FACULTAD DE INGENIERÍA
UNICEN • OLAVARRÍA

Fundamentos de la Programación y Métodos Numéricos

CLASE 5.
5.2.2. Funciones en Python

Introducción

Una función es un **bloque de código que realiza alguna operación**. Por ejemplo, consideremos el caso un caso visto previamente:

```
type("32") // <type 'string'>
```

La función es **type** muestra el **tipo de dato** de un valor o de una variable. El valor o variable se denomina llamado el **argumento o parámetro** de la función. En general una función "toma" un argumento y "devuelve" un resultado. El resultado se llama **valor de retorno**.

Creación de nuevas funciones

La creación de nuevas funciones para resolver sus problemas particulares representa la mayor utilidad de los lenguajes de programación.

La sintaxis de la definición de una función es:

def nombre_funcion(<parámetros>):
    <sentencias>

- **nombre_funccion**: nombre identifica torio de la función, no puede usarse palabras reservadas de Python.
- **<parámetros>**: lista de parámetros necesarios para la ejecución de la función.
- **<sentencias>:** secuencia de instrucciones de ejecución de la función.

Parámetros

La mayoría de las funciones precisan parámetros o argumentos, en otras palabras, valores que controlan como la función lleva a cabo su tarea.

```
def suma(s1, s2):
        print (s1+s2)

suma(5,6)
```

La función suma contiene dos parámetros **s1** y **s2** que son utilizados para resolver la lógica de la función.

Valores de retorno

La sentencia return: permite retornar un valor o terminar la ejecución de una función antes de alcanzar su final. Una razón para usarla es detectar una condición de error:

```
import math
def imprimeLogaritmo(x):
        if x <= 0:
                print ("Solo numeros positivos, por favor." )
                return
        result = math.log(x)
        print ("El log de x es", result)
imprimeLogaritmo(x)
```

La función **imprimeLogaritmo** toma un parámetro llamado **x**. Lo primero que hace es comprobar si **x** es menor o igual que **cero**, si es **TRUE** muestra un mensaje de error y luego usa **return** para salir de la función, caso contrario, calcula el logaritmo e imprime el resultado.

Valores de retorno

Ejemplo de calculo del valor absoluto:

```
def valorAbsoluto(x):
        if x < 0:
                return -x
        return x
```

Invocacion de la funcion:

```
print(valorAbsoluto(10)) // imprime el valor de 10.

print(valorAbsoluto(-10)) // imprime el valor de 10.
```

Funciones booleanas

Las funciones pueden devolver valores booleanos, lo que frecuentemente es conveniente para ocultar complicadas comprobaciones dentro de funciones.

Por ejemplo:

```
def esDivisible(x, y):
        if (x % y == 0):
                return 1 // Valor de retorno true
        else:
                return 0  //Valor de retorno false
```

La función **esDivisible** devuelve **1** o **0** para indicar si la **x** es o no divisible por **y**.

Funciones con Matrices

MULTIPLICACION ENTRE MATRICES:

```
def mult_matrices(A, B):
    C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                C[i][j] += A[i][k] * B[k][j]
    return C


# A = 2x2 B = 2x2 C = 2x2
A = [[1, 2], [3, 4]] # 2x2
B = [[5, 6], [7, 8]] # 2x2
# A = 3x3 B = 3x1 C = 3x1
#A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#B = [[5], [6], [7]]

print (mult_matrices(A, B))
```

Funciones con Matrices

```
import numpy as np

# A = 2x2 B = 2x2 C = 2x2
A = [[1, 2], [3, 4]] # 2x2
B = [[5, 6], [7, 8]] # 2x2
# A = 3x3 B = 3x1 C = 3x1
#A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#B = [[5], [6], [7]]

# Multiplicar las matrices A y B
C = np.dot(A, B)
print(C)
```

NumPy (acrónimo de "Numerical Python") es una biblioteca de Python que se utiliza para trabajar con arreglos numéricos.

Funciones con Matrices

TRASPUESTA DE UNA MATRIZ:

```
def traspuesta(matriz):
        T = [[0 for j in range(len(matriz))] for i in range(len(matriz[0]))]

        for i in range(len(matriz)):
            for j in range(len(matriz[0])):
                    T[j][i] = matriz[i][j]
        return T

A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] # 3x3

print (traspuesta(A))
```

Funciones con Matrices

```
import numpy as np

T = np.transpose(A)

print (T)
```

Funciones con Matrices

IDENTIDAD DE UNA MATRIZ:

```
def matriz_identidad(n):
    # Función para crear una matriz identidad de tamaño n x n
    identidad = [[0 for x in range(n)] for y in range(n)]
    for i in range(n):
        identidad[i][i] = 1.0
    return identidad

print(matriz_identidad(3))

import numpy as np

# Crear una matriz identidad de 3x3
I = np.eye(3)
print(I)
```

Funciones con Matrices

INVERSA DE UNA MATRIZ:

```
import numpy as np

A = [[1, 2], [3, 4]]

I = np.linalg.inv(A)

print(I)
```