Séance 01 :

Objets mutables et non mutables

1. Cours

C.F. cours

2. Exemple 1

On donne le code suivant.

```
1  A = [k**2 for k in range(5)]
2  B = [k**2 for k in range(10, 15)]
3  C = A.extend(B) # Concatene B a A
4  del A, B
```

1. ***Que contient la liste*** `C` ***et pourquoi ?***
   La liste `C` est vide, ce n'est même pas une liste, car la méthode `extend` pour une `list` ne retourne rien.
2. ***Comment modifier ce code pour qu'il remplisse son rôle ?***
   Il faut modifier la ligne 4. par `C = A + B`.

3. Exemple 2

On donne les commandes suivantes :

```
1  a = ["t", "o", "t", "o"]
2  a.remove("o")
3  c = a.remove("o")
4  b = "toto"
5  b.strip("o")    # supprime "o" s'il se trouve en debut ou fin de chaine
6  d = b.strip("o")
```

1. ***Deviner le contenu des variables*** `a` , `b` , `c` ***et*** `d` ***sans exécuter le code.***

   `a = "tt"`, `b = "tot"`, `c = null` et `d = null`.

4. Exercice 1 : Différence entre deux listes

1. ***Ecrire une fonction*** `diff(a, b)` ***qui calcule la différence entre deux listes*** `a` ***et*** `b`***.***

```
1  def diff(a, b):
2      result = a.copy()
3      for el in b:
4          if (el in result):
5              result.remove(el)
6      return (a, b, result)
7
8
9  # ----- Tests  -----
10 if __name__ == "__main__":
11     a = [2, 7, 8, 15, 2, 3, 4, 5, 2, 3]
12     b = [2, 13, 8, 3, 2, 7, 8, 3]
13     d = diff(a, b)
14     print(
15         'liste a :\n\t{}\nliste b :\n\t{}\ndiff(a, b) :\n\t{}'
16         .format(a, b, d[2])
17     )
```

5. Exercice 2 : Nombres premiers

1. ***En utilisant*** `remove` ***donner la liste des 168 nombres premiers de 1 à 1 000 en utilisant le crible d'Eratosthène. Faire moins de 1 500 tours de boucle. De même trouver les 1 229 nombres premiers de 1 à 10 000 en faisant moins de 20 000 tours de boucle.***

```
def prime(n):
    """Returns the list of the n first n prime numbers,
    and and the number of loop turns"""

    count = 0  # loop counter

    nb = [k for k in range(2, n+1)]
    for i in range(2, n+1):
        for el in nb:
            if el % i == 0 and el != i:
                count += 1  # On compte combien de tours de boucle on fait
                nb.remove(el)
    return (count, nb)
```

Résultats :

```
========= Test pour n = 1000 =========
Le programme fait 831 tours (max : 1500 tours).
Temps d execution : 0.008s
Nombre de resultats : 168
Resultats :[2, 3, 5, 7, '...', 991, 997]

========= Test pour n = 10000 =========
Le programme fait 8770 tours (max : 20000 tours).
Temps d execution : 0.595s
Nombre de resultats : 1229
Resultats :[2, 3, 5, 7, '...', 9967, 9973]
```

6. Exercice 3 : Suite de Syracuse

(a) ***Déclarer la fonction*** `S` ***en utilisant reste et quotient dans les nombres entiers.***

```
def S(n):
    if n % 2 == 0:
        return n/2
    elif n != 1:
        return 3*n+1
    else:
        return 1
```

(b) ***Afficher la liste des valeurs*** $u_p = S^p(c)$ ***pour*** p ***allant de*** 0 ***à*** $+\infty$ ***et en s'arrêtant dès que*** $u_p = 1$***.***

```
def qb(x):
    up = x
    while up == 1:
        print(up)
        up = S(up)
```

(c) ***Tester pour*** x ***dans la liste*** `[2, 7, 19, 23, 29]`.

```
def qc():
    X = [2, 7, 19, 23, 29]
    for x in X:
        qb(x)
```

(d) ***Trouver les entiers*** x ***compris entre*** 3 ***et*** 300 ***pour lequel le vol (c'est-à-dire la longueur de la suite pour obtenir 1) est le plus long.***

```python
def qb(x):
    up = x
    results = [up]
    while up > 1:
        up = S(up)
        results.append(up)
    return results


def qd():
    X = [k for k in range(3, 301)]
    volMax = 0
    result = []
    for x in X:
        a = qb(x)
        vol = len(a)
        if vol > volMax:
            result = []
            volMax = vol
            result.append(x)
        elif vol == volMax:
            result.append(x)
    return result, volMax
```

On trouve que le vol le plus long est $v = 128$, et est atteint pour $x \in \{231, 235\}$.

(e) ***Calculer l'intersection entre deux suites de syracuse en renvoyant la longueur de l'intersection et les derniers éléments où les deux suites sont différentes. En cas d'inclusion, on renvoie le complexe*** $1j$ ***pour indiquer qu'une suite est incluse dans l'autre.***

```python
def contains(small, big):
    for i in range(len(big)-len(small)+1):
        for j in range(len(small)):
            if big[i+j] != small[j]:
                break
        else:
            return i, i+len(small)
    return False


def interSyracuse(i1, i2):
    i1, i2 = qb(i1), qb(i2)
    grande, petite = [], []
    # On selectionne la plus petite liste
    if max(len(i1), len(i2)) == len(i1):
        grande, petite = i1, i2
    else:
        grande, petite = i2, i1
    for k in range(len(petite), 1, -1):
        subPetite = petite[-k:]
        if contains(subPetite, grande):
            if subPetite == petite:
                return (1j)
            else:
                maxi = subPetite[0]
                o = petite.index(maxi)
                p = grande.index(maxi)
                return(len(subPetite), petite[:o][-1], grande[:p][-1])
```

On trouve :

$$interSyracuse(7, 320) = (9, 80, 13)$$
$$interSyracuse(7, 230) = 1j$$