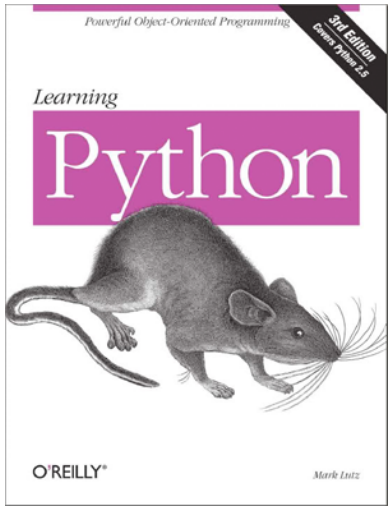
Introduction to Python

YRNCS Workshop

References

- Books: Learning Python (O'Reilly), ...



- Official documentation:

  `https://docs.python.org/2.7/`

  `https://wiki.python.org/moin/BeginnersGuide`
- Learning sites:
  - `http://www.learnpython.org/`
  - OpenClassRooms (french): `http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-python`
  - ...

Introduction

- Two version of Python : **2.7** ; 3.0 and above
- Interpreted language $\Rightarrow$ no compiling
  - Python shell
  - Scripts running (*old school* editor + shell)
  - IDEs (Integrated Development Environments) : CodeBlocks, Idle, ...
  - iPython (in-navigator interpreter)

First steps: Variables

- Dynamic typing for variables

```
a = 2          (integer)
a = 2.         (float)
s = 'hello'    (string)
s = "hello"    (also string)
```

- Only long versions of numbers
- Booleans as binary integers : False = 0, True = 1
- Type change :

```
a = int(2.)      (to integer)
a = float(2)     (to float)
a = str(2)       (to string)
```

- Usual operations on numbers : +, -, *, /, ** (power), % (modulo)
- Very simple concatenation for strings : +

First steps: Lists

- No memory handling, no size definition, dynamic typing

```
Tab = [1,5,3,65]
Tab = []
Tab = ["hello",5,3.3913,True]
```

- Dynamic size redefinition

```
Tab = [1,2,3,4]
Tab.append(5)      ([1,2,3,4,5])
Tab.remove(2)      ([1,3,4,5])
```

- Indexing

```
Tab = [1,2,3,4,5]
Tab[2:]      ([3,4,5])
Tab[:2]      ([1,2])
Tab[2:4]     ([3,4])
Tab[0:5:2]   ([1,3,5])
```

- Utilities : index, sort, ...
- Also valid for higher dimensions

First steps: Tuples, Sets & Dictionaries

- Tuple : immutable list

```
a = (1,45,32)
```

- Set : unsorted list with no duplicates (useful for ensemble operations)

```
a = set([1,45,32])
```

- Dictionary: unsorted list, with custom index (keys and values)

```
Age = {}
Age['Luc'] = 34
Age['Marc'] = 24
Age['Jean'] = 28
Age = {'Luc':34,'Marc':24,'Jean':28}
Age.keys()        (['Luc','Marc','Jean'])
Age.values()      ([34,24,28])
del Age['Luc']    ({'Marc':24,'Jean':28})
```

- . . . and other specific types

Code structure

- Code structured with colons (:) and indentation
  - Conditions

```
if a < b:
    print a
elif (c < b)and(c < a):
    print c
else:
    print b
```

  - Functions

```
def func(a):
    return a
```

  - for loops

```
for i in range(10):
    print i
```

  - while loops

```
while a > 10:
    a = a - 1
    print a
```

- Implicit definitions:

```
Tab = [a**2 for a in range(200) if a%7 == 0]
```

Libraries

- Lots of libraries :
  - more than 38 000 packages on the Python Package Index
  - ∼200 on the official Module index
  - 60 of them have their own Wikipedia page
- math : mathematical operations and constants (exp, log, pi, . . .)
- random : random numbers generators and sampling
- numpy/scipy : scientific tools
- matplotlib/pyplot : plotting
- . . . and many more
- Use:

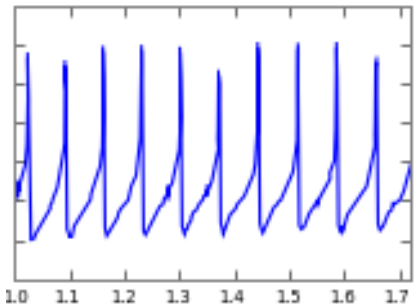
```
from math import log
a = log(10)
import random as rd
b = rd.randint(0,10)
```

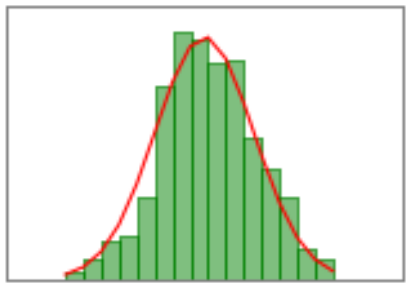
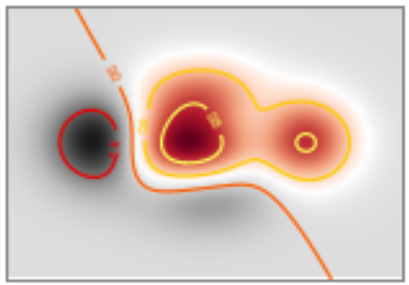
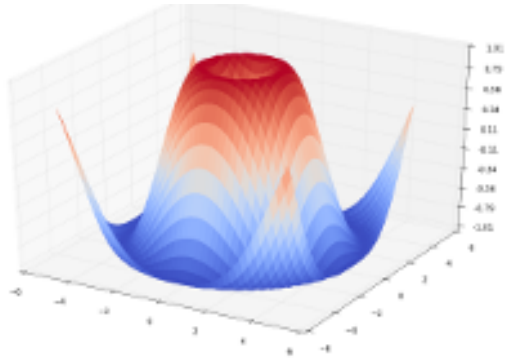
Numpy (http://www.numpy.org/)

- Array creation/manipulations
- Binary operations
- String operations
- Datetime Support Functions
- Data type routines
- Floating point error handling
- Discrete Fourier Transform
- Financial functions
- Functional programming
- Indexing routines
- Input and output
- Linear algebra
- Logic functions
- Mathematical functions
- Matrix library
- Polynomials
- Random sampling
- Sorting, searching, and counting
- Statistics
- ...

Matplotlib (http://matplotlib.org/)

In a few words, you can pretty much do anything you want.



NetworkX (http://networkx.github.io/)

Overview :

- Network creation and manipulation library
- Uses Graph() objects, based on the dictionary type
- Handles undirected, directed and multiplex graphs
- Built-in functions for statistics on networks
- Built-in functions for networks visualization (using matplotlib)
- Use:

```
import networkx as nx
```

Graph creation

- By hand:

```
G = nx.Graph()
G.add_nodes_from([0,1,2,3,4,5,6,7,8,9])
G.add_edges_from([(0,1),(2,3),(4,5),(6,7),(8,9)])
```

- By a generator:

```
G = nx.complete_graph(5)
G = nx.erdos_renyi_graph(100,0.15)
G = nx.watts_strogatz_graph(30,3,0.1)
G = nx.barabasi_albert_graph(100,5)
...
```

Graph manipulation

- Modifications:

```
G.add_node(12)
G.add_edge((34,45))
G.remove_node(12)
G.remove_edges_from([(4,5),(8,9)])
...
```

- Accessing nodes/edges:

```
G.nodes()     (List of the nodes)
G.edges()     (List of the edges)
```

- Graph operations:

```
subgraph(G, nbunch)
union(G1,G2)
disjoint_union(G1,G2)
...
```

Attributes

- Attributes can be attached to graph, nodes, edges:

```
G.add_node(12,nom="John",age=34)
G.node[12]['nom'] = "John"
G.node[12]['age'] = 34

G.add_edge((12,45),weight = 11)
G[12][45]['weight'] = 11
```

- Gathering attributes:

```
nx.get_node_attributes(G,'nom')
nx.get_edge_attributes(G,'nom')
```

- Setting attributes:

```
nx.set_node_attributes(G,'age',0)
nx.set_edge_attributes(G,'weight',1)
```

Graph analysis

- Node degrees

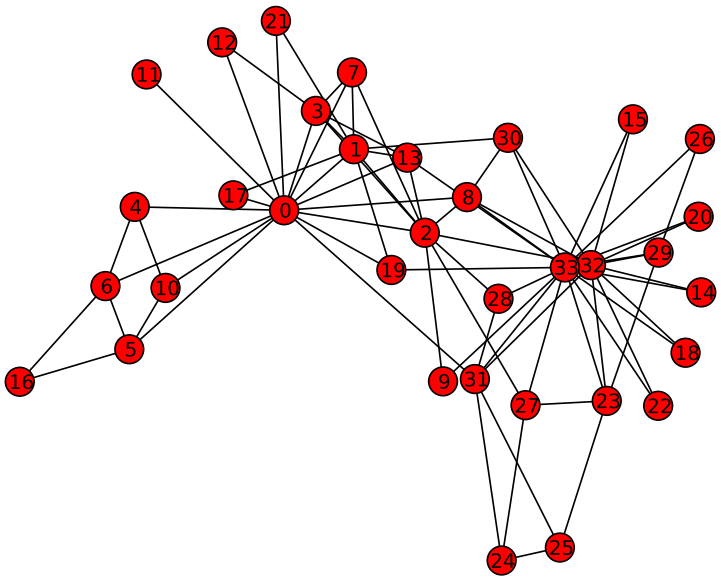
```
nx.degree(G)            (dictionary)
nx.degree(G).values()   (list)
```

- Clustering
- Assortativity
- Centralities (betweenness, closeness, . . .)
- Cliques finding
- Connectivity
- Shortest paths
- . . .

Graph visualization (with Matplotlib)

- Very simple way to plot a Graph:

```
import matplotlib.pyplot as plt
nx.draw(G)
plt.savefig("plot.pdf")
```



Graph visualization (with Matplotlib)

- Very simple way to plot a Graph:

```
import matplotlib.pyplot as plt
pos = nx.circular_layout(G)
nx.draw(G,pos)
plt.savefig("plot.pdf")
```