TP 1 – Présentation de Python

Introduction

Un algorithme est une suite d'instructions élémentaires qui s'appliquent dans un ordre déterminé à des données et fournissant en un nombre fini d'étapes des résultats.
Une recette de cuisine est un algorithme. En effet il s'agit de :

- réunir les ingrédients ;
- préparer le plat ;
- arriver au plat prêt à être servi.

On peut considérer un algorithme comme une « boîte noire » fonctionnant selon le schéma suivant :

Entrée (données) → **Traitement** (instructions) → **Sortie** (résultats

Voici un exemple d'algorithme : « choisir un entier, le multiplier par 3, ajouter 1, donner le résultat ».
Si l'entier choisi est 4, l'algorithme donne . . .

Les algorithmes devront être écrits en **pseudo-code** ; c'est une écriture « en français » standardisée, indépendamment d'un langage spécifique comme Python, C, Java…

L'algorithme ci-dessus s'écrira ainsi.

Saisir a
$b \leftarrow 3a + 1$
Afficher b

Algorithme 1

La flèche $\leftarrow$ désigne l'affectation des variables. L'algorithme ci-dessus fonctionne avec deux variables, a et b. Pour suivre l'état des variables à chaque étape, on construit ce tableau :

	a	b
Saisir a	4	
$b \leftarrow 3a + 1$	4	13
Afficher b	4	**13**

L'algorithme affiche b, c'est-à-dire 13.

1. Que retourne l'algorithme suivant s'il est exécuté avec 5 comme valeur de a ?

Saisir a
$b \leftarrow 2a - 1$
$a \leftarrow 2b - 1$
$b \leftarrow 2a - 1$
Afficher b

Algorithme 2

Compléter le tableau ci-dessous.

	a	b
Saisir a	5	

2. Même question avec l'algorithme suivant.
Construire le tableau pour suivre l'état des variables.

Saisir a
$a \leftarrow a - 1$
$b \leftarrow a - 1$
$c \leftarrow b - 1$
$a \leftarrow a^2 + c$
$b \leftarrow a/b$
Afficher b

Algorithme 3

Les ordinateurs ne sont fondamentalement capables de comprendre que quatre catégories d'instructions :

- la lecture / écriture ;
- l'affectation de variables ;
- les tests ;
- les boucles.

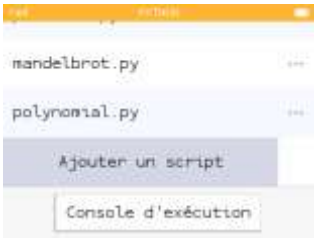
Nous venons d'utiliser les deux premières catégories dans les algorithmes 1, 2 et 3.

Présentation de Python

Python est un langage qui va permettre de programmer nos algorithmes.
Nous utiliserons la calculatrice NumWorks, ou son émulateur sur téléphone. Vous utiliserez en SNT une version sur ordinateur de Python.

Il existe deux façons de travailler.

- Utiliser la console où vous pouvez saisir des commandes courtes, ou appeler un programme. Elle est facilement reconnaissable au symbole >>> qui commence chacune des lignes où vous pouvez rentrer du code.
- Créer un programme, auquel on devra donner un nom, qui sera exécuté ensuite dans la console.



Affectation et comparaison

3. Quelle valeur retourne l'algorithme suivant (à la main) ? . . . . . .

$a \leftarrow 3$
$b \leftarrow a + 2$
$c \leftarrow a \times b$
$b \leftarrow c - 10$
Afficher b

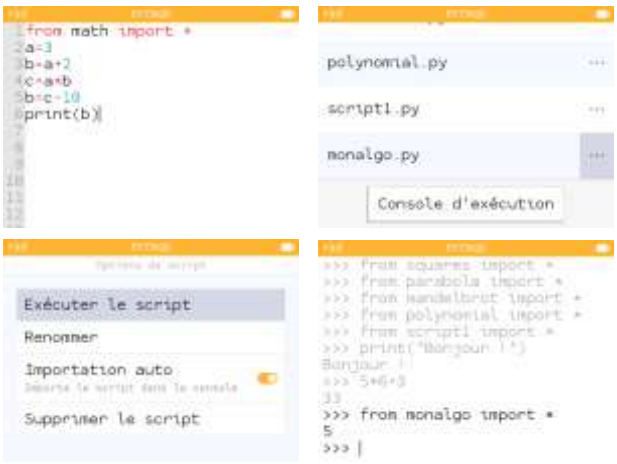
L'affectation d'une valeur à une variable, que l'on note ← en pseudo-code, se fait par le symbole = en Python.
Traduire l'algorithme ci-dessus en Python.

```
a=3



print(  )
```

Programmons-le sur NumWorks.
Sélectionner « ajouter un script », on le nommera `monalgo`. Valider une deuxième fois pour ouvrir l'éditeur de programme.
Une fois le programme saisi, retourner en arrière, aller sur « … » et sélectionner « Exécuter le script ».



4. Compléter les réponses de Python ci-dessous dans les trous laissés vides. Il faut utiliser la console ici.

```
>>> a=5
>>> a*10

>>> b=1.2
>>> b*10

>>> 2**3

>>> 10/5
```

Que permet de faire l'opérateur ** ? . . . . . .

5. Dire si les affirmations suivantes sont vraies ou fausses.

$5 = 6$: . . . . . $5 \neq 6$: . . . . .
$5 > 7$: . . . . . $2 \leq 2$: . . . . .

En Python, les commandes pour effectuer des tests sont :

- comparaison : . . . .
- différent de : . . . .
- plus grand ; plus grand ou égal : . . . .
- plus petit ; plus petit ou égal : . . . .

La réponse est vrai (True) ou faux (False)

```
>>> 5==6
False
>>> 5!=6
True
```

6. On pose $A = 600^{700}$, $B = 700^{600}$ et $C = 650^{650}$.

a. À l'aide de Python, déterminer parmi les réels A et B lequel est le plus petit : . . . . . .
Recopier les commandes saisies.

```
>>> A=600**700
```

b. Classer par ordre croissant les nombres A, B, C.

Types de variables

Il existe plusieurs types de variables en Python.

- Le type **int** désigne les entiers (2, -3, ...) ;
- Le type **float** désigne les nombres à virgule ;
- Le type **string** désigne les chaînes de caractères, *toujours écrites entre guillemets ou apostrophes* : « toto », 'bonjour', ... ;
- Le type **bool** qui désigne les variables booléennes ne pouvant prendre que les valeurs « vrai » ou « faux » (généralement ce sont des résultats de tests, voir ci-dessus).

7. Exécuter les commandes ci-dessous et compléter les trous.

```
>>> a=5
>>> type(a)

>>> a*3

>>> b='bonjour'
>>> type(b)

>>> b*3
```

8. Donner le type des variables suivantes :

- 5-28+1 : . . . .
- 6/3 : . . . . .
- 'abc' : . . . .
- 5!=7 : . . . . .

Les fonctions print et input

Exécuter le script suivant. Que se passe-t-il ? . . .

```
a=8
b=2*a
```

Et pour celui-ci ? . . . . . .

```
a=8
b=2*a
print(b)
```

La fonction **print** permet aussi d'afficher plusieurs variables, à condition de les séparer par des virgules :

```
a=8
b=2*a
print("le double de",a,"vaut",b)
```

Exécuter le script ci-dessous.

```
nom=input("Comment vous appelez-
vous ?")
print("Bonjour",nom)
```

Que permet de faire la fonction **input** ? . . . . . .
. . . . . .

Exécuter le script ci-dessous avec le nombre de votre choix.

```
a=input("Choisissez un entier")
b=2*a
print("le double de",a,"vaut",b)
```

Le résultat est-il celui attendu ? . . . . .
Donner le type de la variable **a** . . . . .
Résoudre le problème rencontré à l'aide de la fonction **int()** qui convertit une variable en type entier (quand c'est possible).
Recopier le script ci-dessous.

```

```

*On retiendra que la fonction **input()** renvoie une variable de type str et que par conséquent selon l'usage que l'on en faire, il faut la convertir en réel, avec la **float()**, ou en entier, avec **int()**.*

TP 2 – Instructions conditionnelles

La résolution de certains problèmes nécessite la mise en place d'un test pour effectuer une tâche :

- si le test est positif, on effectue la tâche ;
- sinon, c'est-à-dire si le test est négatif, on effectue une autre tâche.

Le « sinon » n'est pas obligatoire. En son absence, lorsque le test est négatif, la tâche ne sera pas effectuée et l'algorithme passera à l'instruction suivante.

Le programme ci-dessous détermine, à partir de l'âge a d'une personne, si elle est mineure ou majeure.

```
Si a < 18 alors
    afficher « vous êtes mineur »
Sinon
    afficher « vous êtes majeur »
Fin Si
```

Voici sa traduction en Python.

```
age=input("Entrez votre âge")
age=float(age)
if age < 18:
    print(" .   .   .   .   .  ")
else:
    print(" .   .   .   .   .  ")
```

Le « alors » s'écrit avec deux points en Python. L'indentation délimite une instruction conditionnelle.

1. Compléter le programme, saisir et exécuter le script avec deux âges pour s'assurer du bon fonctionnement, l'un inférieur à 18, l'autre supérieur.

2. On considère un algorithme qui affiche
   - « normal » si la température est strictement supérieure à 5° ;
   - « froid » si elle est comprise entre $-5°$ et 5° ;
   - « grand froid » sinon.

   On propose deux algorithmes, prenant en entrée la température T.
   Compléter le second et le programmer.

```
Si T > 5 alors
    afficher « normal »
Fin Si
Si (T ≥ −5) et (T ≤ 5) alors
    afficher « froid »
Fin Si
Si T < −5 alors
    afficher « grand froid »
Fin Si
```

```
Si . . . alors
    afficher « normal »
Sinon
    Si . . . alors
        afficher « froid »
    Sinon
        afficher « grand froid »
    Fin Si
Fin Si
```

TP 3 – La boucle « pour »

Il est aussi possible de demander à un ordinateur de répéter une même tâche autant de fois que l'on veut. Cela se fait grâce à ce qu'on appelle une **boucle**. Regardons de suite un exemple pour voir l'intérêt de cette notion.

L'algorithme ci-contre affiche les carrés des nombres entiers de 1 à 10.

```
Pour K de 1 à 10
    afficher K²
```

Le principe de fonctionnement est le suivant : l'ordinateur affecte à K la valeur 1 comme il lui est demandé puis effectue les instructions comprises entre « Pour » et « Fin Pour ». Il augmente ensuite automatiquement K de 1 (donc K vaut 2) puis effectue à nouveau le bloc d'instruction. K est alors augmenté de 1 et ainsi de suite jusqu'à ce que K vaille 10.

En Python, la syntaxe est la suivante.

```
for k in range(1,11):
    print(k*k)
```

Notez qu'il faut terminer à 11 et non à 10 (l'explication est que Python comprend ceci : « pour k entier dans l'intervalle $[1;11[$ », par conséquent si l'on écrit range(1,10), la boucle s'arrêtera à $k = 9$ qui est le dernier entier de l'intervalle $[1;10[$).

1. Saisir le script précédent et l'exécuter.

2. Faire afficher les nombres impairs de 1 à 99.

3. Faire afficher les puissances de 2 de 1 à $2^{12} = 4096$.

4. Que fait l'algorithme ci-contre ? Combien retourne-t-il ? . . . . . .

```
S ← 0
Pour K de 1 à 100
    S ← S + K
Fin Pour
Retourner S
```

5. Au premier 1er janvier 2020, Sophie a placé 1000 € sur un compte rémunéré à 2 %. Elle verse chaque 1er janvier une somme de 100 € sur ce compte.
On suppose qu'aucun retrait n'est effectué sur ce compte.

a. Justifier que Sophie disposera de 1120 € sur ce compte le 1er janvier 2021, puis calculer la somme au 1er janvier 2022.

b. Même question pour le 1er janvier 2040, à l'aide d'un algorithme.

6. On considère la suite (de Fibonacci) des nombres
0 ; 1 ; 1 ; 2 ; 3 ; 5 ; 8 ; 13 ; 21 ; 34 ; …
dans laquelle un terme est la somme des deux précédents.
Écrire un algorithme qui demande à l'utilisateur un entier $n \geq 1$ et qui renvoie le n^{e} terme de la suite. Par exemple pour $n = 8$, l'algorithme renverra 13.

7. Depuis le 1er janvier 2000, Robert fume 20 cartouches de cigarettes par an. En 2000, cela représentait un budget de 700 € par an.
Chaque année, le prix du tabac augmente de 6 %.
Compléter l'algorithme ci-dessous pour calculer l'argent dépensé par Robert en cigarettes du 1er janvier 2000 au 31 décembre 2020, puis le programmer.

```
A ← 700  #dépense annuelle
S ← 700  #dépense cumulée
Pour k de … à …
    A ← ……
    S ← ……
Fin Pour
Afficher …
```

TP 4 – La boucle « tant que »

En algorithmique, on peut être amené à répéter un bloc d'instructions tant qu'une condition est vérifiée. En langage naturel, une telle répétition en boucle peut se formuler par « tant que », traduit par « while » dans la plupart des langages.
Si la condition qui suit l'instruction « tant que » est vérifiée, alors le programme exécute toutes les instructions du bloc qui suit ; à la fin de cette exécution il regarde à nouveau si la condition est vérifiée et ainsi de suite jusqu'à ce qu'elle ne le soit plus.

```
K ← 1
Tant que K ≤ 100
    afficher K²
Fin Tant que
    K ← K + 1
```

1. Toute boucle « pour » peut être remplacée par une boucle « tant que » à l'aide d'un compteur.
L'algorithme ci-contre affiche lui aussi les carrés des entiers de 1 à 100.
Voici sa traduction en Python. L'expérimenter.

```
k=1
while k<=10:
    print(k*k)
    k=k+1
```

En revanche il existe des utilisations de « tant que » qui ne peuvent pas être remplacée par celle de « pour ».

2. L'effectif d'une population de bactéries double chaque jour. Si la population est de 100 bactéries le premier jour, au bout de combien de jours dépassera-t-elle 2 000 bactéries ? (à la main) . . . . . .

Si la question avait été « au bout de combien de temps la population dépassera-t-elle 6 milliards de bactéries ? », le calcul à la main aurait été fastidieux. Compléter l'algorithme ci-dessous et le programmer.

```
J ←
P ←
Tant que P ≤
    P ←
    J ←
Fin Tant Que
Retourner
```

J compte le nombre de jours et P est le nombre de bactéries le J-ième jour.

3. Une balle est lâchée de 100 mètres de haut et rebondit aux $4/5^{ème}$ de sa hauteur à chaque rebond.
   a. Calculer la hauteur du premier et du second rebond.
   b. On considère que la balle ne rebondit plus si le rebond est inférieur à 1 cm. Au bout de combien de rebonds la balle ne rebondit plus ?

4. À l'aide d'un algorithme, déterminer en quelle année Sophie aura dépassé les 10 000 € sur son compte.

5. Déterminer le premier terme de la suite de Fibonacci plus grand que 1000.

Programmation d'un jeu

6. Nous allons créer un programme dans lequel l'ordinateur choisi un nombre au hasard entre 1 et 10 puis nous demande de le deviner.

```
import random
nbadeviner=random.randint(1,10)
essai=0
while essai!=nbadeviner:
   essai=int(input("Essai :"))
print("C'est gagné !")
```

Comment s'appelle la variable contenant le nombre choisi par l'ordinateur ? . . . . . .

Que fait l'opérateur `!=` ? . . . . . .

Résumer en français les trois dernières lignes du programme : . . . . . .
. . . . . .
. . . . . .

7. Modifions le programme pour que l'ordinateur indique « c'est plus », « c'est moins » ou « c'est gagné » :

```
import random
nbadeviner=random.randint(1,10)
essai=0
while essai!=nbadeviner:
   essai=int(input("Essai : "))
   if essai<nbadeviner:
      print("                ")
   elif essai>nbadeviner:
      print("                ")
   else:
      print("                ")
```

8. Modifier le programme pour compter le nombre de coups joués.

```
>>> (executing file "<tmp 1>")
Essai : 5
C'est moins
Essai : 2
C'est plus
Essai : 4
C'est gagné en 3 coups.
```

TP 5 – Les fonctions en Python

Considérons la fonction f qui a un nombre associe son double. Cela s'écrit $f: x \mapsto 2x$.
En Python pour définir une fonction, on utilise la commande **def** ; pour afficher la valeur à retourner, on utilise **return**. Ne surtout pas utiliser de print dans une fonction, sauf pour débuguer.
La définition de la fonction s'arrête lorsque l'indentation se termine.
Exécuter le script suivant.

```
def double(nombre):
   return 2*nombre
```

Dans la console, il suffira de saisir :

```
>>> double(10)
20
>>> double(-8)
-16
```

Un des avantages des fonctions est de pouvoir les « enchaîner ».

```
>>> double(double(10))
40
```

1. Écrire une fonction **age** dont l'appel donnera par exemple :

```
>>> age(15)
'vous êtes mineur'

>>> age(19)
'vous êtes majeur'
```

2. Écrire une fonction **pair(n)** qui prend en argument un entier n et retourne 1 s'il est pair, 0 sinon.
   Indication : expérimenter la commande n%2 où n est un entier

```
>>> pair(10)
1
>>> pair(11)
0
```

3. Écrire une fonction **sophie(n)** qui renvoie l'argent disponible au 1^{er} janvier de l'année n.

```
>>> sophie(2021)
1120
```

4. Écrire une fonction **fibo(n)** qui renvoie le n^e nombre de la suite de Fibonacci.

```
>>> fibo(8)
13
```

5. Écrire une fonction **somme(n)** qui calcule la somme
$$1 + 2 + 3 + \cdots + n$$
où n est donné en argument à la fonction.

```
>>> somme(4)
10
```

6. Écrire une fonction **sommefibo(n)** qui renvoie la somme des n premiers nombre de la suite de Fibonacci.

```
>>> sommefibo(4)
7
```

7. Écrire une fonction qui prend en argument un entier *à deux chiffres* et renvoie la somme des ses chiffres.
On pourra utiliser
   - `a%b` qui renvoie le reste de a par b ;
   - `a//b` qui renvoie le quotient de a par b.

```
>>> sommechiffres(59)
14
```

Généraliser à un nombre quelconque de chiffres.

```
>>> sommechiffres(25155)
18
```

TP 6 – Les listes en Python

1. Une liste est une variable qui contient plusieurs variables, elle se note entre crochets : […]
La longueur d'une liste s'obtient par `len(L)`, le i^{e} terme par la commande `L[i]` (en commençant à 0).

```
>>> maliste=[1,8,6,-7]
>>> len(maliste)
4
>>> maliste[0]
1
```

On peut « concaténer » deux listes en une seule en utilisant l'opérateur + ; cela permet par exemple d'ajouter un élément à la fin d'une liste :

```
>>> maliste2=[50,51]
>>> maliste+maliste2
[1, 8, 6, -7, 50, 51]
```

Pour récupérer le rang de la première occurrence d'un élément, on utilise la méthode `list.index()`.

```
>>> maliste.index(6)
2
```

Voici la liste des carrés des entiers de 0 à 6 :

```
>>> [n**2 for n in range(0,7)]
[0, 1, 4, 9, 16, 25, 36]
```

2. Écrire une fonction **moyenne()** qui calcule la moyenne des éléments d'une liste.

```
>>> moyenne([3,2,7])
4
```

3. Donner la liste des entiers compris entre 1 et 200 qui sont des multiples de 7, mais qui ne sont ni des multiples de 5, ni de 2.

4. Écrire une fonction **listeversnombre()** qui convertit une liste de chiffres en le nombre composé de ces chiffres.

```
>>> listeversnombre([2,3,1,5])
2315
```

5. Écrire la fonction **nombreversliste()**, inverse de la précédente.

6. Écrire une fonction **doublon()** qui supprime les doublons dans une liste.

```
>>> doublon([4,5,1,5,7,1])
[4,5,1,7]
```

7. Soit (u_n) la suite de Fibonacci : $u_0 = 0$, $u_1 = 1$ et $u_{n+2} = u_{n+1} + u_n$. Écrire une fonction qui renvoie la liste des premiers termes.

```
>>> fibonacci(8)
[0, 1, 1, 2, 3, 5, 8, 13]
```

Calendrier perpétuel

La formule de Zeller permet de calculer le jour de la semaine (lundi, mardi, …) à partir de sa date. Cette formule fonctionne pour le calendrier grégorien, c'est-à-dire pour les dates postérieures au 15 octobre 1582.
On note
- j la date du jour ;
- m le numéro du mois : 3 pour mars, 4 pour avril, …, 13 pour janvier et 14 pour février ;
- a le numéro de l'année. Dans la formule, janvier et février sont comptés pour les mois 13 et 14 de l'année précédente. Par exemple, pour calculer le jour du 12 janvier 2007, on cherchera le jour du « 12/13/2006 » ;
- h le numéro du jour avec 0 pour samedi, 1 pour dimanche, 2 pour lundi, …, 6 pour vendredi.

On note $\lfloor x \rfloor$ la partie entière d'un nombre x, par exemple $\lfloor 13{,}52 \rfloor = 13$ et $\left\lfloor \frac{5}{3} \right\rfloor = 1$. Ainsi $\left\lfloor \frac{a}{b} \right\rfloor$ est le quotient de la division euclidienne de a par b.
La formule de Zeller affirme que h est le reste de la division euclidienne par 7 de

$$j + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + a + \left\lfloor \frac{a}{4} \right\rfloor - \left\lfloor \frac{a}{100} \right\rfloor + \left\lfloor \frac{a}{400} \right\rfloor.$$

Par exemple, pour le 12 janvier 2007, c'est-à-dire le « 12/13/2006 », on a $j = 12$, $m = 13$ et $a = 2006$.
Puisque $\left\lfloor \frac{13(m+1)}{5} \right\rfloor = \left\lfloor \frac{13\times14}{5} \right\rfloor = \lfloor 36{,}4 \rfloor = 36$, $\left\lfloor \frac{2006}{4} \right\rfloor = 501$, $\left\lfloor \frac{2006}{100} \right\rfloor = 20$ et $\left\lfloor \frac{2006}{400} \right\rfloor = 5$, le calcul est

$$12 + 36 + 2006 + 501 - 20 + 5 = 2540.$$

Comme $2540 = 7 \times 362 + 6$, on en déduit $h = 6$ et le jour cherché est vendredi.

1. Vérifier sur un exemple « à la main » (avec la date d'aujourd'hui par exemple).

2. Expérimenter les commandes suivantes dans la console Python.

```
>>> 5//3
```

```
>>> 5%3
```

```
>>> 13//7
```

```
>>> 13%7
```

a et b étant deux entiers, que calcule :

- `a//b` ? . . . . . .
- `a%b` ? . . . . . .

Voici le programme et un exemple d'exécution.

```
j=int(input("Jour : "))
m=int(input("Mois : "))
a=int(input("Année : "))

if m<=2:
   m=m+12
   a=a-1

h=(j+(13*(m+1))//5+a+a//4-
a//100+a//400)%7

J=['samedi','dimanche','lundi','mar
di','mercredi','jeudi','vendredi']

print("C'était un",J[h])
```

```
>>> (executing file « zeller.py")
Jour : 12
Mois : 1
Année : 2007
C'était un vendredi
```

3. Quelle est le type de la variable `J` ? . . . . . .

4. Si `h` est un entier entre 0 et 6, que renvoie `J[h]` ?
. . . . . .

TP 7 – Les chaînes de caractères

Une chaîne de caractère (type str) est une liste de caractère délimitée par des guillemets.
On utilise alors les mêmes commandes que pour les listes.

```
>>> ch='merci'
>>> len(ch)
5
>>> ch[2]
'r'
>>> ch+'catrice'
'mercicatrice'
```

1. Écrire une fonction `unsurdeux` qui renvoie une chaîne en ne prenant qu'un caractère sur deux.

```
>>> unsurdeux('Bonjour !!!')
'Bnor!!'
```

2. Écrire une fonction `joie` qui renvoie une chaîne en répétant les caractères comme ci-dessous.

```
>>> joie(ch)
'meerrrccccciiiii'
```

3. Écrire une fonction `voyelle` qui ne renvoie que les voyelles d'une chaîne de caractère.

```
>>> voyelle('Bonjour !!!')
'oou'
```

4. Donner une nouvelle version de la fonction `sommes-chiffres` sans utiliser la division euclidienne.

```
>>> sommechiffres(25155)
18
```

TP 8 – Problèmes

1 **Le code EAN-13** (*European Article Numbering*) est un code-barres utilisé par le commerce et l'industrie.

Les 12 premiers chiffres sont propres à l'article, le treizième est la clé de contrôle calculée de la façon suivante.
Si les douze premiers chiffres sont $a_1 a_2 \ldots a_{12}$, on calcule le reste r dans la division euclidienne par 10 de

$$S = a_1 + a_3 + \cdots + a_{11} + 3(a_2 + a_4 + \cdots + a_{12}).$$

- si ce reste est 0, la clé est 0
- sinon, la clé est $10 - r$.

1. Écrire une fonction `clé` calculant la clé à partir des douze premiers chiffres.

```
>>> clé(590123412345)
7
```

2. Écrire une fonction `clé_valide` calculant la clé à partir des douze premiers chiffres qui renvoie `True` ou `False` selon qu'une clé saisie de treize chiffre est valide ou pas.

3. Modifier la fonction `clé` pour qu'elle admette en paramètre une clé avec des éventuels espaces ou tirets.

```
>>> clé(5 9012-3412-345)
7
```

2 **Conjecture de Syracuse.** On choisit un nombre entier plus grand que zéro ; s'il est pair, on le divise par 2 ; s'il est impair, on le multiplie par 3 et on ajoute 1. En répétant l'opération, on obtient une suite d'entiers positifs dont chacun ne dépend que de son prédécesseur.

Exemple 1. En partant de 1 on a successivement $1 \to 4 \to 2 \to 1 \to 4 \to 2 \to 1 \to \ldots$ On constate que la suite des nombres obtenus se répète (c'est un cycle de longueur 3).

Exemple 2. On part ici de 3. On a alors

$$3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1,$$

et ce n'est pas la peine de continuer, puisque ayant obtenu 1, on sait que l'on va retomber indéfiniment dans le cycle $1 \to 4 \to 2 \to 1$ de l'exemple 1.

Les mathématiciens ont toutes les raisons de penser que quel que soit le nombre initial choisi, on retombe toujours sur 1, mais aucun n'a réussi à prouver ce résultat jusqu'à présent, ce qui justifie le statut de conjecture de ce résultat.

On appelle **durée de vol** d'un entier le nombre de nombres calculés avant de tomber sur 1 et **hauteur de vol** la plus grande valeur prise par la suite des nombres obtenus avant de tomber sur 1. Ainsi la durée de vol de 3 est 7 et sa hauteur de vol est 16.

Compléter le tableau suivant.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
durée	0		7												
hauteur	1		16												

1. Écrire un algorithme qui renvoie le vol d'un entier n saisi par l'utilisateur.
La commande `print(end=' ')` permet d'éviter les retours à la ligne.

```
Entrez un entier : 3
Le vol de 3 est : 3, 10, 5, 16, 8,
4, 2, 1
```

2. Rajouter la durée de vol, puis la hauteur de vol.

```
Entrez un entier : 3
Le vol de 3 est : 3, 10, 5, 16, 8,
4, 2, 1
Sa durée de vol est 8 et sa hau-
teur est 16
```

3. Modifier les algorithmes précédents en fonctions qu'on appellera `vol`, `duree` et `hauteur`. Si les listes vous font peur, omettez la fonction `vol`.

```
>>> vol(7)
[7, 22, 11, 34, 17, 52, 26, 13,
40, 20, 10, 5, 16, 8, 4, 2, 1]
```

```
>>> duree(7)
16
```

```
>>> hauteur(7)
52
```

4. Écrire une fonction `dureemax` prenant en entrée deux entiers a et b et renvoyant l'un des entiers compris entre a et b ayant la plus grande durée de vol ainsi que cette durée de vol maximale.

```
>>> dureemax(8,14)
(9, 19)
```

5. Faire de même avec une fonction `hauteurmax` renvoyant l'un des entiers compris entre a et b et ayant la plus grande hauteur ainsi que cette hauteur maximale.

```
>>> hauteurmax(1,15)
(15, 160)
```