FIEP SESI SENAI IEL

SENAI

Comandos Python: Funções - Parte 2

antes disto,

Exercícios

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 1: Reverso

```
def reverso(n):
  s = str(n)
  rev = s[::-1]
  return int(rev)
```

Exercício 1: Reverso

```
def reverso(n):
  s = str(n)
  rev = s[::-1]
  return int(rev)
```

```
def reverso(n):
  return int(str(n)[::-1])
```

Exercício 1: Reverso

```
def reverso(n):
  s = str(n)
  rev = s[::-1]
  return int(rev)
```

```
def reverso(n):
  return int(str(n)[::-1])
```

```
def reverso(n): return int(str(n)[::-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)
```

Exercício 2: Número de dígitos

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

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

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 **).**

Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.

```
base = int(input("Digite a base: ")) # base a
expoente = int(input("Digite o expoente: ")) # expoente b

resultado = 1

for numero in range(1,expoente+1):
    # base ** expoente  = base * base (expoente vezes)
    resultado = resultado * base
print(base, "elevado a", expoente, "=", resultado)
```

Exercício 3: Potência (números inteiros)

```
def potencia(base,expoente):
  if expoente >= 0 :
    resultado = 1
    for numero in range(expoente):
      resultado = resultado * base
    return resultado
  else:
    resultado = 1
    for numero in range(-expoente):
      resultado = resultado / base
    return resultado

print(potencia(2,3))
```

Comandos Python: Funções - Parte 2

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.

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():
    global a
    a = [10,10]

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

```
[10, 10]
```

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

Exercício

Neste exercício, você irá praticar o uso de variáveis globais e locais em funções, além de manipular listas dentro de funções.

1. **Variáveis Globais e Locais**:
   - Crie uma variável global chamada `contador`, com valor inicial igual a 0.
   - Escreva uma função chamada `incrementar_contador()`, que incremente o valor da variável global `contador` em 1 e imprima o novo valor de `contador`.
   - Escreva outra função chamada `exibir_contador()`, que imprima o valor atual de `contador`.
2. **Listas em Funções**:
   - Crie uma função chamada `adicionar_na_lista(item, lista)`, que recebe dois parâmetros: um item a ser adicionado e uma lista.
   - A função deve adicionar o item na lista fornecida e retornar a lista modificada.
   - Crie uma função chamada `exibir_lista(lista)`, que apenas imprime os elementos da lista.
3. **Teste suas Funções**:
   - No bloco principal do programa, crie uma lista vazia chamada `minha_lista`.
   - Adicione três itens à lista utilizando a função `adicionar_na_lista()` e exiba o conteúdo da lista com a função `exibir_lista()`.
   - Incremente o valor de `contador` duas vezes com a função `incrementar_contador()` e exiba o valor final com a função `exibir_contador()`.

Resultado esperado

```
Itens na lista: ['Maçã', 'Banana', 'Laranja']
Contador atualizado: 1
Contador atualizado: 2
Valor atual do contador: 2
```

Referências & Exercícios

- Os *slides* deste curso foram baseados nos slides produzidos e cedidos gentilmente pela Professora Sandra Ávila, do Instituto de Computação da Unicamp. Parte dos slides foram baseados no material 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