# Exercícios - Lista XIV - Revisão Geral - Python para Zumbis

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

Exemplo de programa:		Programa	Saída	
<pre>print ('x', end = ' ') print ('x', end = ' ')</pre>		print (3 3.0)	TRUE	
Saída (colocar que dá erro se for o caso):		print (1/3)	O.33333333	
xx		print (1//3)	0	
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 a = 10 print (b)		print (3 -= '3')	FALSE	
		print ('x' != 'x')	FALSE	
		print (2/1)	2.0 (com a	
		print (2//1)	2	
		print (not False)	TRUE	
		print (not True)	FALSE	
		print (not 0)	TRUE	
		Programa (Tables of Mana)		
Saída		print (True and (False or not True))		
12		Saida FALSE		
Programa	Saída	Programa		
print (type(0))	INT	a = 20		
print (type(0.0))	FLOAT	print (15-(a-15), a = 10		
print (type(3.14))	FLOAT	print (15-(a-15), end = '')		
print (type('Py'))	STR	Saída		
orint (type(True))	BOOL	10 20	Programa	
orint (type(1/2))	FLOAT			
rint (type(1//2))	INT	a = 12.75 print (a - int(a)		
rint (type(2//1))	INT	a = int((a - int( print (a)		
rint (type(3**3))	INT	Saída		
cint (type(0==0))	BOOL			
rint (type(3<0))	BOOL	Programa		
int (type(3!=3))	BOOL	a = 3 b = 4		
		a = a + b7	a = a + b +	
grama		b = a - b 3 $a = a - b 4$		
int (type(str(int(3	.14159265358979))))	print (a, b)		
3		Saída		
		4 3		

#### Programa Saída Condicionais if/else/elif O comando if executa um bloco de comandos somente print (3 % 2) T se a condição é True. Esta condição pode ser qualquer print (0 % 2) O colsa. Os comandos else e elif são opcionais que são testadas apenas quando condições anteriores não print (123%356254) 123 forem satisfeitas. Programa Saída Programa print (type([1, 2])) List a = ?if a > 10 and a % 6 == 3: print (type({1:2})) ERRO print ('A', end = ' ') elif a > 10 and a < 20: print (type([])) ERRO print ('B', end = ' ') else: print ('C', end = ' ') Programa a = 'abacate' print ('e' in a, 'x' in a, end = ' ') Dê os valores de a que produzem a saída ('N/A' se não print ('ate' in a, end = ' ') houver valor possível para a): print ('' in a, end = ' ') Valores de a Saída print ('eta' in a, end = ' ') print ('eta' not in a) N/A A B Saída 15 Α TRUE FALSE TRUE TRUE FALSE TRUE 11 В **Programa** 100 C a = '0123456789'print (a[0], a[3], a[-1], end = ' ')Feliz Natal! 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]) Comandos while/for/break/continue Os laços while e for permitem que você repita um Saída bloco de comandos várias vezes, break interrompe o laço e continue faz a execução voltar ao início do 012 345 678 012 789 03 laço. 02468 9876543210 Programa Programa while a < 10: print ('X', end = ' ') a = [1, 2, [3, 4]]print (1 in a, end = ' ') Saída print ([1, 2] in a, end = ' ') print ([3, 4] in a, end = ' ') infinitor X's print (3 in a, end = ' ') print (3 in a[2], end = ' ') **Programa** print (5 not in a) a = -1while a < 3: print ('X', end = ' ') Saída a = a + 1TRUE FALSE TRUE FALSE TRUE TRUE Saída **Programa** $a = \{1: 'ab', 2: 'cd', 'x':3.14\}$ Programa print (1 in a, 3 in a, end = ' ')

Saída

TRUE FALSE TRUE FALSE ab 3.4

print ('x' in a, 'z' in a, end = ' ')

print (a[1], a['x'])

VAZIO

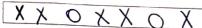
Saída

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



```
a = 5
b = 9
while a <= b:
    print ('x', end = ' ')
    if a % 2 == 0: print ('0', end = ' ')
    a = a + 1</pre>
```

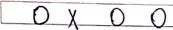
#### Saída



#### Programa

```
a=1
while a % 7 != 0:
   if a % 2 == 0: print ('0', 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 ('0', end = ' ')
    if a + b >= 24:
        repete = False
    a=a+5
    b=b+7
```

#### Saída



### Programa3

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

#### Saída

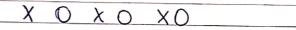


Laços dentro de laços. Determine bem os comandos do bloco de cada laço. break e continue se aplicam ao laço do seu bloco apenas. Aponte loops infinito caso ocorra.

#### Programa

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

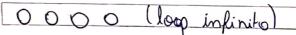
#### Saída



#### Programa

```
a=1
while a < 3:
    while a < 3:
        print ('0', end = ' ')
    a=a+1</pre>
```

#### 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</pre>
```

#### Saída



### Programa

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

#### Saída

otinifica goal 0000

Programa

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

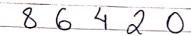
Saída

ABACATE BACATE ACATE CATE ATE TE E

Programa

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

Saída



Programa

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

Saída

```
NOME: BRUND
BOM DID BRUND
NOME: FIM
```

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

```
X
```

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]:
```

Saída

```
A 3.14 3.5
```

Programa

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

Saída

12

Programa

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

Saída

4

Programa

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

Saída

```
and see iii ooo uu
```

Programa

```
L = [1, 2, 3, 4, 5]

for x in range(len(L)):

L[x] += 1

print (L)
```

Saída

```
[2,3,4,5,6]
```

Programa

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

Saída

```
a0 a1 a2 b0 b1 b2 c0 c1 c2
```

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]

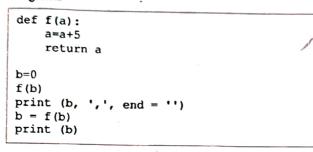
print (x)
```

Saída



Uma função é uma sequência 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



#### 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)	100 mm and	X
f(2)	3	X×
f(3)	6	XXX
f(4)	10	XXXX

## 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

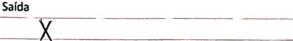
#### Valor de retorno

```
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 = "0"
func()
print (a)
```



## Programa

```
a = 'x'
def func():
   global a
    a = '0'
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

3 5 2