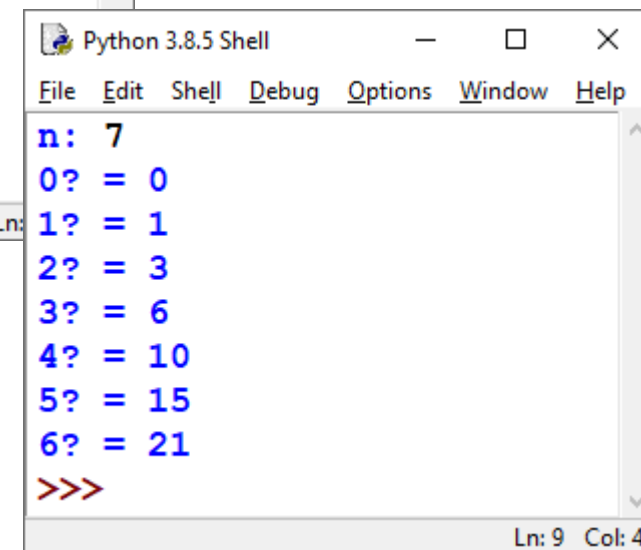
**[EXTRA 3]** Crie um programa que leia um natural  $n$  e exiba os termiais dos  $n$  primeiros naturais.

```
n = int(input('n: '))
for x in range(n):
    t = 0
    for i in range(1, x+1):
        t = t + i
    print('%d? = %d' % (x, t))
```



A screenshot of a Python 3.8.5 Shell window. The window title is "Python 3.8.5 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The output shows the program running for n=5, displaying the sum of the first x natural numbers for x from 0 to 4. The prompt ">>>" is visible at the bottom.

```
n: 5
0? = 0
1? = 1
2? = 3
3? = 6
4? = 10
>>>
```



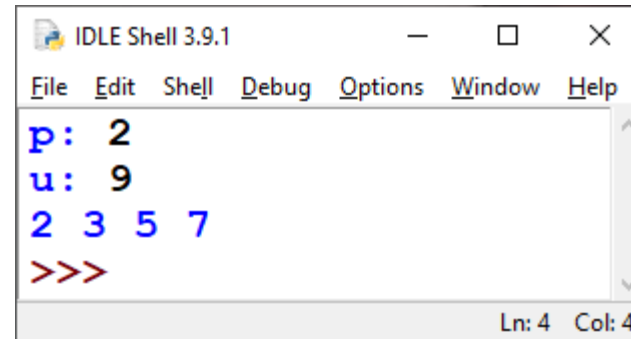
A screenshot of a Python 3.8.5 Shell window. The window title is "Python 3.8.5 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The output shows the program running for n=7, displaying the sum of the first x natural numbers for x from 0 to 6. The prompt ">>>" is visible at the bottom. The status bar at the bottom right shows "Ln: 9 Col: 4".

```
n: 7
0? = 0
1? = 1
2? = 3
3? = 6
4? = 10
5? = 15
6? = 21
>>>
```

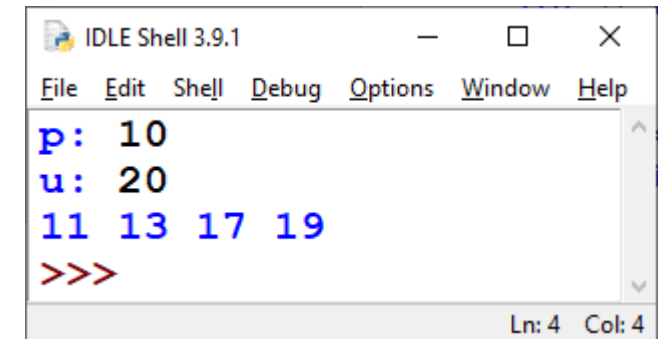
# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXTRA 4]** Crie um programa que leia dois naturais  $p$  e  $u$  ( $1 < p \leq u$ ) e exiba todos os números primos do intervalo  $[p..u]$ .

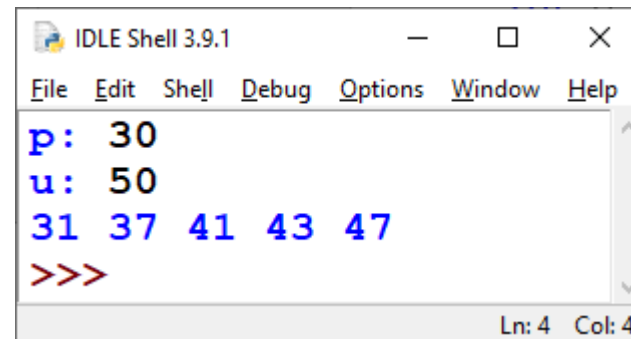
```
p = int(input('p: '))
u = int(input('u: '))
for n in range(p, u+1):
    qtd = 0
    for d in range(1, n+1):
        if n%d == 0:
            qtd += 1
    if qtd==2:
        print(n, end=' ')
```



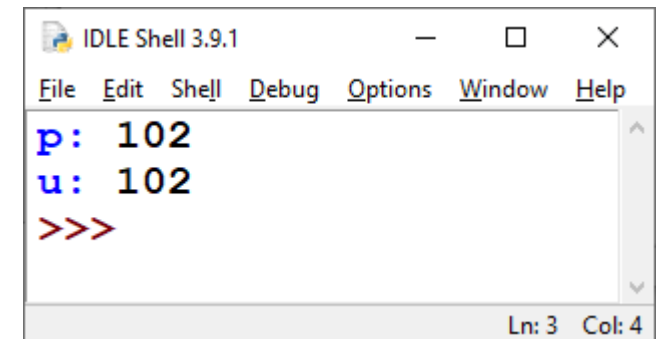
```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
p: 2
u: 9
2 3 5 7
>>>
Ln: 4 Col: 4
```



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
p: 10
u: 20
11 13 17 19
>>>
Ln: 4 Col: 4
```



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
p: 30
u: 50
31 37 41 43 47
>>>
Ln: 4 Col: 4
```

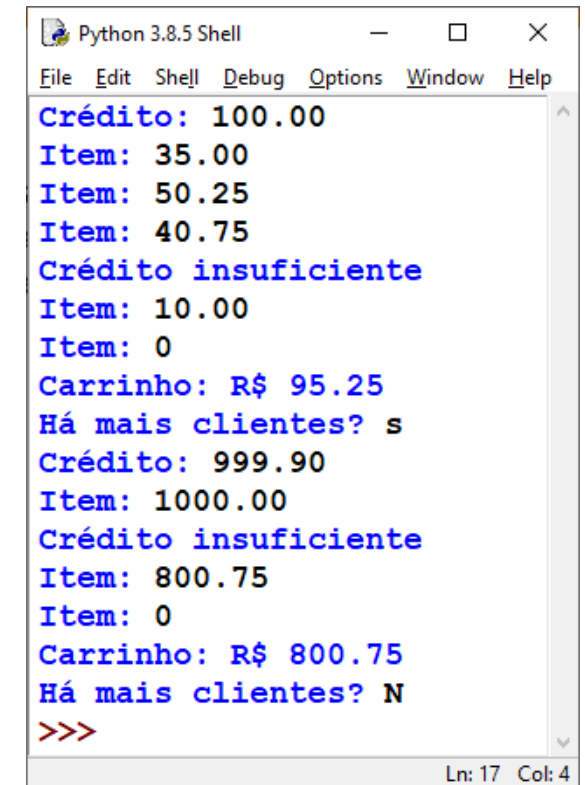


```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
p: 102
u: 102
>>>
Ln: 3 Col: 4
```

# ESTRUTURAS DE REPETIÇÃO ANINHADAS

**[EXTRA 5]** Crie um programa para um site de compras que solicita aos clientes seus créditos e verifica, a cada item comprado, se há crédito suficiente para incluir o item no carrinho. A compra de cada cliente será finalizada quando inserido um item com valor zero.

```
while True:
    credito = float(input('Crédito: '))
    carrinho = 0.0
    item = float(input('Item: '))
    while item != 0.0:
        if carrinho+item > credito:
            print('Crédito insuficiente')
        else:
            carrinho += item
        item = float(input('Item: '))
    print('Carrinho: R$ %.2f' % carrinho)
    clientes = input('\nHá mais clientes? ')
    if clientes=='N': break
```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Crédito: 100.00
Item: 35.00
Item: 50.25
Item: 40.75
Crédito insuficiente
Item: 10.00
Item: 0
Carrinho: R$ 95.25
Há mais clientes? s
Crédito: 999.90
Item: 1000.00
Crédito insuficiente
Item: 800.75
Item: 0
Carrinho: R$ 800.75
Há mais clientes? N
>>>
```