TP 2
List and algorithms

Objectifs du TP : Functions, List manipulation and Sort algorithm.

1 Functions

```
def random_function(arg1, arg2):
        # Do what ever you want
        return out1, out2, out3
```

Exercise 1 : First functions
Without using the module `math`, the function `abs`, the function `pow` or the operator `**`

- Define a function `absolute(n)` which take an integer `n` as input and return the absolute value of n
- Define a function `fact(n)` which take an integer as input and return its factorial value.
- Define a function `power(x, n)` which take a float `x` and an integer positive `n` and return x^n.

Exercise 2 : On geometry

- Create two functions `VolumeSphere(r)` and `AreaSphere(r)` which return the volume and the area of a sphere of radius `r`.
- Create a function `IsRectangle(A, B, C)` and `IsIsocele(A, B, C)` which take three tupples as input (`A =(x, y)`) and return True is the triangle ABC is a triangle rectangle respectively isocele.
- Create a fonction `SquareCircumscribedCircle(r)` which return the area an d the perimeter of the biggest square circumscribed by a circle of the radius `r` (use a drawing if needed).
- Create a fonction `AreAligned(A, B, C)` which return True, if the points A, B and C are aligned and else False. A, B and C are tuple.

Exercise 3 : On numeral system

- Create a fonction `decimal2base(N, b)`, which takes a input an integer `N` and a integer `b` and which return the expression of N in the base b. For instance, if $N = 100$ and $b = 6$, the result will be `[2, 4, 4]` or if $N = 100$ and $b = 2$ the result will be `[1, 1, 0, 0, 1, 0, 0]`.
- On the opposite, create a fonction `base2decimal(L, b)` which take as input a list representing an integer in the base b and return its value in decimal base.

Help :

2 List manipulation

Exercise 4 : Prime number

- Create a function `isprime(p)` which take an integer and return True, if p is prime number. For that we will check if `p` is divisible by all integer between `2` and `p -1`. Tester on 1033 (prime), 4353 (non prime).
- What is the complexity of the previous question ? How to reduce the complexity ?

- Implement a function `get_prime_list(N)` which return the list of the prime number inferior to N. This function should use the previous function `isprime`.
- What is the complexity of the function `get_prime_list` ?
- To reduce the complexity, implement the sieve of Eratosthenes in a function `eratosthenes_sieve(N)` :
  - Create a list `L` of boolean of size `N`, initially all equal to True. Put `L[0]` and `L[1]` equal to False
  - Then for each position `L` equal to True, put all the multiples of `i` to False.
  - At the end, all the position equal to True are prime number
  - Return the list of prime number inferior to N
- Check the number of prime number inferior to 1^3. You should find 168.
- Using the module `time` and the function `time.time`, calculate the calculation time for both functions `get_prime_list` and `eratosthenes_sieves`.

Exercise 5 : On finding an element

- Create a function `RandomIntList(N, a, b)` which return a random list of integers between a and b. Use the module random.
- Create a fonction `IsInList(element, L)` which return True if the element is in the list else True
- Create a fonction `GetFirstIndexList(element, L)` which return the index of the first position of the element in L. If the element is not in L, the function returns -1
- Create a fonction `GetAllIndexList(element, L)` which return a list with all the index of the positions of the element in the list L. If the element is not in the list, the function returns an empty list.
- Create a fonction `CountElementList(element, L)` which count the number of times that the element is in the list L. Different solutions are possible.

Exercise 6 : On statistics

- Create a function `RandomFloatList(N)` which return a random list of floats between a and b. Use the module random.
- Create a function `FindMin(L)` and `FindMax(L)` to return the minimum and maximal value of L.
- Create a function `FindAmplitude(L)` which return the amplitude of the list (the difference between minimal and maximal value). Do a first method using `FindMin` and `FindMax` and a second method with only one function.
- Create a function `GetAverage(L)` which return the average value of the list `L`.
- Create a function `GetStd(L)` which return the standard deviation of the list `L`.

Help : %, //, for, if, return

3 Sort

Exercise 7 : Selection sort
The selection sort consist in finding the minimum value in the list `L`, put it in position 0 and then do it again to the list `L[1:]` and so on...

- Implement a function `FindMin(L)` which return the minimum value and the index of this value in the list of integers `L`
- Implement a function `exchange(L, i, j)` which exchange the value in position `i` and `j` in the list `L`.
- Complete the following code to implement the `selection_sort` function using the two previous functions `FindMin(L)` and `exchange(L, i, j)`.

```
def selection_sort(L):

    n = len(L)
    for i in range(n-1):
        print(L)
        ...

        print('Minimal value position : {}'.format(index))
        print('Exchange between {} and {}'.format(index, i))
        print(L)
```

Exercise 8 : Bubble sort
The bubble sort will swap consecutive element `i` and `i + 1` if they are not in the good order. It will stop when all the element are ordered.

- Using the following code, implement the bubble sort.
- How are we sure that the while loop will terminate ? Provide an other implementation without the while loop ? Is it more efficient ?

```
def bubble_sort(L):

    n = len(L)
    list_sorted = False

    while not list_sorted:
        list_sorted = True
        for i in range(n -1):
                ...

        n -= 1
```

Exercise 9 : Insertion sort
The insertion sort consist to insert one element at the good position in a already ordered list. If we consider the list `L[:i]` already sorted, we will take the element `L[i+1]` and find its good position.

- Using the following code, implement the insertion sort.
- Why do we start the sorting procedure at position 1 ?
- How are we sure that the while loop will terminate ? Provide an implementation with a for loop.

```
def insertion_sort(L):

    n = len(L)
    for i in range(1, n):
        j = i
        x = L[i]

        while j>0 and L[j-1]>x:
                        ...

        ...
```

4 Source

- http://perso.numericable.fr/jules.svartz/prepa/IPT_sup/archives_TP_sup/TP3_MPSI.pdf
- http://s15847115.domainepardefaut.fr/moodle/pluginfile.php/1352/mod_resource/content/1/TP_numpy.pdf
- http://www.armelmartin.mon-site-a-moi.fr/doc/info/tp2_numpy_matplotlib.pdf