TP4

Partie 1 — Théorie des nombres

Ex. 1 — Primalité d'un entier

Rappels: - Un entier `n` est premier si il n'est divisible par aucun entier `i` tel que `1<i<n`. - Pour savoir si un nombre est premier, il suffit de tester la divisibilité de `n` par tous les entiers entre `2` et `n/2` (arrondi à l'inférieur). - Si aucun diviseur de `n` n'est trouvé parmi les entiers candidats, cela signifie que `n` est premier. - Dès lors qu'un diviseur `n` est trouvé parmi les candidats, alors le nombre n'est pas premier.

Par exemple: - `11` n'est divisible par aucun des nombres `2`, `3`, `4`, et `5`, donc il est premier. - `18` est divisible par `3`, donc il n'est pas premier.

Quelques outils utiles: - `a` est divisible par `b` lorsque le reste de la division entière de `a` par `b` vaut `0`. En python, utiliser l'opérateur `%`. - Pour effectuer une division entière, autrement dit la division arrondie à l'entier inférieur, il faut utiliser l'opérateur `//`. - Pour énumérer l'ensemble des candidats possibles de diviseurs, vous utiliserez la fonction `range(x,y)` qui énumère tous les nombres de `[x,y[`. Attention au fait que `y` n'est pas énuméré. Si vous en avez besoin, appelez range(x,y+1).

Question 1 : Écrire une fonction `is_prime` ayant un paramètre `n`, qui renvoie `True` lorsque `n` est premier et `False` sinon.

Question 2 : Écrire une fonction `print_prime` ayant un paramètre `n`, faisant appel à la fonction `is_prime` et qui affiche soit: - une chaîne de la forme "Le nombre 5 est premier." pour les nombres premiers. - une chaîne de la forme "Le nombre 6 n'est pas premier." sinon.

Question 3 : Appeler la fonction `print_prime` avec comme argument la valeur `4`.

Question 4 : Appeler la fonction `print_prime` avec comme argument la valeur `7`.

Question 5 : Appeler la fonction `print_prime` avec comme argument la valeur `15`.

Question 6 : Appeler la fonction `print_prime` avec comme argument la valeur `101`.

Question 7 : Écrire une fonction `print_all_prime` ayant un paramètre `n`, qui affiche la suite de tous les entiers premiers entre `2` et `n`. Par exemple, `print_all_prime(11)` affichera: `2	3	5	7	11`

Question 8 : Appeler `print_all_prime` avec comme argument la valeur `101`.

Ex. 2 : Décomposition en facteurs premiers

Le Problème: - La décomposition en facteurs premiers d'un entier `n` est la liste des nombres premiers dont le produit vaut `n` (par exemple: `20=2*2*5`) - On veut ici écrire une fonction 'decompose' affichant les différents facteurs le composant, dans l'ordre croissant. - Pour cela, on va avoir besoin d'une variable `i`, qui au départ vaudra 2. on va tester si `i` divise `n`. Tant que l'on n'a pas trouvé tous les diviseurs (`n` n'est pas égal à `1`) - Si `i` divise `n`, cela signifie que `i` est un diviseur de `n`. Il faut alors: - Afficher `i` - Diviser `n` par `i` - Ne pas modifier `i` (sinon on va oublier de vérifier si `i` n'apparaitrait pas plusieurs fois dans la décomposition) - Si `i` ne divise pas `n`, on ajoute `1` à la valeur de `i`

Question 1 : Écrire une fonction `decompose` ayant un paramètre `n`, qui affiche une première ligne du type "L'entier 12 se décompose en:", suivi de la liste des facteurs premiers de `n`, un par ligne et du plus petit au plus grand.
Ainsi, `decompose(12)` doit afficher sur quatre lignes:

```
L'entier 12 se décompose en:
```

```
2
2
3
```

Question 2 : Appeler la fonction `decompose` avec comme argument la valeur `6`.

Question 3 : Appeler la fonction `decompose` avec comme argument la valeur `19`.

Question 4 : Appeler la fonction `decompose` avec comme argument la valeur `90`. ### Ex. 3 Calcul du PGCD par l'Algorithme d'Euclide

- Le pgcd de deux entiers `a` et `b` est le plus grand entier `n` qui divise `a` et `b`.
- L'algorithme d'Euclide permet de le calculer, selon le principe suivant:
  - Si `a` est divisible par `b`, le pgcd de `a` et `b` vaut `b`.
  - Sinon, soit `r` le reste de la division entière de `a` par `b`
  - Le pgcd de `a` et `b` est égal au pgcd de `b` et `r`

Par exemple, le calcul du PGCD de `90` et de `35` passe par les étapes suivantes:

```
a  = b  x q + r
90 = 35 x 2 + 20
35 = 20 x 1 + 15
20 = 15 x 1 + 5
15 = 5  x 3 + 0
```

Le PGCD de `90` et `35` est donc `5`.

Question 1 : Écrire une fonction `euclide` ayant deux paramètres `a` et `b`, qui affiche les différentes étapes du calcul du pgcd de `a` et de `b`.

L'affichage sera composé d'une première ligne donnant les arguments de la fonction, de la su

Par exemple, `euclide(90,35)` affichera:

```
Calcul du PGCD de 90 et de 35 :
90 = 35 x 2 + 20
35 = 20 x 1 + 15
20 = 15 x 1 + 5
15 = 5 x 3 + 0
Le PGCD est donc 5
```

Question 2 : Appeler la fonction `euclide` avec comme arguments `18` et `12`.

Question 3 : Appeler la fonction `euclide` avec comme arguments `7` et `5`.

Partie 2 — Arithmétique

Ex. 4 — Exponentiation Rapide

Le but de cet exercie est d'écrire une fonction réalisant l'opérateur d'exponentiation (x**n) en n'utilisant que les opérations + et *

- L'exponentiation d'un entier `x` par un entier `n` est l'entier xn=`x*x*...*x` (`n` fois).
- L'algorithme d'exponentiation rapide permet de calculer xn efficacement, selon le principe suivant:
  - Si `n` est égal à `0`, xn = 1
  - Si `n` est égal à `1`, xn = x
  - Si `n` est pair, alors `n` peut s'écrire `2*m` (division entière), et xn = xm * xm
  - Si `n` est impair, alors `n` peut s'écrire `2*m+1`, et xn = xm * xm * x

On utilisera une variable `resultat` initialisée à `1`. Si `n` est pair, on peut faire x=x2 et n=n/2 sans changer le résultat. Si `n` est impair, on peut multiplier `resultat` par `x`, puis faire x=x2 et n=(n-1)/2.

Question 1 : Écrire une fonction `exponentiation_rapide` ayant deux paramètres `x` et `n`, qui retourne l'entier xn, calculé par exponentiation rapide.

Question 2 : Appeler et afficher le résultat de la fonction `exponentitiation_rapide` avec comme arguments `3` et `4`.

Question 3 : Appeler et afficher le résultat de la fonction `exponentitiation_rapide` avec comme arguments `2` et `10`.

Ex. 5 — Suite de Syracuse

- La suite de Syracuse depuis `x` est la suite (un) telle que:
  - u0 = x
  - Si `n` est pair, alors un+1 = un/2
  - Si `n` est impair, alors un+1 = 3*un+1
- On observe que quel que soit `x`, après un certain nombres d'étapes `n`, un=`1`, Après ce `n`, la suite alterne entre trois valeurs: `4`, `2` et `1`.
- L'objectif ici est d'afficher les termes de la suite depuis `x`, jusqu'à la première apparition de la valeur `1`.

Question 1 : Écrire une fonction `syracuse` ayant un paramètre `x`, qui affiche les entier produits par la suite de syracuse initialisée en `x`, jusqu'à atteindre `1`. Ainsi, `syracuse(10)` doi afficher sur sept lignes:

```
10
5
16
8
4
2
1
```

Question 2 : Appeler la fonction `syracuse` avec comme argument la valeur `4`.

Question 3 : Appeler la fonction `syracuse` avec comme argument la valeur `13`.

Question 4 : Appeler la fonction `syracuse` avec comme argument la valeur `25`.