Cours 1 : Objets (im)mutables

1. Cours

C.F. cours

2. Exemple 1

```
A = [k**2 for k in range(5)]
B = [k**2 for k in range(10, 15)]
C = A.extend(B)  # Concat B à A
del A, B
```

- ***Que contient la liste C et pouquoi ?***
  C est une liste vide car la fonctino [List].extend ne renvoie rien.
- ***Comment modifier ce code pour qu'il remplisse son rôle ?***

```
A = [k**2 for k in range(5)]
B = [k**2 for k in range(10, 15)]
C = A +B
del A, B
```

3. Exemple 2

```
a = ['t', 'o', 't', 'o']
a.remove('o')
c = a.remove('o')
b = 'toto'
b.strip('o')
d = b.strip('o')
```

- ***Deviner le contenu des variables a, b, c et d sans exécuter le code.***
  A la fin, on a :

variable	content
a	'tt'
b	'tot'
c	∅
d	∅

3. Exercice n° 1 : *Différence entre deux listes*

```
def diff(a, b):
    result = a.copy()
    for el in b:
        if (el in result):
            result.remove(el)
    return (a, b, result)
```

4. Exercice n° 2 : *Nombres premiers*

```
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)
```

5. Exercice n° 3 : *Suite de Syracuse*

a) Déclarer la fonction S en utilisant les 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) Liste des termes de la suite de Syracuse

```
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 lmes entiers x compris en 3 et 300 pour lequel le vol est le plus long

```
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
if __name__ == '__main__':
    a = qd()
    print(
        'Liste des x pour lesquels le vol est le plus long ({}) :\n{}'
        .format(a[1], a[0])
    )
```

Résultats :

```
Liste des x pour lesquels le vol est le plus long (128) :
[231, 235]
```

e) Intersection de deux suites de Syracuse

```
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 sélectionne 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])
```

Résultats :

```
print(interSyracuse(7, 320))
(9, 80, 13)

print(interSyracuse(7, 230))
1j
```