cours1

October 15, 2018

1 Installation de Python

- On utilise la version 3 de Python (3.6 est la version courrante)
- Quelques différences avec la version 2 de Python (version 2.7 est la version courrant)
- Distribution : Anaconda. Installation de tout les package. Linux, Mac OS et Windows
- Utilisation de Python :
  - Console : "ipython" ou "jupyter console"
  - Spyder : environnement à la matlab
  - Jupyter notebook : pratique pour des petits exemples. Ne permet pas de faire un projet en Python

2 1. Les types d'objets en Python

3 Les nombres

- entier (taille infinie)
- flottant
- complexe

```
In [16]: z=(2+1j)**2
         z.real
         z.imag
```

```
In [81]: a = True
         b = False
         a|b
         a or b

         from math import sqrt

         (True and False) or True
```

```
Out[81]: True
```

4 Les chaînes de caratères

- création (“, ’, et”””)
- quelques méthodes utiles

```
In [39]: "pipi"
         'opoipoipoipoi'
         "Aujourd'hui"
         s='Aujourd\'hui'
         s = """mklmk

         lklkjl


         lkjljk"""
         print(s)

         s = 'α'

         # oubliez ce qui suit
         α = 1/137.035
         print(α)

         s='aujourd\'hui'
         s[2]
         s.upper()

         s.endswith('ui')

         from math import pi, e
         s='La valeur de pi est {:.3f} et celle de e est {}'
         s.format(pi, e)
```

```
mklmk

lklkjl


lkjljk
0.007297405772247966
```

```
Out[39]: 'La valeur de pi est 3.142 et celle de e est 2.718281828459045'
```

5 Les listes

- création
- modification

- les boucles (enumerate et zip)
- les list comprehension

```
In [49]: l = []
         l.append(3)
         ll = l
         ll.append(5)
         print(l)

         for i, elm in enumerate(ll):
             print(i, 2*elm)

         autre_liste = []
         for elm in ll:
             autre_liste.append(elm**2)

         autre_liste = [elm**2 for elm in ll if elm>4]
         autre_liste

         l1 = ['Pierre', 'Jacques', 'Jean']
         l2 = [34, 56, 31]

         for nom, age in zip(l1, l2):
             print(nom, age)
```

```
[3, 5]
0 6
1 10
Pierre 34
Jacques 56
Jean 31
```

```
In [63]: from math import sin, cos

         l = [sin, cos]

         for function in l:
             print(function(4))

         l = [[1]]
         l[0][0]

         l = [2]
         l.append(l)
         print(l)
         l[1][1][1][0]
```

```
-0.7568024953079282
-0.6536436208636119
```

```
[2, [...]]
```

```
Out[63]: 2
```

6 Les n-uplets (tuple)

- exemple d'utilisation
- avantage par rapport aux listes

```
In [55]: a = (1, 2, 4)
         b, _, d = a
         print(d)
```

```
4
```

7 Les dictionnaires

- création
- boucles

```
In [67]: dico = {'Pierre':4}
         dico['Pierre']
         dico['Jacques'] = 34
         dico

         for nom, age in dico.items():
             print(nom, age)
```

```
Pierre 4
Jacques 34
```

8 Les ensembles

- création
- operation sur les ensembles

```
In [72]: ens1 = {1, 2, 5}
         ens2 = {2, 3, 5}
         ens1 | ens2

         s = "operation sur les ensembles"
         for elm in set(s):
             print(elm, s.count(elm))
```

```
  3
s 4
r 2
n 2
t 1
b 1
p 1
i 1
u 1
m 1
a 1
l 2
o 2
e 5
```

9 2. Elements de syntaxe

10 Arguments optionels

```
In [92]: def ma_function(a, b=3, c=3, temperature=25):
             return (a + b + 2*c)*temperature/25

         ma_function(1)

         ma_function(c=5, a=2)

         g=ma_function
         g(1)
```

```
Out[92]: 10.0
```

11 Fonction lambda

```
In [97]: (lambda x:x**2)(2)
         lambda x, y, z:x+y+z
```

```
Out[97]: <function __main__.<lambda>>
```

12 Exemple : fonction quad de scipy

```
In [100]: from scipy.integrate import quad
          quad(lambda x:x**2, 0, 1, epsrel=1E-12)
```

```
Out[100]: (0.33333333333333337, 3.700743415417189e-15)
```

13 Les boucles

- Boucles for (liste, générateur). range en Pyhon 3
- break, continue
- else

```
In [108]: l = [1,2,3]
          for elm in range(10000000000):
              if elm>5:
                  break
              print(elm)
```

```
0
1
2
3
4
5
```

```
In [111]: def mon_range(N):
              i = 0
              while(i<N):
                  print('A')
                  yield i
                  print('B')
                  i = i + 1

          for elm in mon_range(10):
              print(elm)
```

```
A
0
B
A
1
B
A
2
B
A
3
B
A
4
B
A
5
```

```
B
A
6
B
A
7
B
A
8
B
A
9
B
```

```
In [114]: def coordonnees(N, M):
              for i in range(N):
                  for j in range(M):
                      yield i, j

          for ij in coordonnees(3, 3):
              print(ij)

          list(coordonnees(3, 3))
```

```
(0, 0)
(0, 1)
(0, 2)
(1, 0)
(1, 1)
(1, 2)
(2, 0)
(2, 1)
(2, 2)
```

```
Out[114]: [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]
```

```
In [120]: for i in range(30):
              if i>20:
                  break
          else:
              print('COUCOU')
```

14 3. Quelques subtilités de Python

14.1 Assignation

```
In [121]: l = [1, 2]
          ll = l
```

```
In [121]: del l
          ll

Out[121]: [1, 2]
```

14.2 Variable locale / variable globale

```
In [129]: from math import sin
          a = 1

          def f(b=2):
              print(b)
              a = 2
              b = 3
              print(sin(3))

          f()
          print(a)

          #def f():
          #    global a
          #    a = 2
          #print(a)
          #f()
          #print(a)
```

```
2
0.1411200080598672
1
```

15 4. Les erreurs

- Soulever les erreurs
- Tester les erreurs

```
In [139]: l = [1, 2, 3]
          #l(2)

          #sin[1]

          #sqrt(-1)

          #l[10]


          def ma_fonction():
              a = (1 + 3 + sin(3)
```

```
                + 2)
            return a
```

```
In [144]: a = -1
          if a>0:
              b = sqrt(a)
          else:
              b = 1j*sqrt(-a)
          print(b)

          try:
              b = sqrt(a)
          except ValueError:
              b = 1j*sqrt(-a)
          print(b)
```

```
1j
1j
```

```
In [158]: def aire_triangle(a, b, c):
              s = (a+b+c)/2
              try:
                  return sqrt(s*(s-a)*(s-b)*(s-c))
              except ValueError:
                  raise Exception("Le triangle n'existe pas")

          2*aire_triangle(3, 4, 50)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-158-6e22c09501ca> in aire_triangle(a, b, c)
      3     try:
----> 4         return sqrt(s*(s-a)*(s-b)*(s-c))
      5     except ValueError:


ValueError: math domain error


During handling of the above exception, another exception occurred:


Exception                                 Traceback (most recent call last)

<ipython-input-158-6e22c09501ca> in <module>()
```

```
      6         raise Exception("Le triangle n'existe pas")
      7
----> 8 2*aire_triangle(3, 4, 50)


<ipython-input-158-6e22c09501ca> in aire_triangle(a, b, c)
      4         return sqrt(s*(s-a)*(s-b)*(s-c))
      5     except ValueError:
----> 6         raise Exception("Le triangle n'existe pas")
      7
      8 2*aire_triangle(3, 4, 50)


Exception: Le triangle n'existe pas
```

```
In [156]: def f():
              pass

          f()==None

          2*None
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-156-c8911253f405> in <module>()
  4 f()==None
  5
----> 6 2*None


TypeError: unsupported operand type(s) for *: 'int' and 'NoneType'
```

```
In [145]: open('unzepripzeiorpozier')
```

```
---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call last)

<ipython-input-145-8c476c9cacac> in <module>()
----> 1 open('unzepripzeiorpozier')


FileNotFoundError: [Errno 2] No such file or directory: 'unzepripzeiorpozie
```

16 5. Créer des objets

- Attributs
- Méthodes

In [211]:
```
from math import atan2

class Complex(object):
    def __init__(self, partie_reelle, partie_imaginaire):
        self.real = partie_reelle
        self.ima = partie_imaginaire

    def display(self):
        print('{}+{}i'.format(self.real, self.ima))

    def __str__(self):
        if self.ima>0:
            return '{}+{}i'.format(self.real, self.ima)
        else:
            return '{}-{}i'.format(self.real, -self.ima)

    def __repr__(self):
        return "Complexe({}, {})".format(self.real, self.ima)

    def __add__(self, other):
        if isinstance(other, Complex):
            return Complex(self.real+other.real,
                           self.ima+other.ima)
        else:
            return Complex(self.real+other,
                           self.ima)

    def __radd__(self, other):
        return self + other

    def conj(self):
        return Complex(self.real, -self.ima)
    # __mul__, __truediv__, __sub__, __pow__, __neg__

    @property
    def theta(self):
        return atan2(self.ima, self.real)

class ImaginairePur(Complex):
    def __init__(self, val):
        self.real = 0
        self.ima = val
```

```
            def __str__(self):
                return '{}i'.format(self.ima)


        z = Complex(3.14, 2)
        #z.real = 3.14
        #z.imag = 2
        print(z.real)
        z.display()
        print(z)
        z

        print(z + z)
        print(z + 2)
        print(2 + z)
        print(z.conj())

        i = ImaginairePur(1)
        print(i)
        print(i+2)

        i.theta
```

```
3.14
3.14+2i
3.14+2i
6.28+4i
5.140000000000001+2i
5.140000000000001+2i
3.14-2i
1i
2+1i
```

```
Out[211]: 1.5707963267948966
```

```
In [183]: 1/2
```

```
Out[183]: 0.5
```

16.1 Les décorateurs

- vectorize

```
In [196]: #@nom_du_decorateur
          #def ma_fonction():
          #    pass

          #def ma_fonction():
```

```
            #    pass
            #ma_fonction = nom_du_decorateur(ma_fonction)

            def vectorize(fonction_initiale):
                def la_nouvelle_fonction(x):
                    if isinstance(x, list):
                        return [fonction_initiale(elm) for elm in x]
                    return fonction_initiale(x)

                return la_nouvelle_fonction

            @vectorize
            def ma_fonction(x):
                if x>0:
                    return x
                else:
                    return x**2

            ma_fonction([1,2, -5])
```

```
Out[196]: [1, 2, 25]
```

```
In [205]: def ma_fonction(a, b, c=2):
                return a+b+c

            def ma_fonction(a, b, *args):
                print(a, b, args)

            ma_fonction(1, 2, 3, 4, 5)

            def ma_fonction(a, b, *args, **kwd):
                print(a, b, args, kwd)

            ma_fonction(1, 2, 3, 4, 5, alpha=1, beta=2)

            def ma_somme(*args):
                out = 0
                for elm in args:
                    out += elm
                return out

            print(ma_somme(1, 2, 4, 5))

            def truc(a, b):
                return a*b

            param = (1, 2)
            truc(*param)
```

```
param = {'b':1, 'a':2}
truc(**param)
```

```
def puissance(U, I, **kwd):
    print(kwd)
    return U*I

parametres = {'U':1, 'I':3, 'T':26, 'P':1.3}

puissance(**parametres)
```

```
1 2 (3, 4, 5)
1 2 (3, 4, 5) {'beta': 2, 'alpha': 1}
12
{'T': 26, 'P': 1.3}
```

```
Out[205]: 3
```

```
In [207]: DEBUG = False
          def affiche_arguments(f):
              def nouvelle_fonction(*args, **kwd):
                  if DEBUG: print(args, kwd)
                  return f(*args, **kwd)
              return nouvelle_fonction

          @affiche_arguments
          def ma_fonction(a, b, c):
              return (a+b+c)/2

          ma_fonction(1, 3, 4)
```

```
Out[207]: 4.0
```

16.2 Exemples d'objets

- Partition de musique. Pour simplifier, suite de son définie par hauteur, durée, intensité.
- Arbre : généalogique, livre, équation symbolique. Exemple de méthode : parentée, ascendance

```
In [ ]:
```