

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

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

x x Correto

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
b = a + 7 = 12
a = 10
print (b)
```

Saída

12

Programa

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

Saída

int

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

Saída

float

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

Saída

float

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

Saída

str

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

Saída

boolean

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

Saída

float

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

Saída

int

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

Saída

int

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

Saída

int

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

Saída

comparativo

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

Saída

boolean

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

Saída

comparativo

Programa

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

Saída

str

Programa

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

Saída

True

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

Saída

0,33

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

Saída

0

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

Saída

False

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

Saída

False

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

Saída

2 → 2.0

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

Saída

2

```
print (not False)
```

Saída

True

```
print (not True)
```

Saída

False

```
print (not 0)
```

Saída

True

Programa

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

Saída

f

Programa

```
a = 20
print (15-(a-15), end = ' ')
a = 10
print (15-(a-15), end = ' ')
```

Saída

10 10

Programa

```
a = 12.75
print (a - int(a), end = ' ')
a = int((a - int(a))*100)
print (a)
```

Saída

0.75 75 → 0.75 0

Programa

```
a = 3
b = 4
a = a + b → 7
b = a - b → 3
a = a - b → 4
print (a, b)
```

Saída

4 3



Programa

print (3 % 2)

print (0 % 2)

print (123%356254)

Saída

5 → 1

0

123

P resto

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

print (type([1, 2]))

print (type({1:2}))

print (type([]))

Saída

list

dict

list

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

True False true true true true

False

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

0 3 9 0 3 2 3 4 5 6 7 8 0 3 2 7 8 9

0 2 4 6 8 9 8 7 6 5 4 3 2 1 0

Fim que consegue

Programa

```
0 3 2
a = [1, 2, [3, 4]]
print (1 in a, end = ' ')
print ([1, 2] in a, end = ' ') → F
print ([3, 4] in a, end = ' ') → F
print (3 in a, end = ' ') → F
print (3 in a[2], end = ' ') → F
print (5 not in a) ✓
```

Saída

Programa

→ existe a na posição 3? Verdadeiro

```
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

? AB

ab 3.14 || true false true false ab 3.14

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` ten que consegue Saída

(N/A)	A B
15	A
16 ate 19	B
20	C
(N/A)	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

X X X X

Programa

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

Saída

X X X X

Programa

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

Saída

N/A Sai NADA

Programa

```

a = 5678
b = 9
while a <= b:
    print ('X', end = ' ')
    if a % 2 == 0: print ('O', end = ' ')
    a = a + 1

```

612
03

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.

Saída

Programa

```

a=1  1 2 3 4 5 6
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=5  10
b=7  12 14 24
while repete:
    print ('O', end = ' ')
    a=a+5
    b=b+7
    if a + b >= 24:
        repete = False

```

Saída

Programa2

```

repete = True
a=5  10
b=0  12 14 24
while repete:
    print ('O', end = ' ')
    if a + b >= 24:
        repete = False
    a=a+5
    b=b+7

```

Saída

Programa3

```

repete = True
a=5  10
b=0  12 14 24 15
while repete:
    print ('O', end = ' ')
    if a + b > 24:
        repete = False
    a=a+5
    b=b+7

```

Programa

```

a=0  2
while a < 3:
    while True:
        print ('X', end = ' ')
        break
    print ('O', end = ' ')
    a=a+1

```

Saída

Programa

```

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

```

tem que corrigir

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

```

Saída



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

0 ~~5~~ 0

Preencha os quadros segundo a função abaixo

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

Muito louco

A função que é PRINT X chama a mesma função que é PRINT X

Chamada Valor de retorno Saída

f(1)	1	x
f(2)		
f(3)		
f(4)	10	x x x x

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

comum('azul', 'amarelo')
comum(range(5), [1,3,5])
comum('azul', ['a','b'])

Valor de retorno

A L
1 3 ~~5~~
A

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

~~2~~ X

Programa

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

Saída

0

`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
```

Contador

1 2 3 4 5
1 2 6 24 120

Saída

1 2 6 24 120

Programa

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

Posição

1 A=1 b=1
2 A=1 b=2
3 A=2 b=3
4 A=3 b=5
5 A=5 b=8

Saída

1 2 3 5 ~~8~~

