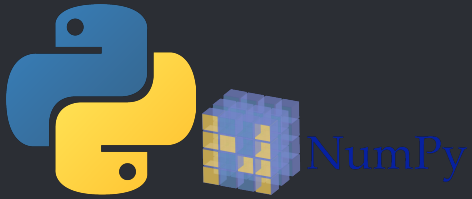
FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Procesamiento de señales, fundamentos

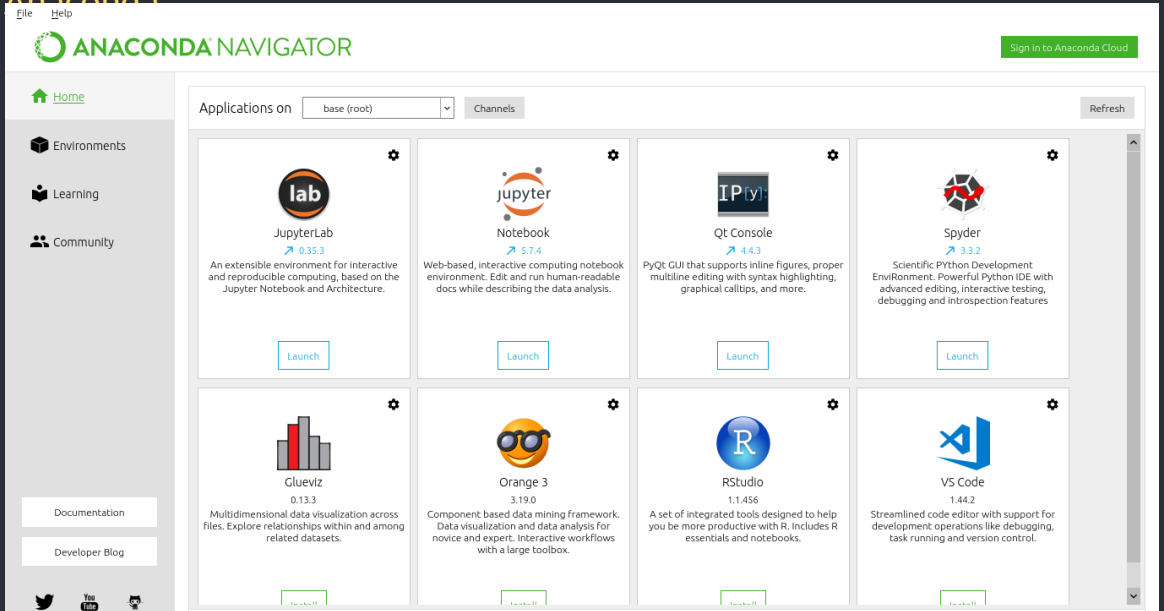
Maestría en sistemas embebidos
Universidad de Buenos Aires
MSE 5Co2020

Introducción a Python y NumPy



Ing. Pablo Slavkin
slavkin.pablo@gmail.com
wapp:011-62433453

Anaconda


File
Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
Refresh
JupyterLab
0.35.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
5.7.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Qt Console
4.4.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
Spyder
3.3.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Orange 3
3.19.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
VS Code
1.44.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Anaconda

jupyter-notebook

Engineered Arts Cha | (1) WhatsApp | Home | Untitled

localhost:8889/notebooks/Untitled.ipynb?kernel_name=python3# 110%

jupyter Untitled Last Checkpoint: 5 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

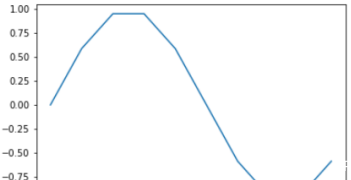
Trusted | Python 3

Run | Markdown

Esto es un notebook

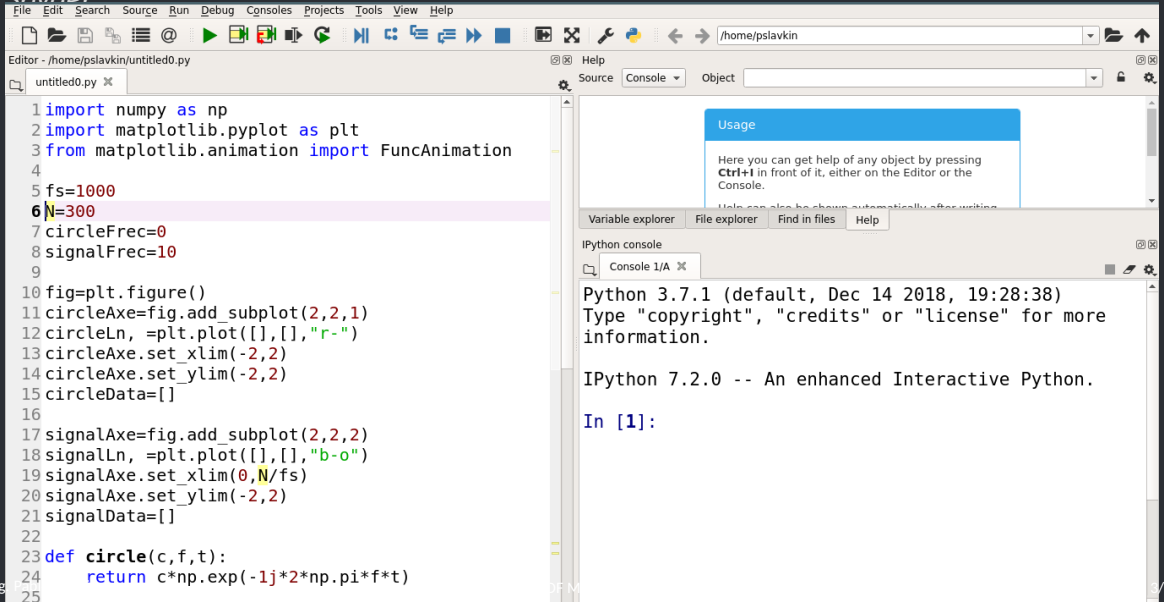
In [9]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 t=np.arange(0,1,0.1)
5 signal=np.sin(2*np.pi*t)
6
7 plt.plot(t,signal)
8 plt.show()
9
```



Anaconda

Spyder



Anaconda

Ipython

```
pslavkin@work1:/h/p/m/clases$ ipython3
/usr/local/lib/python3.7/site-packages/IPython/core/history.py:226: UserWar
ning: IPython History requires SQLite, your history will not be saved
  warn("IPython History requires SQLite, your history will not be saved")
Python 3.7.3 (default, Nov  5 2019, 00:08:28)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.9.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: a=[1,2,3]

In [2]: print(a)
[1, 2, 3]

In [3]:
```

python

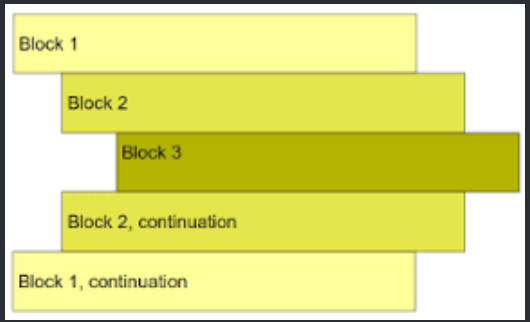
python

```
pslavkin@work1:/h/pslavkin$ python3
Python 3.7.3 (default, Dec 20 2019, 18:57:59)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a="soy python"
>>> print(a)
soy python
>>>
```

Introducción

- Lenguaje Interpretado
- Tipado dinámico: El tipo se define en tiempo de ejecución
- Fuertemente tipado: Durante las operaciones se chequea el tipo

Indentación




```
mi_variable = 27
mi_flag     = True

if mi_flag:
    while mi_variable > 0:
        print(mi_variable)
        mi_variable-=1
```

Números

- Enteros: Con signo, sin límite
- Punto flotante
- Complejos
- Booleanos

```
nint     = 27
nfloat   = 3.14
ncomplex = 1+2j

print(type(nint))
print(type(nfloat))
print(type(ncomplex))
------------------------
>> <class 'int'>
>> <class 'float'>
>> <class 'complex'>

flag1 = True
flag2 = False

print(type(flag1))
print(flag2)

if flag1:
    print("verdadero!")
------------------------
>> <class 'bool'>
>> False
>> verdadero!
```

Cadenas

- Strings: se guardan codificadas. Ej. UTF8 y son inmutables
- Byte array: sin codificar, raw, son mutables

```
msg = "Hola mundo"
msg = 'Hola mundo'

print(msg)
print(msg[2])
print(msg[5:10])
-----------------------------
>> Hola mundo
>> l
>> mundo

b = bytearray()
b.append(0x02)
b.append(0x10)
b.append(0x05)
b.append(0x10)
b.append(0x03)
print(b)
print(b[3])
print(len(b))
-----------------------------
>> bytearray(b'\x02\x10\x05\x10\x03')
>> 16
>> 5
```

Listas

```
l = [1, 2, 3, 4, 5 ];
print(l)
for i in l:
    print(i)
----------------------
>>
[1, 2, 3, 4, 5]
1
2
3
4
l = [1, 2, 3, "casa", 5.2 ];
print(l[2])
print(l[-1])
print(l[1:4])
----------------------
>> 3
>> 5.2
>> [2, 3, "casa"]
l.append(6)
l.remove(2) #por valor
print(l)
----------------------
>> [1, 3, "casa" , 5.2, 6]
lista = [1, 2, 3, 4, 5 ];
cantidad_elementos = len(lista)
print(cantidad_elementos)
----------------------
>> 5
```

funciones

```
def miFuncion(arg1,arg2,arg3=1):
    print(arg1)
    print(arg2)
    print(arg3)
    return 5,6,7

cinco,seis,siete = miFuncion(1,2)
miFuncion("hola",arg2=2+1j,arg3=3)
```

NumPy

arrays

```
a=np.array([1,2,3])
print(type(a))
print(a)
print(a+1)
-----------------------------
>>numpy.ndarray
>>[1 2 3]
>>[2 3 4]
```

NumPy

linspace, arange

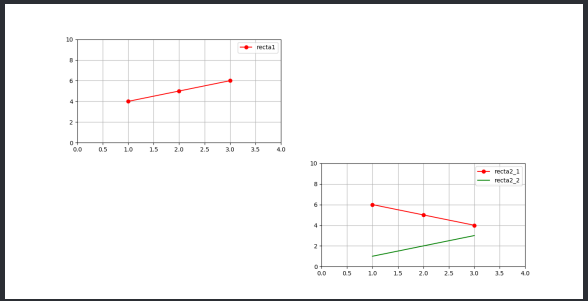
```
import numpy as np

a=np.linspace(0,1,10)
print(a)
b=np.linspace(0,1,10,endpoint=False)
print(b)
c=np.arange(0,1,0.1)
print(c)
print(c[:2])
print(c[2:])
print(c[2:2])
print(c[::2])
print(c[::-1])
print(c[2])
print(c[-2])
for i in c:
    print(i)
------------------------------
```

```
>>[0. 0.11111111 0.22222222 0.33333333 0.44444444
     0.55555556 0.66666667 0.77777778 0.88888889 1.]
>>[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
>>[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
>>[0.  0.1]
>>[0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
>>[]
>>[0.  0.2 0.4 0.6 0.8]
>>[0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0. ]
>>0.2
>>0.8
>>0.0
>>0.1
>>0.2
>>0.30000000000000004
>>0.4
>>0.5
>>0.6000000000000001
>>0.7000000000000001
>>0.8
>>0.9
```

matplotlib

pyplot

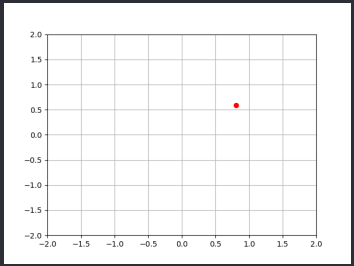
```
import matplotlib.pyplot as plt

fig      = plt.figure()

graph1Axe = fig.add_subplot(2,2,1)
graph1Ln1, = plt.plot([1,2,3],[4,5,6],'r-o')

graph1Axe.grid                    ( True     )
graph1Axe.set_xlim                ( 0,4      )
graph1Axe.set_ylim                ( 0,10     )
graph1Ln1.set_label               ( "recta1" )
graph1legendLn    = graph1Axe.legend (         )

data2_1=[6,5,4]
data2_2=[1,2,3]
t=[1,2,3]
graph2Axe           = fig.add_subplot(2,2,4)
graph2Ln1,graph2Ln2 = plt.plot(t,data2_1,'r-o', t,data2_2,'g-',)

graph2Axe.grid                    ( True       )
graph2Axe.set_xlim                ( 0,4        )
graph2Axe.set_ylim                ( 0,10       )
graph2Ln1.set_label               ( "recta2_1" )
graph2Ln2.set_label               ( "recta2_2" )
graph2legendLn    = graph2Axe.legend (           )

plt.show ( )
```

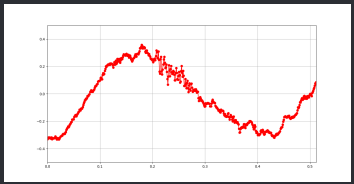


FuncAnimation

Animation

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

fig       = plt.figure ( )
fs        = 10
N         = 10
circleAxe = fig.add_subplot(1,1,1)
circleLn, = plt.plot([],[],'ro')
circleAxe.grid(True)
circleAxe.set_xlim(-2,2)
circleAxe.set_ylim(-2,2)
circleFrec = 1
circle  = lambda c,f,n: c*np.exp(-1j*2*np.pi*f*n*1/fs)

def update(n):
    circleLn.set_data(np.real(circle(1 ,circleFrec ,n)) ,np.
         imag(circle(1 ,circleFrec ,n)))
    return circleLn,

ani=FuncAnimation(fig,update,N,interval=1000 ,blit=False,
      repeat=True)
plt.show()
```





Módulos y paquetes

Módulos:

defino un archivo:

```
mi_modulo.py
# en minuscula
def func1():
    print("f1")
def func2():
    print("f2")
-----------------
import mi_modulo
mi_modulo.func1()
-----------------
>>f1
```

Paquetes:

```
|-- paquete
|   |-- __init__.py
|   |-- mi_modulo1
|   |   |-- mod1_file1.py
|   |-- mi_modulo2
|   |   |-- __init__.py
-------------------------
lo invoco como:
-------------------------
from paquete.mi_modulo1 import mod1_file1
    as m1
m1.mod1_file1_func1()
```