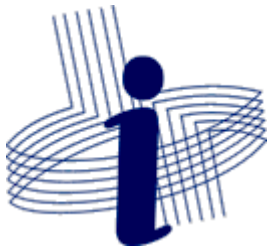
Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas
Departamento de Informática



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. Em geral evitamos usar variáveis globais dentro de funções por causa do risco de efeitos colaterais, perda de generalidade da função etc.

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. Em geral evitamos usar variáveis globais dentro de funções por causa do risco de efeitos colaterais, perda de generalidade da função etc.

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.

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

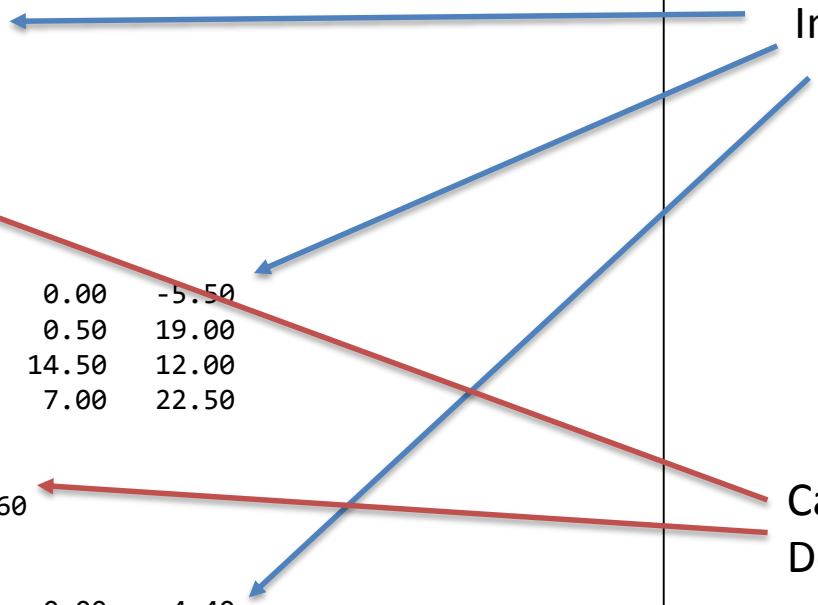
Exercício 3

```python
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
```

```python
n = int( input('Quantidade de valores: '))
# Gerar valores aleatórios em [0,10)
a = np.random.uniform( 0, 10, n )
print( a, '\n’)

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

Exercício 4
(Prática 11)

```
Entre com a Ordem da matriz (número par > 0): 0
Entre com a Ordem da matriz (número par > 0): 1
Entre com a Ordem da matriz (número par > 0): 4

Matriz A:
  12  15   0   3
   3   7   9  19
  18   4   6  12
   1   6   7  14

Média: 8.50

Matriz B:
   20.50   15.00    0.00   -5.50
    3.00   15.50    0.50   19.00
   18.00   -4.50   14.50   12.00
   -7.50    6.00    7.00   22.50

Média: 8.50
Desvio padrão: 9.60

Matriz C:
    2.40    0.00    0.00   -4.40
    0.00    0.00    0.00    0.90
    0.00   -3.40    0.00    0.00
   -6.40    0.00    0.00    4.40
```



Impressão de matriz

Calcular média e desvio padrão
De matriz

Exercício 4

```
import numpy as np


def imprimeMatriz( matriz, titulo, formato ):
    m, n = matriz.shape
    if titulo != '':
        print( titulo )
    for i in range( 0, m ):
        for j in range( 0, n ):
            print( formato % matriz[i][j], end='')
        print()


def calcMediaDP_matriz( matriz ):
    m, n = matriz.shape
    soma = 0
    for i in range( 0, m ):
        for j in range( 0, n ):
            soma = soma + matriz[i][j]
    media = soma / (m*n)
    soma = 0
    for i in range( 0, m ):
        for j in range( 0, n ):
            soma = soma + (matriz[i][j] - media) ** 2
    desvpad = (soma / (m*n)) ** 0.5
    return media, desvpad

np.random.seed( 0 )


n = 0
while n < 2 or (n % 2 != 0):
    n = int( input('Entre com a Ordem da matriz (número par > 0): '))


A = np.random.randint( 0, 21, (n,n) )


imprimeMatriz( A, '\nMatriz A:', '%4d')


media, desvpad = calcMediaDP_matriz( A )
print('\nMédia: %.2f' % media )


B = np.empty( (n,n) )
for i in range( 0, n ):
    for j in range( 0, n ):
        if i == j:
            B[i][j] = A[i][j] + media
        elif j == n-1-i:
            B[i][j] = A[i][j] - media
        else:
            B[i][j] = A[i][j]


imprimeMatriz( B, '\nMatriz B:', '%8.2f')

media, desvpad = calcMediaDP_matriz( B )
print('\nMédia: %.2f' % media )
print('Desvio padrão: %.2f' % desvpad )


C = np.empty( (n,n) )
for i in range( 0, n ):
    for j in range( 0, n ):
        b = B[i][j]
        if b < media-desvpad:
            C[i][j] = b - (media-desvpad)
        elif b > media+desvpad:
            C[i][j] = b - (media+desvpad)
        else:
            C[i][j] = 0


imprimeMatriz( C, '\nMatriz C:', '%8.2f')
```

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()
```

Para Casa

Faça os exercícios do PVANet.