

MC102 - Algoritmos e Programação de Computadores

Turma Z - Segundo Semestre de 2019

A partir desse slide, utilizaremos o material desenvolvido pela professora Sandra Avila e disponível em <http://www.ic.unicamp.br/~sandra/>

Exercício 1: Reverso

```
def reverso(n):  
    invert = str(n)  
    print(invert[::-1])
```

Exercícios

1. Faça uma função que retorne o reverso de um número inteiro informado. Por exemplo: 127 -> 721.
2. Faça uma função que informe a quantidade de dígitos de um determinado número inteiro informado.
3. Faça uma função que computa a potência a^b para valores a e b (assuma números inteiros) passados por parâmetro (não use o operador `**`).

Exercício 2: Número de dígitos

```
def digitos(n):  
    s = str(n)  
    return len(s)
```

Exercícios

1. Faça uma função que retorne o reverso de um número inteiro informado. Por exemplo: 127 -> 721.
2. Faça uma função que informe a quantidade de dígitos de um determinado número inteiro informado.
3. Faça uma função que computa a potência a^b para valores a e b (assuma números inteiros) passados por parâmetro (não use o operador `**`).

Exercício 3: Potência

```
def potencia(base, expoente):  
    resultado = 1  
    for numero in range(1, expoente+1):  
        # base ** expoente = base * base (expoente vezes)  
        resultado = resultado * base  
    return resultado
```

Agenda

- Variáveis locais e globais
- Listas em funções

Variáveis Locais e Variáveis Globais

- Uma variável é chamada **local** se ela é criada ou alterada **dentro de uma função**.
- Nesse caso, ela existe somente dentro daquela função, e após o término da execução da mesma a variável deixa de existir.
- Variáveis parâmetros também são variáveis locais.

Variáveis Locais e Variáveis Globais

- Uma variável é chamada **global** se ela for criada **fora de qualquer função**.
- Essa variável pode ser visível por todas as funções.
- Qualquer função pode alterá-la.

Organização de um Programa

```
variáveis globais
```

```
def main():  
    variáveis locais  
    comandos
```

```
def função1(parâmetros):  
    variáveis locais  
    comandos
```

```
def função2(parâmetros):  
    variáveis locais  
    comandos
```

```
...
```

```
main()
```

Escopo de Variáveis

- O **escopo** de uma variável determina de quais partes do código ela pode ser acessada, ou seja, de quais partes do código a variável é visível.
- A regra de escopo em Python é bem simples:
 - As variáveis **globais** são **visíveis por todas as funções**.
 - As variáveis **locais** são **visíveis apenas na função onde foram criadas**.

Variáveis Locais e Variáveis Globais

```
def f1(a):  
    print(a+x)  
  
def f2(a):  
    c = 10  
    print(a+x+c)  
  
x = 4  
f1(3)  
f2(3)  
print(x)
```

```
7  
17  
4
```

Tanto **f1** quanto **f2** usam a variável `x` que é global pois foi criada fora das funções.

http://www.pythontutor.com/visualize.html

Get live help!

Start private chat

[How do I use this?](#)

These Python Tutor users are asking for help right now. Please volunteer to help!

user_2d1 from Tunisia needs help with Python3 - 2 people chatting - [click to help](#) (active a few seconds ago, requested an hour ago)

user_b07 from Fuzhou, China needs help with Python3 - 3 people chatting - [click to help](#) (active a few seconds ago, requested 4 minutes ago)

user_55d from Nanning, China needs help with Python3 - [click to help](#) (IDLE: last active 8 minutes ago, requested 2 hours ago)

user_e41 from Seoul, Republic of Korea needs help with Java - [click to help](#) (IDLE: last active an hour ago, requested an hour ago)

Python 3.6

```
1 def f1(a):
2     print(a+x)
3
4 def f2(a):
5     c = 10
6     print(a+x+c)
7
8 x = 4
9 f1(3)
10 f2(3)
11 print(x)
```

[Edit this code](#)

→ line that has just executed

→ next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First

< Back

Step 11 of 13

Forward >

Last >>

Created by [@pgbovine](#). Support with a [small donation](#).

Help improve this tool by clicking whenever you learn something:

Print output (drag lower right corner to resize)

```
7
```

Frames

Objects

Global frame

f1 → function f1(a)

f2 → function f2(a)

x | 4

f2

a	3
c	10

Variáveis Locais e Variáveis Globais

```
def f1(a):  
    x = 10  
    print(a+x)  
  
def f2(a):  
    c = 10  
    print(a+x+c)
```

```
x = 4  
f1(3)  
f2(3)  
print(x)
```

```
13  
17  
4
```

Neste outro exemplo **f1** cria uma variável local `x` com valor 10. O valor de `x` global permanece com 4.

Variáveis Locais e Variáveis Globais

```
def f1(a):  
    print(a+x)  
  
def f3(a):  
    x = x + 1  
    print(a+x)  
  
x = 4  
f1(3)  
f3(3) # este comando vai dar um erro
```

Por que vai dar erro? O erro ocorre pois está sendo usado uma variável local `x` antes dela ser criada!

Variáveis Locais e Variáveis Globais

```
def f1(a):  
    print(a+x)
```

```
def f3(a):  
    global x  
    x = x + 1  
    print(a+x)
```

```
x = 4
```

```
f1(3)
```

```
f3(3)
```

```
print(x)
```

```
7
```

```
8
```

```
5
```

Para que **f3** use `x` global devemos especificar isto utilizando o comando `global`.

Variáveis Locais e Variáveis Globais

```
def f2(a):  
    c = 10  
    print(a+x+c)  
  
x = 4  
f2(3)  
print(x)  
print(c) # este comando vai dar um erro
```

Por que vai dar erro? A variável `c` foi criada dentro da função `f2` e ela só existe dentro desta.

Ela é uma **variável local** da função `f2`.

Variáveis Locais e Variáveis Globais

```
def f4(a):  
    c = 10  
    print("c de f4:", c)  
    print(a+x+c)
```

```
x = 4  
c = -1  
f4(1)  
print("c global:", c)
```

```
c de f4: 10  
15  
c global: -1
```

Neste caso existe uma variável `c` no programa principal e uma variável local `c` pertencente à função `f4`.

Alteração no valor da **variável local** `c` dentro da função não modifica o valor da **variável global** `c`, a menos que esta seja declarada como global.

Variáveis Locais e Variáveis Globais

```
def f4(a):  
    global c  
    c = 10  
    print("c de f4:", c)  
    print(a+x+c)
```

```
x = 4  
c = -1  
f4(1)  
print("c global:", c)
```

```
c de f4: 10  
15  
c global: 10
```

Neste caso a variável `c` de dentro da função `f4` foi declarada como global. Portanto é alterado o conteúdo da variável `c` fora da função.

Variáveis Locais e Variáveis Globais

- O **uso de variáveis globais deve ser evitado** pois é uma causa comum de erros:
 - Partes distintas e funções distintas podem alterar a variável global, causando uma grande interdependência entre estas partes distintas de código.

Listas em Funções

```
def f5(a):  
    a.append(3)
```

```
a = [1,2]  
f5(a)  
print(a)
```

```
[1, 2, 3]
```

Neste caso mesmo havendo uma variável local `a` de `f5` e uma global `a`, o conteúdo de `a` global é alterado. O que aconteceu?

Lembre-se que `a` local de `f5` recebe o identificador da lista de `a` global. Como uma lista é mutável, o seu conteúdo é alterado.

Listas em Funções

```
def f5(a):  
    a = [10,10]  
  
a = [1,2]  
f5(a)  
print(a)
```

```
[1, 2]
```

Neste caso a variável `a` local de `f5` recebe uma nova lista, e portanto um novo identificador.

Logo a variável `a` global não é alterada.

Listas em Funções

```
def f5(a):  
    global a  
    a = [10,10]  
  
a = [1,2]  
f5(a)  
print(a)
```

```
[10, 10]
```

Neste caso `a` de `f5` é global e portanto corresponde a mesma variável fora da função.

Referências & Exercícios

- Os slides dessa aula foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp).
- <https://wiki.python.org.br/ExerciciosFuncoes>
- <https://panda.ime.usp.br/aulasPython/static/aulasPython/aula06.html>
- <https://panda.ime.usp.br/aulasPython/static/aulasPython/aula10.html>