Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas

INF 100 – Introdução à Programação

Funções
(cont.)

Escopo das Variáveis

Variáveis Locais

```
def f( x, y ):
    k = 2*x + y
    return k

z = 2
w = 1
print( f( 2*z, w ))
```

x, **y** e **k** são variáveis 'visíveis' apenas **dentro** da função **f()**. Elas só podem ser **utilizadas dentro de f()**.
Por isso, são chamadas de **variáveis locais**.


Escopo das Variáveis

Variáveis Locais

```
def f( x, y ):
    z = 2*x + y
    return z

z = 2
w = 1
print( z )
print( f( 2*z, w ))
print( z )
```

As variáveis **z** da função **f()** e do programa principal são duas **variáveis distintas**! Estão em escopos diferentes (por isso **podem ter o mesmo nome**).


Escopo das Variáveis

Variáveis Globais

```
def f( x, y ):
    k = 2*x + y + z
    return k

z = 2
w = 1
print( f( 2*z, w ))
```

Escopo local de **f()**

Neste caso a variável **z** em **f()** é tratada como uma **variável global**, pois ela não foi criada no escopo de **f()**, e sim antes de **f()** ser chamada.

Escopo local do programa principal


Escopo das Variáveis

Variáveis Globais

```
def f( x, y ):
    k = 2*x + y + z
    z = 1
    return k

z = 2
w = 1
print( f( 2*z, w ))
```

Este exemplo gera um erro, pois existe uma ambiguidade. Isto é, **z** deve ser tratada como uma variável local ou global?


Escopo das Variáveis

Variáveis Globais

```
def f( x, y ):
    k = 2*x + y + z
    global temp
    temp = 2
    return k

temp = 0
z = 2
w = 1
print( temp )
print( f( 2*z, w ))
print( z, w )
print( temp )
```

Escopo local de f()

Escopo global (não recomendado)

Escopo local do programa principal

Qual será o valor de temp?

Escopo das Variáveis

Variáveis Globais

```
def f( x, y ):
    k = 2*x + y + z
    global temp
    temp = 2
    return k

temp = 0
z = 2
w = 1
print( temp )
# print( f( 2*z, w ))
print( z, w )
print( temp )
```

Escopo local de f()

Escopo global (não recomendado)

Escopo local do programa principal

Qual será o valor de temp?

Retornando mais de um Resultado

```
def ordena( x, y ):
    if (x <= y):
        return x, y
    else:
        return y, x

a = float( input('Digite um número: '))
b = float( input('Digite outro número: '))
a, b = ordena( a, b )
print( a, '<=', b )
```


Exercício 1

- Usando a função `ordena()` do slide anterior, faça um programa que leia três valores inteiros e escreva esses valores ordenados.
  Exemplo:

  Entrada: A=7 B=3 C=5

  Saída na tela: A=3 B=5 C=7

Exercício 1

```
def ordena( x, y ):
    if (x <= y):
        return x, y
    else:
        return y, x

a = int( input('Digite o 1º número: '))
b = int( input('Digite o 2º número: '))
c = int( input('Digite o 3º número: '))
a, b = ordena( a, b )
b, c = ordena( b, c )
a, b = ordena( a, b )
print( a, '<=', b, '<=', c )
```


Resumo – Escopo e Parâmetros

- Variáveis Globais: compartilhadas com todas funções do programa;
- Variáveis Locais: internas a cada função;
- Os parâmetros são variáveis Locais:
  - Os parâmetros passados por valor são **independentes** das variáveis, expressões etc. que geraram os valores passados para a função;


Passagem de arranjos como parâmetros

- Ao passar um arranjo como parâmetro, o programa não faz uma cópia do arranjo para dentro da variável local. Isso seria muito custoso!
- Em vez disso, ele só passa o endereço da memória onde esse arranjo está armazenado.
- Assim, qualquer alteração feita ao arranjo altera a própria variável passada como parâmetro, e não uma cópia dessa variável.


Passagem de arranjos como parâmetros

- Exemplo:

```
def dobra( vetor ):
    for i in range( 0, len( vetor )):
        vetor[i] = vetor[i] * 2

a = numpy.array([4, 3, 2, 1])
print( a )
dobra( a )
print( a )
```


Exercício 2

- Desenvolva uma função que, dada uma string *s* e um caractere *c*, retorna o número de ocorrências de *c* em *s*.


Exercício 2

```
# Retorna o número de ocorrências do
# caractere c na string s
def ocorrencias( c, s ):
    n = 0
    for i in range( 0, len( s )):
        if s[i] == c:
            n = n + 1
    return n

# Exemplo de uso da função:
s = input('Digite uma frase: ')
c = input('Digite um caractere: ')
print('Encontradas', ocorrencias( c, s ),
      'ocorrências de', c, 'na frase.')
```


Exercício 3

- Faça uma função em Python que recebe como parâmetro um arranjo de números, e retorne a média e o desvio padrão dos valores do arranjo.

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad s = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)^2}{n}}$$


Exercício 3
(versão mais simples)

```
import numpy as np

# Retorna a média e o desvio
# padrão do arranjo v
def calcMediaDesvio( v ):
    n = len( v )
    soma = 0.0
    for i in range(0,n):
        soma = soma + v[i]
    media = soma / n
    soma = 0.0
    for i in range(0,n):
        soma = soma + (v[i] - media) ** 2
    desvio = (soma / n) ** 0.5
    return media, desvio
```

```
n = int( input('Quantidade de valores: '))
a = np.empty( n )
print('Entre com os valores do arranjo:')
for i in range( 0, n ):
    a[i] = float( input(''))

if n > 0:
    m, d = calcMediaDesvio( a )
    print('Média:', m )
    print('Desvio:', d )
```


Exercício 3 (versão mais elaborada)

```
import numpy as np

# Retorna True se parâmetro é um número de ponto
# flutuante, e False caso contrário
def is_float( s ):
    try:
        # Tenta converter string para float
        float( s )
        return True
    except ValueError:
        # Se der erro, retorna False
        return False

# Retorna a média e o desvio padrão do arranjo v
def calcMediaDesvio( v ):
    n = len( v )
    soma = 0.0
    for i in range(0,n):
        soma = soma + v[i]
    media = soma / n
    soma = 0.0
    for i in range(0,n):
        soma = soma + (v[i] - media) ** 2
    desvio = (soma / n) ** 0.5
    return media, desvio

a = np.empty( 0 )  # começa com um arranjo vazio
n = 0  # número de elementos em a
print('Entre com os valores do arranjo (ENTER
termina):')
while True:
    x = input('')
    if not is_float( x ):
        break
    n = n + 1
    a = np.resize( a, n )
    a[n-1] = float(x)

if n > 0:
    m, d = calcMediaDesvio( a )
    print('Média:', m )
    print('Desvio:', d )
```


Exercício 4

- Desenvolva uma função `CPF_digitos( cpf )` que, dado um número de CPF no formato xxxxxxxxx (sem os dígitos verificadores), retorna uma string contendo os dois dígitos verificadores. Para saber como se calcula esses dígitos, consulte o seguinte link:

http://www.geradorcpf.com/algoritmo_do_cpf.htm


Exercício 4

```
# Retorna os 2 dígitos de verificação de um CPF,
# ou None se o CPF não tiver 9 dígitos
def CPF_digitos( cpf ):
    n = len( cpf )
    if n != 9:
        return ''
    # calcula somatórios
    m = o = 0
    for i in range( 0, 9 ):
        dig = int( cpf[i] )
        m = m + dig * (10 - i)
        o = o + dig * (11 - i)
    # calcula 1º dígito
    d1 = m % 11
    if d1 < 2:
        d1 = 0
    else:
        d1 = 11 - d1
    # calcula 2º dígito
    o = o + 2 * d1
    d2 = o % 11
    if d2 < 2:
        d2 = 0
    else:
        d2 = 11 - d2
    # concatena e retorna os dois dígitos
    return str( d1 ) + str( d2 )

while True:
    cpf = input('\nCPF (sem os dígitos
verificadores): ')
    if cpf == '': break
    print( cpf + '-' + CPF_digitos( cpf ))
```

Exercício 5

- Usando a função do Exercício 4, desenvolva uma função `CPF_valido( cpf )` que, dado um número de CPF completo no formato xxxxxxxxxyy ou xxxxxxxxx-yy ou xxx.xxx.xxx-yy, retorna True se essa string representa um CPF válido, e False caso contrário.
- Dica: faça uma terceira função para obter somente os dígitos de uma string *s* qualquer. Com isso sua função não precisará lidar com caracteres especiais e diferentes formatos de entrada.


Exercício 5

```
# retorna somente os dígitos 0-9 contidos em s
def getDigitos( s ):
    r = ''
    for i in range( 0, len( s )):
        if s[i] >= '0' and s[i] <= '9':
            r = r + s[i]
    return r

# Extrai dígitos da string contendo CPF e
# retorna True se o número contém CPF válido
# no formato xxxxxxxxxyy.
def CPF_valido( cpf ):
    s = getDigitos( cpf )
    return len( s ) == 11 and\
           CPF_digitos( s[:9] ) == s[9:]

# Exemplo de uso da função
while True:
    cpf = input('\nCPF: ')
    if cpf == '': break
    if CPF_valido( cpf ): print('CPF válido.')
    else: print('CPF inválido.')
    s1 = getDigitos( cpf )[:9]
    print( s1 + '-' + CPF_digitos( s1 ))
```


Tartarugas em Python

```
import turtle as t

def poligono_regular( n, tamanho ):
    # Calcular o complemento do ângulo interno do polígono
    ang_interno = 180 - (n-2)*180/n
    for i in range(0, n):
        t.forward( tamanho )
        t.right( ang_interno )

while True:
    n = int(input('\nNúmero de lados do polígono: '))
    if n < 3: break
    tam = int(input('Comprimento de cada lado: '))
    poligono_regular( n, tam )
```


```
# Desenhar quadrados concêntricos

import turtle as t

def quadrado( tamanho ):
    for i in range(0, 4):
        t.forward( tamanho )
        t.right( 90 )

def shift( delta_x, delta_y ):
    t.up()
    t.goto( t.xcor() + delta_x, t.ycor() + delta_y )
    t.down()

h = int(input('Tamanho do quadrado externo: '))

while h > 0:
    quadrado( h )
    shift( 5, -5 )
    h = h - 10

t.Screen().exitonclick()
```