Arithmétique et listes en compréhension - Corrigé

NUMWORKS

Cette fiche a été rédigée par **Alain Busser**. Il enseigne au lycée Roland-Garros du Tampon. Il est aussi animateur à l'IREM de La Réunion et créateur du langage de programmation Sophus.

1 Exercices préliminaires

Décrire en compréhension la liste des entiers pairs de 2 chiffres.

```
[entier for entier in range(100) if entier%2==0]
```

Décrire en compréhension la liste des nombres inférieurs à 100 qui sont multiples, ou bien de 5, ou bien de 7.

```
[n for n in range(100) if n%5==0 or n%7==0]
```

Décrire en compréhension la liste **fizzbuzz** des entiers inférieurs à 100 qui sont multiples de 5 ou de 7 mais pas de 5 et 7 en même temps.

```
fizzbuzz=[n for n in range(100) if (n%5==0 or n%7==0) and n%35!=0]
```

Si le temps le permet, programmer le crible d'Eratosthène avec des compréhensions

```
crible = list(range(2,100))
for n in range(2,10):
  crible=[k for k in crible if k<=n or k%n!=0]
print(crible)
```

2 Exercice

Rédiger un test de primalité sous forme d'une fonction **estPremier(n)** acceptant en entrée un entier n, et renvoyant **True** si n est premier, et **False** sinon.

```
def estPremier(n):
  return len(divisors(n))==2
```

3 Autres exercices

Écrire une fonction Python **cd(a,b)** qui accepte deux entiers a et b en entrée et qui renvoie la liste des diviseurs communs à a et b. On peut utiliser la fonction **inter()** définie au début de cette activité.

```
def cd(a,b):
  return inter(divisors(a),divisors(b))
```

En déduire un algorithme «naïf» de calcul du pgcd, qui est le plus grand élément de `cd(a,b)`, comme son nom «plus grand commun diviseur» l'indique. On pourra utiliser la fonction `max(L)` qui accepte en entrée une liste L de nombres et renvoie le plus grand élément de cette liste. On demande une fonction `pgcd(a,b)` qui accepte en entrée deux entiers a et b, et renvoie leur pgcd.

```
def pgcd(a,b):
  return max(cd(a,b))
```

En déduire également un test `premiersEntreEux(a,b)` qui accepte en entrée deux entiers a et b et renvoie `True` si a et b sont premiers entre eux et `False` s'ils ont un diviseur commun autre que 1. Là encore, la fonction `len` peut être utile, puisque le fait que deux entiers soient premiers entre eux est lié au nombre de leurs diviseurs communs.

```
def premiersEntreEux(a,b):
  return len(cd(a,b))==1
```

```
def premiersEntreEux(a,b):
  return pgcd(a,b)==1
```

Déduire de la fonction précédente, une fonction `eulerphi(n)` où n est un entier, renvoyant le nombre d'entiers inférieurs à n et premiers avec n. Par exemple, les nombres inférieurs à 8 et premiers avec 8 sont 1, 3, 5 et 7 ce qui fait que `eulerphi(8)==4`.

```
def eulerphi(n):
  return len([x for x in range(1,n) if premiersEntreEux(x,n)])
```

Écrire un test de perfection sous la forme d'une fonction Python `estParfait(n)` acceptant un entier n en entrée et renvoyant `True` si n est parfait et `False` sinon.

```
def estParfait(n):
  return sum(divisors(n))==2*n
```