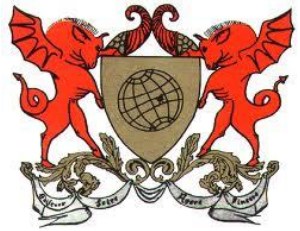


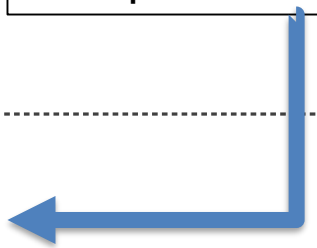
Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas

INF 100 – Introduction to Programming

Functions
(additional examples)

```
import numpy as np

m = int( input('Number of lines in matrix A: '))
n = int( input('Number of columns in matrix A: '))
p = int( input('Number of columns in matrix B: '))

print()
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))

print('\nMatrix A:')
for i in range(0, m):
    for j in range(0, n):
        print('%5.1f' % (A[i][j]), end='')
    print()

print()
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))

print('\nMatrix B:')
for i in range(0, n):
    for j in range(0, p):
        print('%5.1f' % (B[i][j]), end='')
    print()

C = np.zeros( (m, p) )
for i in range(0, m):
    for j in range(0, p):
        for k in range(0, n):
            C[i][j] += A[i,k] * B[k,j]

print('\nProduct AB :')
for i in range(0, m):
    for j in range(0, p):
        print('%5.1f' % (C[i][j]), end='')
    print()
```

Multiplication of matrices



Identifying possibilities for code reuse...

```
import numpy as np

m = int( input('Number of lines in matrix A: '))
n = int( input('Number of columns in matrix A: '))
p = int( input('Number of columns in matrix B: '))

print()
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))

print('\nMatrix A:')
for i in range(0, m):
    for j in range(0, n):
        print('%5.1f' % (A[i][j]), end='')
    print()
```

...

```
import numpy as np

m = int( input('Number of lines in matrix A: '))
n = int( input('Number of columns in matrix A: '))
p = int( input('Number of columns in matrix B: '))

print()
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))

print('\nMatrix A:')
for i in range(0, m):
    for j in range(0, n):
        print('%5.1f' % (A[i][j]), end='')
    print()
```

read matrix A

print matrix A

...

... continuing

```
print()
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))

print('\nMatrix B:')
for i in range(0, n):
    for j in range(0, p):
        print('%5.1f' % (B[i][j]), end='')
    print()
```

...

... continuing

```
print()
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))

print('\nMatrix B:')
for i in range(0, n):
    for j in range(0, p):
        print('%5.1f' % (B[i][j]), end='')
    print()
```

read matrix B

print matrix B

...

... and finally:

```
C = np.zeros( (m, p) )
for i in range(0, m):
    for j in range(0, p):
        for k in range(0, n):
            C[i][j] += A[i,k] * B[k,j]

print('\nProduct AB :')
for i in range(0, m):
    for j in range(0, p):
        print('%5.1f' % (C[i][j]), end='')
    print()
```

... and finally:

```
C = np.zeros( (m, p) )
for i in range(0, m):
    for j in range(0, p):
        for k in range(0, n):
            C[i][j] += A[i,k] * B[k,j]

print('\nProduct AB :')
for i in range(0, m):
    for j in range(0, p):
        print('%5.1f' % (C[i][j]), end='')
    print()
```

calculates C=AxB

print matrix C

Code reuse with functions

- In the program for multiplication of matrices, a similar code is used for reading matrices A and B:

```
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))
```

```
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))
```


Code reuse with functions

- In the program for multiplication of matrices, a similar code is used for reading matrices A and B:

```
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))
```

```
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))
```

- The only differences between the codes are the name of the matrices and the dimensions.


Code reuse with functions

- Similar codes are also used for printing A, B and C:

```
for i in range(0, m):
    for j in range(0, n):
        print('%5.1f' % (A[i][j]), end='')
    print()
```

```
for i in range(0, n):
    for j in range(0, p):
        print('%5.1f' % (B[i][j]), end='')
    print()
```

```
for i in range(0, m):
    for j in range(0, p):
        print('%5.1f' % (C[i][j]), end='')
    print()
```


Code reuse with functions

- Suppose that the following functions are available:
  - **readMatrix**: given the dimensions, reads all the elements of a matrix from the keyboard and returns this matrix;
  - **printMatrix**: produces a formatted output of a given matrix.
  - **calcProduct**: returns the product of 2 given matrices.


Code using funtions

```
m = int( input('Number of lines in matrix A: '))
n = int( input('Number of columns in matrix A: '))
p = int( input('Number of columns in matrix B: '))

print("\nreading matrix A...")
A = readMatrix(m, n)

print('\nMatrix A:')
printMatrix(A)

print("\nreading matrix B...")
B = readMatrix(n, p)

print('\nMatrix B:')
printMatrix(B)

C = calcProduct(A, B)

print('\nProduct AB :')
printMatrix(C)
```

Code using funtions

```
m = int( input('Number of lines in matrix A: '))
n = int( input('Number of columns in matrix A: '))
p = int( input('Number of columns in matrix B: '))

print("\nreading matrix A...")
A = readMatrix(m, n)

print('\nMatrix A:')
printMatrix(A)

print("\nreading matrix B...")
B = readMatrix(n, p)

print('\nMatrix B:')
printMatrix(B)

C = calcProduct(A, B)

print('\nProduct AB :')
printMatrix(C)
```

The new version is much shorter and clearer.
It resembles the descritpion of an algorithm in English.

Code using funtions

```
m = int( input('Number of lines in matrix A: '))
n = int( input('Number of columns in matrix A: '))
p = int( input('Number of columns in matrix B: '))

print("\nreading matrix A...")
A = readMatrix(m, n)

print('\nMatrix A:')
printMatrix(A)

print("\nreading matrix B...")
B = readMatrix(n, p)

print('\nMatrix B:')
printMatrix(B)

C = calcProduct(A, B)

print('\nProduct AB :')
printMatrix(C)
```

The new version is much shorter and clearer.
It resembles the descritpion of an algorithm in English.

Now it is necessary to define the functions...

Functions in Python

- Steps for creating and using functions:
  1. Declare a function and define its code.
  2. Call the function from other parts of the program.

- We have already presented examples of (2). Now we will see how we can build our own functions.


Creating functions in Python

- In order to create a new function, it is important to know:
  - If parameters (input data) are necessary, what are these parameters?
  - If the function returns values, what are these values?
- This information will define how the function will be designed.


Function printMatrix

- How to build the function **printMatrix**, that produces a formatted output of a given matrix?


Function printMatrix

- How to build the function **printMatrix**, that produces a formatted output of a given matrix?
- Parameter: a matrix M
- Return value: none


Function printMatrix

- How to build the function **printMatrix**, that produces a formatted output of a given matrix?
- Parameter: a matrix M
- Return value: none

```
def printMatrix(M):
    code of the function
```


Function printMatrix

- Sample code for printing A, B and C:

```
for i in range(0, m):
    for j in range(0, n):
        print('%5.1f' % (A[i][j]), end='')
    print()
```

```
for i in range(0, n):
    for j in range(0, p):
        print('%5.1f' % (B[i][j]), end='')
    print()
```

```
for i in range(0, m):
    for j in range(0, p):
        print('%5.1f' % (C[i][j]), end='')
    print()
```


Function printMatrix

```
def printMatrix(M):
    lines, columns = M.shape
    for i in range(0, lines):
        for j in range(0, columns):
            print('%5.1f' % (M[i][j]), end='')
        print()
```

Function printMatrix

```
def printMatrix(M):
    lines, columns = M.shape
    for i in range(0, lines):
        for j in range(0, columns):
            print('%5.1f' % (M[i][j]), end='')
        print()
```

Compare with the code for printing A:

```
for i in range(0, m):
    for j in range(0, n):
        print('%5.1f' % (A[i][j]), end='')
    print()
```


Function readMatrix

- Function **readMatrix** reads all the elements of a matrix from the keyboard and returns this matrix, when given the dimensions of the matrix.

Function readMatrix

- Function **readMatrix** reads all the elements of a matrix from the keyboard and returns this matrix, when given the dimensions of the matrix.
- Parameters: ***lines*** (number of lines) and ***columns*** (number of columns)
- Return value: the matrix whose values were typed by the user.


Function readMatrix

- Function **readMatrix** reads all the elements of a matrix from the keyboard and returns this matrix, when given the dimensions of the matrix.
- Parameters: ***lines*** (number of lines) and ***columns*** (number of columns)
- Return value: the matrix whose values were typed by the user.

```
def readMatrix(lines, columns):
    code of the function
```


Function readMatrix

- Sample code for reading A and B:

```
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))
```

```
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))
```


Function readMatrix

- Sample code for reading A and B:

```
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))
```

```
B = np.zeros( (n, p) )
for i in range(0, n):
    for j in range(0, p):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        B[i][j] = float(input(s))
```

Each execution changes the value of a different variable. In cases like this, a good strategy is to create a new variable in the function and return this variable.


Function readMatrix

```
def readMatrix(lines, columns):
    M = np.zeros( (lines, columns) )
    for i in range(0, lines):
        for j in range(0, columns):
            s = 'Type element at position [%d][%d]: ' % (i, j)
            M[i][j] = float(input(s))
    return M
```

Function readMatrix

```
def readMatrix(lines, columns):
    M = np.zeros( (lines, columns) )
    for i in range(0, lines):
        for j in range(0, columns):
            s = 'Type element at position [%d][%d]: ' % (i, j)
            M[i][j] = float(input(s))
    return M
```

Compare with the code for reading A:

```
A = np.zeros( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        s = 'Type element at position [%d][%d]: ' % (i, j)
        A[i][j] = float(input(s))
```

Function calcProduct

- Function **calcProduct** returns the product of 2 given matrices.


Function calcProduct

- Function **calcProduct** returns the product of 2 given matrices.
- Parameters: ***M1*** (first matrix) and ***M2*** (second matrix)
- Return value: the product M1 x M2.

Function calcProduct

- Function **calcProduct** returns the product of 2 given matrices.
- Parameters: ***M1*** (first matrix) and ***M2*** (second matrix)
- Return value: the product M1 x M2.

```
def calcProduct (M1, M2):
    code of the function
```

Function calcProduct

- Sample code for the product AxB:

```
C = np.zeros( (m, p) )
for i in range(0, m):
    for j in range(0, p):
        for k in range(0, n):
            C[i][j] += A[i,k] * B[k,j]
```


Function calcProduct

```
def calcProduct(M1, M2):
    m, n = M1.shape
    n, p = M2.shape
    M3 = np.zeros( (m, p) )
    for i in range(0, m):
        for j in range(0, p):
            for k in range(0, n):
                M3[i][j] += M1[i,k] * M2[k,j]
    return M3
```