

Exercícios – Lista XIV – Revisão Geral – Python para Zumbis

Imprima e resolva no papel (baseado no curso 6.189 do MIT).

Exemplo de programa:

```
print ('x', end = ' ')\nprint ('x', end = ' ')
```

Saída (colocar que dá erro se for o caso):

x x

Variáveis, operadores e expressões

As variáveis possuem um tipo, que pode ser verificado através da função *type(x)*. Você pode converter dados por meio de funções: *int(x)*, *str(x)*, *float(x)*, *bool(x)*. Elas darão erro algumas vezes quando não houver sentido na conversão, por exemplo, *int("abacate")*.

Programa

```
a = 5\nb = a + 7\na = 10\nprint (b)
```

Saída

Programa

```
print (type(0))
```

```
print (type(0.0))
```

```
print (type(3.14))
```

```
print (type('Py'))
```

```
print (type(True))
```

```
print (type(1/2))
```

```
print (type(1//2))
```

```
print (type(2//1))
```

```
print (type(3**3))
```

```
print (type(0==0))
```

```
print (type(3<0))
```

```
print (type(3!=3))
```

Programa

```
print (type(str(int(3.14159265358979))))
```

Saída

Programa

```
print (3 == 3.0)
```

```
print (1/3)
```

```
print (1//3)
```

```
print (3 == '3')
```

```
print ('x' != 'x')
```

```
print (2/1)
```

```
print (2//1)
```

```
print (not False)
```

```
print (not True)
```

```
print (not 0)
```

Programa

```
print (True and (False or not True))
```

Saída

Programa

```
a = 20\nprint (15-(a-15), end = ' ')\na = 10\nprint (15-(a-15), end = ' ')
```

Saída

Programa

```
a = 12.75\nprint (a - int(a), end = ' ')\na = int((a - int(a))*100)\nprint (a)
```

Saída

Programa

```
a = 3\nb = 4\na = a + b\nb = a - b\na = a - b\nprint (a, b)
```

Saída

Programa

```
print (3 % 2)
```

```
print (0 % 2)
```

```
print (123%356254)
```

Saída

Programa

```
print (type([1, 2]))
```

```
print (type({1:2}))
```

```
print (type([]))
```

Saída

Programa

```
a = 'abacate'
print ('e' in a, 'x' in a, end = ' ')
print ('ate' in a, end = ' ')
print ('' in a, end = ' ')
print ('eta' in a, end = ' ')
print ('eta' not in a)
```

Saída

Programa

```
a = '0123456789'
print (a[0], a[3], a[-1], end = ' ')
print (a[0:3], a[3:6], a[6:9], end = ' ')
print (a[:3], a[7:], end = ' ')
print (a[:9:2], end = ' ')
print (a[::-1])
```

Saída

Programa

```
a = [1, 2, [3, 4]]
print (1 in a, end = ' ')
print ([1, 2] in a, end = ' ')
print ([3, 4] in a, end = ' ')
print (3 in a, end = ' ')
print (3 in a[2], end = ' ')
print (5 not in a)
```

Saída

Programa

```
a = {1: 'ab', 2: 'cd', 'x':3.14}
print (1 in a, 3 in a, end = ' ')
print ('x' in a, 'z' in a, end = ' ')
print (a[1], a['x'])
```

Saída

Condicionais if/else/elif

O comando `if` executa um bloco de comandos somente se a condição é `True`. Esta condição pode ser qualquer coisa. Os comandos `else` e `elif` são opcionais que são testadas apenas quando condições anteriores não forem satisfeitas.

Programa

```
a = ?
if a > 10 and a % 6 == 3:
    print ('A', end = ' ')
elif a > 10 and a < 20:
    print ('B', end = ' ')
else:
    print ('C', end = ' ')
```

Dê os valores de `a` que produzem a saída ('N/A' se não houver valor possível para `a`):

Valores de a

Saída

A B

A

B

C

Feliz Natal!

Comandos while/for/break/continue

Os laços `while` e `for` permitem que você repita um bloco de comandos várias vezes. `break` interrompe o laço e `continue` faz a execução voltar ao início do laço.

Programa

```
a = 1
while a < 10:
    print ('X', end = ' ')
```

Saída

Programa

```
a = -1
while a < 3:
    print ('X', end = ' ')
    a = a + 1
```

Saída

Programa

```
while False: print ('X', end = ' ')
```

Saída

Programa

```
a = 5
b = 9
while a <= b:
    print ('X', end = ' ')
    if a % 2 == 0: print ('O', end = ' ')
    a = a + 1
```

Saída

Programa

```
a=1
while a % 7 != 0:
    if a % 2 == 0: print ('O', end = ' ')
    if a == 2: print ('X', end = ' ')
    a=a+1
```

Saída

Cuidado com pequenas mudanças de código...

Programa1

```
repete = True
a=0
b=0
while repete:
    print ('O', end = ' ')
    a=a+5
    b=b+7
    if a + b >= 24:
        repete = False
```

Saída

Programa2

```
repete = True
a=0
b=0
while repete:
    print ('O', end = ' ')
    if a + b >= 24:
        repete = False
    a=a+5
    b=b+7
```

Saída

Programa3

```
repete = True
a=0
b=0
while repete:
    print ('O', end = ' ')
    if a + b > 24:
        repete = False
    a=a+5
    b=b+7
```

Saída

Laços dentro de laços. Determine bem os comandos do bloco de cada laço. **break** e **continue** se aplicam ao laço do seu bloco apenas. Aponte loops infinito caso ocorra.

Programa

```
a=0
while a < 3:
    while True:
        print ('X', end = ' ')
        break
    print ('O', end = ' ')
    a=a+1
```

Saída

Programa

```
a=1
while a < 3:
    while a < 3:
        print ('O', end = ' ')
    a=a+1
```

Saída

Programa

```
a=1
while a < 3:
    if a % 2 == 0:
        b=1
        while b < 3:
            print ('X', end = ' ')
            b=b+1
        print ('O', end = ' ')
    a=a+1
```

Saída

Programa

```
a=1
while a < 3:
    b=1
    while b < 3:
        if a == 2:
            print ('X', end = ' ')
        print ('O', end = ' ')
        b=b+1
    print ('O', end = ' ')
    a=a+1
```

Saída

Programa

```
x = 'abacate'
while x:
    print (x, end = ' ')
    x = x[1:]
```

Saída

Saída

Programa

```
s = 0
for x in [7, 2, -2, 5]:
    s = s + x
print (s)
```

Saída

Programa

```
x = 10
while x:
    x = x - 1
    if x % 2 != 0:
        continue
    print (x, end = ' ')
```

Saída

Programa

```
p = 1
for x in [1, -1, 2, -2]: p = p * x
print (p)
```

Saída

Programa

```
while 1:
    nome = input('Nome:')
    if nome == 'fim': break
    print ('Bom dia ', nome)
```

Saída

Programa

```
p = 1
for x in 'aeiou':
    print (x*3, end = ' ')
```

Saída

Programa

```
L = [1, 2, 3, 4, 5]
for x in range(len(L)):
    L[x] += 1
print (L)
```

Saída

Programa

```
x = 'python'
achou = False
vogal = 'aeiou'
while x and not achou:
    if x[0] in vogal:
        print ('X', end = ' ')
        achou = True
    else:
        x = x[1:]
if not achou:
    print ('O', end = ' ')
```

Saída

Programa

```
for x in 'abc':
    for y in '012':
        print (x + y, end = ' ')
```

Saída

Programa

```
L = [1, 7, 4, 12, -2]
x = L[0]
while True:
    L = L[1:]
    if not L:
        break
    if L[0] > x:
        x = L[0]
print (x)
```

Saída

O laço **for** executa o bloco de comandos uma vez para cada elemento de uma seqüência.

Programa

```
for x in ['a', 3.14, 7/2]:
    print (x, end = ' ')
```

Uma **função** é uma seqüência de comandos definida com um nome via **def**. Ela pode ter parâmetros e retornar um valor via **return** ou **yield**. Somente é executada quando chamada. **return** e **yield** não são funções, apenas palavras reservadas. Também existem **lambda** funções, mais avançadas.

Programa

```
def f(a):
    a=a+5
    return a

b=0
f(b)
print (b, ',', end = '')
b = f(b)
print (b)
```

Saída

Preencha os quadros segundo a função abaixo

```
def f(x):
    print ('x', end = '')
    if x <= 1:
        return 1
    else:
        return x + f(x-1)
```

Chamada	Valor de retorno	Saída
f(1)		
f(2)		
f(3)		
f(4)		

Preencha os quadros segundo a função abaixo

```
def comum(seq1, seq2):
    res = []
    for x in seq1:
        if x in seq2:
            res.append(x)
    return res
```

Chamada	Valor de retorno
comum('azul', 'amarelo')	
comum(range(5), [1,3,5])	
comum('azul', ['a','b'])	

Variáveis globais não são alteradas dentro de funções, a menos que declaradas como **global** dentro delas.

Programa

```
a = 'X'
def func( ):
    a = "O"

func( )
print (a)
```

Saída

Programa

```
a = 'X'
def func( ):
    global a
    a = 'O'

func( )
print (a)
```

Saída

yield é um gerador, podemos utilizá-lo em uma função onde cada elemento é gerado online via **next()**

Programa

```
def fat():
    n = 1
    f = 1
    while True:
        f = f * n
        yield f
        n = n + 1

a = fat()
for i in range(5):
    print (next(a), end = ' ')
```

Saída

Programa

```
def fib():
    a, b = 1, 1
    while True:
        yield a
        a, b = b, a + b

a = fib()
for i in range(5):
    print (next(a), end = ' ')
```

Saída