Solution ❁ Arithmétique et Python

Partie A. Nombres premiers jumeaux

1. ❧ Dans le code de `P0`, la boucle `for` est inutile. En effet,
- ou bien $N \leqslant 2$ et on sort de la fonction en retournant une valeur booléenne avant d'entrer dans la boucle `for` ;
- ou bien on entre dans la boucle `for` avec $d = 2$ et si l'entier $N > 2$ est pair, on retourne `False` ;
- alors que si l'entier $N > 2$ est impair, le test échoue et la fonction retourne la valeur `True`.

Bref, la fonction `P0` retourne `True` si, et seulement si, l'entier N est égal à 2 ou impair et supérieur à 3.

REMARQUE.— Quelle que soit sa position dans le code, l'instruction `return` indique que l'exécution de la fonction est terminée.

❧ Si $N > 2$, la fonction `P1` retourne `False` dès qu'elle trouve un entier $2 \leqslant d < N$ qui divise N et retourne `True` si on parvient au terme de la boucle `for` sans avoir trouvé de diviseur de N.

La fonction `P1` retourne donc `True` si, et seulement si, son argument N est un nombre premier.

REMARQUE.— On aura remarqué que la seule différence entre ces deux fonctions est l'**indentation** de la dernière instruction `return` !

2. La variable k contient successivement les entiers naturels 0, 1, 2, ... et la variable n contient donc les entiers naturels tels que

$$\exists\, k \in \mathbb{N}, \quad n = k^2 + 1 \qquad \text{et} \qquad n \leqslant N.$$

Avant d'entrer dans la boucle `while`, on définit une liste vide `L`. Chaque fois qu'un entier n est premier, il est ajouté à la liste `L`.

La valeur retournée par `P2` est donc la liste des nombres premiers n tels que

$$\exists\, k \in \mathbb{N}, \quad n = k^2 + 1 \qquad \text{et} \qquad n \leqslant N.$$

❧ En particulier, pour $N = 127$,

k	n	Premier
0	1	non
1	2	oui
2	5	oui
3	10	non
4	17	oui
5	26	non
6	37	oui
7	50	non
8	65	non
9	82	non
10	101	oui
11	122	non

3. On va tester successivement les entiers $p > N$ jusqu'à ce qu'on trouve un entier p tel que `P1(p)` retourne `True`.

La valeur initiale du booléen `continuer` nous assure d'entrer dans la boucle `while`.

La valeur initiale de l'entier `p` nous dit que le premier entier testé sera $N + 1$.

On continue tant que `P1(p)` prend la valeur `False` (c'est-à-dire : p n'est pas premier) et on s'arrête donc au plus petit nombre premier $p > N$.

```
def premierSuivant(N):
    p = N
    continuer = True
    while continuer:
        p += 1
        continuer = not(P1(p))
    return p
```

REMARQUE.— On retiendra qu'un choix parlant d'identifiant pour le booléen aide à bien comprendre le fonctionnement du code.

REMARQUE.— Ce code n'est pas efficace (il teste tous les entiers alors qu'il suffirait de tester les entiers impairs), mais comme la fonction `P1` de l'énoncé n'est pas efficace du tout, pourquoi se fatiguer ?

VARIANTE RÉCURSIVE ©Valentin Plantard-Wahl.

```
def premierSuivant(N):
    if P1(N+1):
        return N+1
    else:
        return premierSuivant(N+1)
```

De deux choses l'une : ou bien $N+1$ est premier, ou bien le plus petit nombre premier supérieur à N est aussi le plus petit nombre premier supérieur à $(N + 1)$. Simple, clair, efficace !

4. a. Notons $(p_n)_{n\in\mathbb{N}}$, la suite des nombres premiers strictement supérieurs à N. Ces nombres seront calculés par des appels itérés à la fonction `premierSuivant`.

Nous allons passer en revue les couples (p, q) de la forme

$$(p, q) = (p_n, p_{n+1})$$

(deux nombres premiers consécutifs strictement supérieurs à N) jusqu'à ce qu'on trouve un couple tel que $q - p = 2$.

```
def jumeauxSuivants(N):
    q = premierSuivant(N)
    pas_trouvé = True
    while pas_trouvé:
        p = q
        q = premierSuivant(p)
        pas_trouvé = ((q-p)>2)
    return (p,q)
```

REMARQUE.— ©VP-W

```
def jumeauxSuivants(N):
    p = premierSuivant(N)
    q = p+2
    if P1(q):
        return (p,q)
    else:
        return jumeauxSuivants(p)
```

Si $q = p+2$ est premier, c'est gagné ! Sinon on recommence en partant cette fois de $p > N$.

4. b. Le premier couple d'entiers jumeaux est $(3,5)$.

Si (p,q) est un couple d'entiers jumeaux, le couple suivant (p',q') est obtenu avec `premierSuivant(p)`. Attention, si on a forcément $p' > p$, on n'a pas forcément $p' > q'$ puisque le couple $(p,q) = (3,5)$ est suivi par le couple $(p',q') = (5,7)$.

On continue à calculer le couple suivant tant que $q \leqslant N$.

```
def jumeaux(N):
    L = []
    (p,q) = (3,5)
    while (q<=N):
        L.append((p,q))
        (p,q) = jumeauxSuivants(p)
    return L
```

REMARQUE.— On peut certainement faire beaucoup plus efficace, mais en pareille situation, il vaut mieux un code clair, correct et qui utilise les fonctions précédentes qu'un code efficace (et dont l'efficacité risque d'être la *seule* qualité).

Partie B. Une fonction récursive

5. Comme $101 > 100$, le test réussit et l'appel de `M(101)` retourne $101 - 10 = 91$.

Plus généralement, si $N > 100$, alors l'appel de `M(N)` retourne $N - 10$.

6. L'appel `M(100)` retourne le résultat du double appel $M(M(111))$. Comme $111 > 100$, le premier appel `M(111)` retourne 101 (d'après la question précédente) et comme $101 > 100$, le second appel `M(101)` retourne 91.

De même, l'appel `M(99)` retourne le résultat du double appel $M(M(110))$. Comme $110 > 100$, le premier appel `M(110)` retourne 100 et le second appel `M(100)` retourne 91 (étude précédente).

De même, l'appel `M(98)` retourne le résultat du double appel $M(M(109))$. Le premier appel `M(109)` retourne 99 et d'après ce qui précède, le second appel `M(99)` retourne 91.

7. Il est plus que probable que `M(N)` retourne 91 pour tout entier $N \leqslant 100$.

Cette propriété a déjà été établie pour $M = 100$, puis pour $M = 99$ et pour $M = 98$.

Supposons qu'elle soit vraie pour un entier $91 \leqslant N \leqslant 100$.

L'appel `M(N-1)` retourne alors le résultat du double appel $M(M(N+10))$. Comme $N \geqslant 91$, on a

$$N + 10 \geqslant 101 > 100,$$

donc $M(N+10) = N$ et

$$M(N-1) = M(M(N+10)) = M(N) = 91.$$

On a ainsi démontré par une récurrence finie que

$$\forall\, 90 \leqslant N \leqslant 100, \quad M(N) = 91.$$

Supposons qu'il existe un entier $N_0 \leqslant 90$ tel que

$$\forall\, N_0 \leqslant N \leqslant 100, \quad M(N) = 91.$$

(Cette hypothèse a été établie pour $N_0 = 90$.)

L'appel $M(N_0 - 1)$ retourne alors $M(M(N_0 + 10))$ et comme $N_0 \leqslant 90$, alors $N_0 \leqslant N_0 + 10 \leqslant 100$. D'après l'hypothèse de récurrence,

$$M(N_0 + 10) = 91$$

et donc

$$M(N_0 - 1) = M(91) = 91.$$

On a ainsi démontré par récurrence descendante que

$$\forall\, N_0 \leqslant 90,\ \forall\, N_0 \leqslant N \leqslant 100, \quad M(N) = 91$$

et donc plus simplement que

$$\forall\, N \leqslant 100, \quad M(N) = 91.$$

REMARQUE.— C'est une récurrence diabolique !