Practical sessions tools

- Python: a high level interpreted language
- Numpy: a scientific computing libray
- Keras: a high level deep learning library
- Jupyter notebook: A web application allowing to run code on a user-friendly environment

Contents

Python

Python

- High-level language
- Object-oriented with dynamic typing
- Easy to interface with C++
- Easy to learn (?)

We will use Python 3

Contents

Expressions

Examples of expressions

A session in the interactive interpreter:

```
>>> 2
2
>>> 2+2
4
>>> "hello"
'hello'
>>> # this is a comment
```

Variables

Using variables

Variables are not declared:

```
>>> a = 2
>>> a
2
>>> b = 3
>>> print(b)
3
>>> b = a
>>> a = 1
>>> print(a,b)
1 2
```

Compound types: tuples and lists

Tuples

```
>>> t1 = ( 1, "zz")
>>> type(t1)
<type 'tuple'>
>>> print(t1, t1[0], t1[1])
(1,"zz") 1 "zz"
>>> a,b = t1
>>> print(a)
1
>>> print(b)
'zz'
```

Tuples are inmutable

Once they are created, the can only be erased. They cannot be modified.

Compound types: tuples and lists

Lists

```
>>> l1=[ "abc" ]
>>> l1.append(3.14159) # lists can be heterogeneous
>>> l1 [ 'abc', 3.14159 ]
>>> l1.append( [ 1, t1 ] ) # lists can contain lists
>>> l1 ['abc', 3.14159, [1, (1, 2)]]
>>> print(l1[1]) # indexing starts at 0
3.14159
```

Lists are mutable

You an modify or erase any list element.

Contents

Functions (1)

Declaring a function

In the interpreter:

```
>>> def f(a): # note the ':' (colon)
... return a*2 # there are 4 spaces or 1 tab
# before the "return"
...
>>> f("abcd")
'abcdabcd'
```

Indentation is *critical*: it's part of the syntax

Fonctions (2)

Return value

```
def theAddition( a , b ):
     return a + b
```

Contents

Tests

Tests and conditions

```
if 1 == 2 : # note the ':' !
     print("Wow, Problem!") # <- 4 spaces or 1 tab

if myGlass.isFull(): # note ':'
     myGlass.doEmpty()
elif myGlass.isEmpty():
     myGlass.doFill()
else:
     pass # "empty" action
```

Loops (1)

Simple loops

```
for i in range( 5 ):
    print(i)
```

Will display:

```
0
1
2
3
4
```

Loops (2)

Loops on lists

To loop over the items in a list, 2 possibilities:

```
L = [ 1 , 2 , "abc" ]
for i in range( len(L) ):
     print(L[i])
```

which is equivalent to:

```
for x in L:
     print(x)
```

Python style

The second construct is more python-ish that the first, and should be preferred.

Loops (3)

"while" loops

Infinite "while" loop:

```
a = 0
while True:
      a += 1
      if a == 10:
            break
```

"while" loop with a test:

```
while a < 10:
      a+=1
```

Contents

Packages

Importing packages

```
>>> import numpy
>>> numpy.cos(0)
1.0
```

You can also use a short name for the package:

```
>>> import numpy as np
>>> np.cos(0)
1.0
```

For the lazy ones (bad practice, just for quick tests!):

```
>>> from numpy import *
>>> cos(0)
1.0
```

Contents

Files

Reading and writing (text mode)

To read the content of a text file:

```
>>> f = open( "hop.txt" , "r" )
>>> for l in f.readlines():
        print(l)
```

To write:

```
>>> f2 = open( "hop2.txt" , "w" ) # "a" for "append"
>>> f2.write( "Hello, world!" + str( 3.14 ) + "\n" )
>>> f2.close() # optional
```

Paths

Manipulating paths

In the **os** package:

```
>>> import os
>>> os.getcwd() # current directory
'/home/romain'
>>> os.path.join( os.getcwd() , "hop" , "hip" )
'/home/romain/hop/hip'
```

Paths (2)

Packages import paths

In the file `/home/matthieu/Python/myOwnProgram.py`

```
def hello():      return "bonjour"
```

In the main program, using the **sys** package:

```
>>> import sys
>>> sys.path.append("/home/matthieu/Python")
>>> import myOwnProgram
>>> print(myOwnProgram.hello())
'bonjour'
```

Contents

Numpy

- A python module for scientific computing
- Based on a powerfull multi-dimensional array object
- Efficient core coded in C++

Contents

Contents

The end

Good Luck !