Métodos numéricos

Mauricio Suárez Durán
Departamento de Física y Geología
Universidad de Pamplona

Unidad 2, Clase 2
Introducción a Python

I Semestre, 2020

Introducción a Python

- Objetivo:
  - Aprender a importar librerías
  - Aprender a manejar contenedores en Python

Introducción a Python

- Librerias y/o paquetes
  - Cómo importar paquetes?
    - from math import log, pi, sin
    - from math import *
    - import math
    - import math as mt

Introducción a Python

- En particular
  - import math as mt
    - mt.log, mt.exp
    - mt.log10
    - mt.sin, mt.cos, mt.sin (Simpre en radianes)
    - mt.pi
    - mt.sqrt

Introducción a Python

- Módulos
  - Sub paquete de un paquete. Ejemplo:
    - from numpy.random import rand
      (genera un número aleatorio entre 0. y 1.)

- Funciones:
  - x = input(“Ingrese el valor de h: ”)

Introducción a Python

- Ejercicios:
  - Escriba un código que transforme de coordenadas polares a coordenadas cartesianas.

Introducción a Python

- Ejercicios:
  - Una nave espacial viaja desde la Tierra, en linea recta y a una fracción de c, a otro planeta ubicado a x años luz. Escriba un programa que le pregunte al usuario x y v (como una fracción de c), y eventualmente calcule el tiempo que le toma a la nave especial en llegar al planeta visto desde (a) la Tierra y (b) desde la nave espacial.

Introducción a Python

- Control de ejecución
  - Ejemplos:
    - if x > a:
      haga algo
    - if x == a:
      hace algo
    - if x >= a:
      hace algo
    - if x != a:
      - hace algo

Introducción a Python

- if:
  - Ejemplos:
    - if x > a or x < b:
      haga algo

    - if x == a and y == b:
      hace algo

Introducción a Python

- if - else:
  - Ejemplos:
    - if x > a:
      haga algo
      else:
      print(“yucas, try again”)

Introducción a Python

- if - else:
  - Ejemplos:
    - if x > a:
      haga algo
      elif:
      print(“sigue intentando”)
      else:
      print(“yucas, try again”)

Introducción a Python

- while:
  - Ejemplos:
    - while x > a:
      haga algo

    - while x > a or/and x < b:
      haga algo

Introducción a Python

- break and continue:
  - Ejemplos:
    - x = input(“ingrese un número menor a 10: ”)
    - while x > 10:

```
print(“esto es mayor a 10. Intente de nuevo”)
x = input(“ingrese un número menor a 10: ”)
if x == 111:
break
```

Introducción a Python

- Ejemplos, número pares e impares:
  - n = int(input(“ingrese primer entero: ”))
  - m = int(input(“ingrese segundo entero: ”))
  - while (n+m)%2 == 0:

    print(“Ingreso al menos un número impar”)

    n = int(input(“ingrese primer entero: ”))

    m = int(input(“ingrese segundo entero: ”))

Introducción a Python

- Ejercicio: escriba un código que calcule la secuencia de Fibonacci hasta 1000.
  - $f_1 = 1$; $f_2 = 1$
  - $f_n = f_{n-1} + f_{n-2}$; $n > 2$

Introducción a Python

- Contenedores: listas
  - [2, 3, -5 , 10]
  - [2.0, 3, -5.23, 10]
  - [1, 3.3, 2 + 3j, 15.]
  - [1, 3.3, 2 + 3j, 15., “hi”]
  - [x**2, x*y, x/6]

Introducción a Python

- Contenedores: listas
  - x = [2.0, 3, -5.23, 10]
  - x[2] = 500.
  - print(x)
  - total = sum(x)
  - print(x)
  - print(sum(x) / len(x), max(x), min(x))

Introducción a Python

- Contenedores: listas
  - Iteradores (función map):
    - from math import log
    - r = [1., 1.5, 2.2]
    - logr = list(map(log, r))
    - print(logr)

Introducción a Python

- Contenedores: listas
  - Agregando elementos a una lista
    - r.append(1.8)
  - append se puede usar para agregar elementos a una lista vacía
    - r = []
    - r.append(2.4)
    - r.append(2.9)
    - r.append(3.1)

Introducción a Python

- Contenedores: listas
  - Removiendo elementos de una lista:
    - r.pop() # remueve el último elemento
    - print(r)
    - r.pop(i) # remueve el elemento i

Introducción a Python

- Contenedores: arreglos
  - Diferencias respecto a las listas:
    - El número de elementos es fijo. No se pueden agregar nuevos elementos o removerlos.
    - Todos los elementos del arreglo deben ser del mismo tipo y el tipo de elementos no se puede cambiar.

Introducción a Python

- Contenedores: arreglos
  - ventajas respecto a las listas:
    - Pueden ser de dos dimensiones (matrices). De hecho pueden tener cualquier dimensión.
    - Se comportan como vectores y matrices: aplica álgebra matricial.
    - Se procesan más rápido que las listas.

Introducción a Python

- Contenedores: arreglos
  - Como deben ser inicializados, la librería numpy incluye algunas funciones que permiten inicializar.
    - from numpy import zeros
    - a = zeros(4, float)
    - print(a) # Notar que en la salida no hay comas.

Introducción a Python

- Contenedores: arreglos
  - from numpy import zeros
  - a = zeros([3, 4], float) # Arreglo 2D
  - print(a)
  - Se pueden crear arreglos vacíos:
    - a = empty(5, float)

Introducción a Python

- Contenedores: arreglos
  - Se pueden crear a partir de una lista:
    - r = [1., 1.5, -2.2]
    - a = array(r, float)
    - print(a)
  - Si los elementos, o algún elemento, de la lista es un entero, los convierte en flotantes.
  - a = array([1., 1.5, -2.2, 50, 100], float)

Introducción a Python

- Contenedores: arreglos
  - a = array([1., 1.5, -2.2, 50, 100], int)
  - b = array([[1,3,2],[4,5,6]], int) # el número de columnas debe ser igual
  - Los elementos se acceden de manera similar a las listas
    - print(a[1])
    - print (b[1,0], b[0,1])

Introducción a Python

- Contenedores: arreglos
  - Leyendo arreglos desde un archivo:
    - form numpy import loadtxt
    - a = loadtxt(“data.dat”, float)
    - print(a)
  - De manera equivalente para matrices

Introducción a Python

- Contenedores: arreglos
  - Aritmética de arreglos 1D:
    - a[0] = a[1] +10
    - from numpy import array
    - a = array([1,3,4], int)
    - b = 2*a
    - print(b)

Introducción a Python

- Contenedores: arreglos
  - Aritmética de arreglos 1D:
    - from numpy import array
    - a = array([1,3,4], int)
    - b = 2*a
    - print(b)
    - print(a+1)

Introducción a Python

- Contenedores: arreglos
  - Aritmética de arreglos 1D:
    - from numpy import array
    - a = array([1,3,4], int)
    - b = 2*a
    - print(b)
    - print(a+1)

Introducción a Python

- Contenedores: arreglos
  - Aritmética de arreglos 1D:
    - from numpy import array, dot
    - a = array([1,3,4], int)
    - b = 2*a
    - print(a*b, dot(a,b))

Introducción a Python

- Contenedores: arreglos
  - Aritmética de arreglos nD:
    - a = array([[1,3],[2,4]], int)
    - b = array([[4,-2],[-3,1]], int)
    - c = array([[1,2],[2,1]], int)
    - print(dot(a,b)+2*c)
  - Python multiplica vector por matriz en las formas dot(v,a) y dot(a,v); reconoce cuando el vector v es columna o fila.

Introducción a Python

- Contenedores: arreglos
  - La función map también aplica a los arreglos (solo para arreglos 1D):
    - b = array(list(map (sqrt,a)), float)
  - Para arreglos 2D, las funciones len, max y min, no aplican. En su lugar se tiene:
    - a = array([[1,3],[2,4]], int)
    - print(a.size) # Retorna el total de elementos
    - print(a.shape) # Retorna la dimensión del arreglo

Introducción a Python

- Contenedores: arreglos
  - from numpy import array
  - a = array([1,1], int)
  - b = a
  - a[0] = 2
  - print(a)
  - print(b)

Introducción a Python

- Contenedores: arreglos
  - from numpy import array
  - a = array([1,1], int)
  - b = copy(a)
  - a[0] = 2
  - print(a)
  - print(b)