Introducción a Python y bibliotecas Scipy
Nivel II

LABI
Facultad de Ingeniería de la UBA

Segundo cuatrimestre, 2019

Plan

Anaconda

Es un conjunto de herramientas que incluye interpretes actualizados de los lenguajes Python y R.

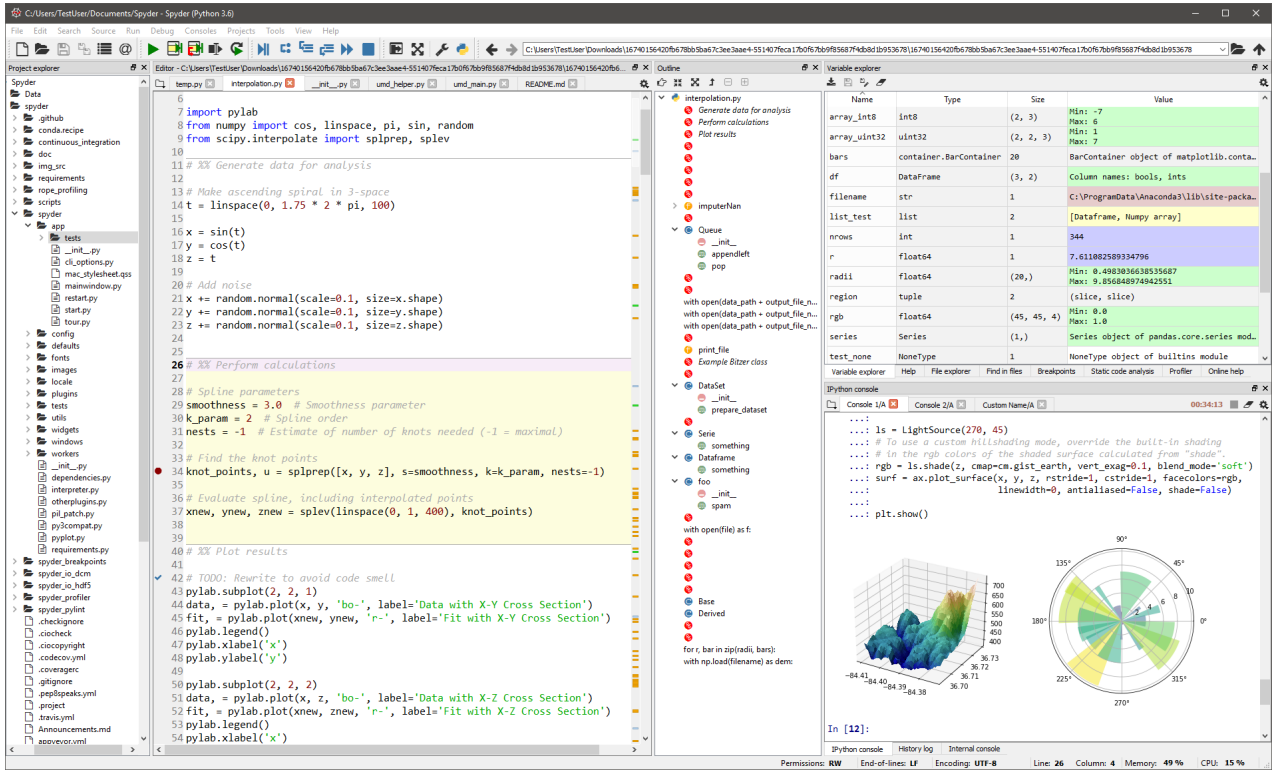
- Incluye bibliotecas científicas para el análisis de datos como numpy, scipy, matplotlib, etc.
- Incluye el IDE Spyder
- Incluye Jupyter Notebook
- Posee un manejador de ambientes de Python

Guía de instalación de Anaconda en linux

https://www.digitalocean.com/community/tutorials/how-to-install-anaconda-on-ubuntu-18-04-quickstart

Anaconda: Spyder

Spyder es un IDE para desarrollo en Python con consola interactiva y explorador de variables.



Clases y objetos

- Son conceptos que pertenecen al paradigma de la programación orientada a objetos.
- Permiten modelar entidades del mundo real que son capaces de interactuar entre sí.
- La programación orientada a objetos es una forma de pensar el desarrollo de aplicaciones distinto a la programación estructurada.

Clases y objetos

Clase: Representación de una entidad del mundo real que posee propiedades y métodos.

Ejemplo de clase: Alumno_de_Fiuba

- Propiedades:
  - Nombre : String
  - Apellido : String
  - Padrón : Entero
- Métodos
  - cursar_materia
  - tomar_apuntes

Clases y objetos

__init__ Es la función que se ejecuta al momento de crear el objeto. El primer argumento siempre es el propio objeto.

Código

```
class alumno_Fiuba:
    def __init__(self, _nombre, _apellido, _padron):
        self.nombre = _nombre
        self.apellido = _apellido
        self.padron = _padron
    def imprimirt_padron(self):
        print(self.padron)
```

Clases y objetos

Objetos : Es la realización de una clase.

Ejemplo de un objeto de la clase Alumno_de_Fiuba:

- alumno.Nombre = "Franco"
- alumno.Apellido = "Nastasi"
- alumno.Padron = 100002

Clases y objetos

Código

```
alumno =alumno_Fiuba("Franco", "Nastasi",100002)
print(alumno.nombre)
print(alumno.apellido)
print(alumno.padron)
alumno.padron = 97825
print(alumno.padron)
alumno.imprimir_padron()
```

Clases y objetos - Ejercicio

Crear una clase para representar vectores en el espacio de $\mathbb{R}^3$ con las funciones:

- **es_unit**: Devuelve un booleano indicando si es un vector de mdulo 1
- **prod_int**: Devuelve el producto interno del objeto con otro objeto vector
- **es_ort**: Devuelve un booleano indicando si es ortogonal a otro vector

NumPy

- Es uno de los paquetes fundamentales para el análisis numérico.
- Permite el manejo de arreglos multidimensionales.
- Utiliza el tipo '*numpy.ndarray*' sobre el cual están definidas las operaciones del álgebra lineal.

NumPy

Importar el paquete

```
import numpy as np
```

Contantes de interés

```
np.pi
np.e
```

Creación de arreglos

```
np.arange(5) # Crea un arreglo NumPy de 0 a 4
np.arange(5,15) # Crea un arreglo NumPy de 5 a 14
np.arange(1,15,2) # Crea un arreglo NumPy de 1 a 14 de a
pasos de 2
np.array([1,3,5]) # Convierte una lista en un arreglo NumPy
```

NumPy

Creación de arreglos vacíos - zeros

x = np.zeros(5)
x = np.zeros(shape = (2,3))

Creación de matrices - reshape

x = np.arange(9)
x = x.reshape(3,3)
x = np.array([1,3,5,7,9,11]).reshape(2,3)

Operaciones sobre arreglos NumPy - elemento a elemento

+ , - , * , / , %

NumPy

Operaciones lineales sobre arreglos NumPy - dot

```
x = np.arange(3)
y = np.arange(1,6,2)
np.dot(x,y) # Productor interno usual de los espacios R^n
A = np.arange(9).reshape(3,3)
B = np.arange(3)
np.dot(A,B)
```

Ejercicio: Simulación

Aplicar el siguiente filtro pasa alto a una señal senoidal de amplitud 1 V y frecuencia 50 Hz, con un valor de continua de 1 V.
El filtro tiene una respuesta:

$$h(n) = \delta(n) - \delta(n-1)$$

Por lo tanto solo hay que calcular la salida del filtro como:

$$y(n) = x(n) - x(n-1)$$

NumPy

Creación de arreglos - linspace

```
x = np.linspace(1,2,num = 10, endpoint=False)
x = np.linspace(1,2,num = 10)
```

Funciones matemáticas

```
x = np.linspace(0,2*np.pi,num = 10, endpoint=False)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)
y4 = np.exp(x)
y5 = np.log(x)
y6 = np.log10(x)
```

matplotlib

- Es una biblioteca que brinda herramientas para graficar datos en 2-D y 3-D
- Permite un mayor control sobre las propiedades de las figuras.

Importar el paquete

```
import matplotlib.pyplot as plt
```

Realizar un gráfico simple

plt.plot(x,y) x e y son arreglos de numpy

Ejercicio - cargar datos de una simulación y graficarlos

En un archivo *mono_etapa.txt* están los datos de una simulación realizada en el programa Ltspice. Se pide generar un gráfico para incluirlo en un informe.

NumPy

Leer un archivo CSV - loadtxt

```
file = "nombre_archivo.txt"
data = np.loadtxt(file, delimiter = ’
t’, skiprows=1)
```

Recorrer matrices - operador :

```
A = np.arange(9).reshape(3,3)
A[0] # Devuelve la primer fila
A[0,0:2] # Devuelve los 2 primeros elementos de la primer fila
A[0, :] # Devuelve la primer fila
A[:,0] # Devuelve la primer columna
A[:,1] # Devuelve la segunda columna
A[1:3,1:3] # ¿Qué devuelve?
```

matplotlib

- Es una biblioteca que brinda herramientas para graficar datos en 2-D y 3-D
- Permite un mayor control sobre las propiedades de las figuras.

Importar el paquete

```
import matplotlib.pyplot as plt
```

matplotlib

Generación de figuras - subplots

```
fig , ax= plt.subplots()
```

Generación de gráficos - plot

```
x = np.linspace(0,np.pi, num = 20)
fig,ax= plt.subplots()
ax.plot(x, np.sin(x),color='r', label=s̈en(x)")
ax.plot(x, np.cos(x),color='b', label=c̈os(x)")
```

matplotlib

Generación de gráficos - semilogx

```
x = np.linspace(0,1e10, num = 20)
fig,ax= plt.subplots()
ax.semilogx(x, np.log10(x), label=Ëscala logaritmica ¨)
```

Establecer títulos y etiquetas para los ejes

```
ax.set(xlabel=Ëtiqueta eje x [unidades]¨)
ax.set(ylabel=Ëtiqueta eje y [unidades]¨)
ax.set(tittle = "Título")
```

matplotlib

Leyendas

```
ax.legend(loc = "best")
# Visualiza la leyenda en la posición especificada por loc.
# Toma el valor de 'label' de cada gráfico como leyenda.
```

Cuadrícula

```
ax.grid() # Visualiza una cuadrícula dentro del gráfico.
```

matplotlib

Parámetros para modificar estilo de gráficos

- color: ’r’ , ’b’ , ’g’ , ’m’ , ’c’ , ’y’ , ’k’ , ’w’
- marker: ’o’ , ’s’ , ’<’ , ’>’ , ’h’ , ’H’ , ’1’ , ’2’ , ’3’
- linestyle: ’-’ . ’–’ . ’-.’ . ’:’

ax.plot(x , np.sin(x), color = ’b’, marker= ’s’, linestyle = ’–’)
ax.plot(x , np.sin(x), ’bs–’)

Guardar figura

fig.savefig("nombre_figura.fmt", dpi = 300)

SciPy

SciPy: Es una de las bibliotecas más útiles para realizar análisis numérico de datos.

Incluye las siguientes funcionalidades:

- Transformada de Fourier: FFT, DST, DCT
- Procesamiento de señales: Aplicación de filtros
- Álgebra lineal: Solución por cuadrados mínimos, resolución de sistemas lineales, descomposición QR, etc
- Funciones especiales: Funciones de Bessel
- Escribir y leer archivos WAV

SciPy

Calcular FFT

```
from scipy.fftpack import fft, ifft
import numpy as np
x = np.array([1.0, 2.0, 1.0, -1.0, 1.5])
y = fft(x)
```