TP 1 : Premier contact avec Python

▶ L'environnement Pyzo

L'espace de travail de Pyzo est divisé en plusieurs fenêtres.
▶ À gauche se trouve l'éditeur, que l'on utilisera principalement pour écrire des programmes de plus d'une ligne. Durant la saisie du code dans cette fenêtre, rien ne se produit tant qu'on ne demande pas l'exécution du code (Menu Exécuter -> Démarrer le script ou Ctrl + ↑ Shift + E).
Pour être exécuté, un script a besoin d'avoir été préalablement sauvegardé.
▶ Le code peut être scindé en plusieurs cellules, chacune d'entre elles commençant par un double dièse (##). Le raccourci Ctrl + Entrée permet alors de n'exécuter que la cellule active.
▶ La fenêtre en haut à droite est la console Python, dans laquelle s'afficheront les résultats des fichiers exécutés, et où on peut également saisir des instructions.
On peut notamment y faire appel à la documentation de Python à l'aide de la commande `help`.
Vous pouvez par exemple consulter les documentations des fonctions `print` ou `abs` à l'aide de `help(print)` et `help(abs)`.

Veillez à soigneusement enregistrer tous vos programmes, afin que ceux-ci soient facilement accessibles lors des prochains TP. Vous apporterez donc un soin particulier aux noms donnés aux dossiers et aux fichiers !

▶ Python en tant que calculatrice

Exercice 1 Les opérations de base sont implémentées dans Python. Essayez les commandes suivantes dans la console, comprenez ce qu'elles font (quitte à essayer avec plusieurs valeurs si nécessaire).
Essayez notamment de bien comprendre les différences entre des commandes similaires.

- `2+3`
- `4*3`
- `2**4`
- `16**0.5`
- `17/5`
- `17//5`
- `17%5`
- `True and False`
- `True or False`
- `True + True`

Exercice 2 Les types en Python
On rappelle qu'en Python, toute expression possède un type, et que celui-ci s'obtient à l'aide de la commande `type`. En particulier, `2` est du type **`int`** alors que `2.` est du type **`float`**.
Par exemple, `type(2)` nous informe que 2 est du type **`int`** et `type(2/5)` nous informe que 2/5 (qui vaut 0.4) est du type **`float`**.
Testez le type des expressions suivantes :

- `3`
- `3.`
- `2+4`
- `2. + 4`
- `2. + 4.`
- `1 /2`
- `12 / 3`
- `12. / 3`
- `13 //3`
- `13.// 3`
- `13 // 3.`
- `14 %3`
- `14. %3`
- `4**(0.5)`
- `True`
- `(2<=4) or False`

Exercice 3 La bibliothèque `math`
Les commandes qui suivent, bien que très utiles, ne sont pas implémentées nativement dans Python et font partie du module (ou bibliothèque) `math`. Autrement dit, pour pouvoir les utiliser il faut avoir précédemment importé le module par exemple à l'aide de **`import`** `math`, puis appeler la commande précédée du préfixe `math`.
Par exemple `math.sqrt(36)` ou `math.floor(2.6)`.
Si le nom de la bibliothèque est trop long, il est possible de la renommer : **`import`** `math as m` nous permet

alors d'utiliser `m.sqrt(36)` au lieu de `math.sqrt(36)`.
Une solution plus facile (mais qui peut être dangereuse lorsque deux bibliothèques contiennent des fonctions de même nom) est d'utiliser **`from math import *`**.
On peut ensuite directement utiliser les commandes `sqrt(36)` ou `floor(2.6)`, sans les faire précéder de `math`.
Notons que si on n'a besoin que d'une ou deux fonctions d'une bibliothèque, on peut n'importer que ces fonctions, par exemple **`from math import floor,sqrt`**.
Si vraiment un nom de fonction vous déplaît, il est même possible de le changer, bien que ce ne soit pas conseillé.

```
from math import sqrt as racine
racine(4)
```

La bibliothèque `math` contient aussi deux constantes, nommées `math.pi` et `math.e` (ou `pi` et `e` suivant la manière dont vous avez importé la bibliothèque), dont la signification se passe de commentaires.
Essayez rapidement les commandes suivantes après avoir utilisé **`from math import *`**

- `sqrt(4)`
- `sqrt(4.)`
- `floor(2.5)`
- `floor(2.9)`
- `floor(-2.4)`
- `ceil(2.5)`
- `ceil(3.1)`
- `log(e)`
- `log10(100)`
- `exp(1)`
- `cos(90)`
- `cos(pi)`
- `sin(radians(30))`
- `degrees(-5*pi/6)`
- `sqrt(-4)`
- `log(exp(5))`

Exercice 4 Surcharge

Si on souhaite par exemple ajouter 2 à une variable `a`, on peut utiliser `a = a+2`.
Mais on peut également utiliser l'opérateur `+=`. Ainsi, l'instruction `a +=2` est équivalente à `a = a+2`
Saurez vous prévoir les valeurs finales de `a` et `b` à l'issue du programme suivant ? Le vérifier.

```
a = 2
b = 3
b += 1
a *= b
a **= 2
```

Exercice 5 Couples de variables

On peut modifier deux ou plusieurs variables à la fois. Essayez de comprendre les codes suivants, et pourquoi ils ne produisent pas le même résultat :

```
n,p = 5,-3
print(n,p)
n,p = p+2,n
print(n,p)
```

```
n,p = 5,-3
n = p+2
p = n
print(n,p)
```

Exercice 6 Échange de deux variables

Créer deux variables `u` et `v` contenant des valeurs numériques (différentes) de votre choix.
Écrire un programme qui échange les valeurs de `u` et `v` suivant chacune des méthodes suivantes :

1. En utilisant temporairement une troisième variable.
2. En commençant par la commande `u = u+v`.
3. En modifiant simultanément `u` et `v`.

Exercice 7 Des conversions

AD

1. On rappelle que dans 2π radians, il y a 360 degrés, chacun étant divisé en 60 minutes, elles-mêmes divisées en 60 secondes.
   (a) Écrire un programme qui calcule la mesure en radians d'un angle de $37°18'26''$ sous la forme d'un nombre flottant.

(b) Écrire un programme qui crée trois variables `deg`, `min` et `sec` de type **`int`** donnant la mesure en degrés d'un angle de $\frac{8\pi}{17}$.

2. Combien d'années, de mois et de jours aurez-vous vécu lorsque vous fêterez votre milliardième seconde ?
On prendra 30 jours comme durée moyenne d'un mois, et on ne se préoccupera pas des années bissextiles.

▶ Fonctions

Pour créer une fonction, on utilise la syntaxe **`def`** `nom(parametre) :`
Les instructions qui composent le corps de la fonction devant alors être **indentées**, par exemple à l'aide de tabulations (touche ⇄).
Pour qu'une fonction retourne une valeur, on emploie l'instruction **`return`**.
Une fonction peut également retourner un couple (ou un triplet, etc) de valeurs. Par exemple :

```
def carre(a) :
        aire = a**2
        perimetre = 4*a
        return aire,perimetre
```

Exercice 8 Écrire une fonction `volume_sphere`, qui prend comme paramètre un nombre `r` et renvoie le volume d'une sphère de rayon `r`.
Sur le même principe, écrire une fonction `volume_cylindre` qui prendrait comme paramètres la hauteur et le rayon d'un cylindre.

Exercice 9 Variables locales vs variables globales
Savez-vous prédire ce que valent les variables `x` et `y` après exécution des codes suivants ? Le vérifier (par exemple avec **`print`**`(x)` et **`print`**`(y)`).

```
x = 2
y = 1
def carre(x) :
        y = x**2
        return y

carre(4)
```

```
x = 2
def carre(x) :
        y = x**2
        return y

carre(4)
```

```
x,y = 2,1
def carre(x) :
        global y
        y = x**2
        return y

carre(4)
```

▶ Instructions conditionnelles

On rappelle que la syntaxe d'une instruction conditionnelle est la suivante :

```
if (test1) :
        Premier bloc
        d'instructions
elif (test2) :
        Second bloc
else :
        Dernier bloc
```

Si la condition définie dans le `test1` (par exemple `a>3` ou `x**2==y+1`) est réalisée, alors les instructions du premier bloc (gare à l'indentation !) sont exécutées.
Sinon, et que la condition `test2` est réalisée, alors les instructions du second bloc sont exécutées.
Enfin, si aucune des deux conditions n'est vérifiée, alors ce sont les instructions du troisième bloc qui sont exécutées.

Le **elif** et le **else** sont facultatifs, et il ne sert à rien de les écrire (ou plutôt, **il ne faut pas les écrire**) si les blocs d'instructions correspondants sont vides.
Un bloc d'instructions doit toujours être indenté, c'est la fin de l'indentation qui marque la fin du bloc d'instructions.
Enfin, il est possible d'utiliser plusieurs **elif** à la suite, par exemple si quatre cas de figure sont possibles :
`if (cas1) : ... elif (cas2) : ... elif (cas3) : ... else : ...`
Les conditions les plus utilisées sont de la forme `a>b`, `a<b`, `a>=b`, `a<=b`, `a==b` et `a !=b`.
⚠ Pour tester l'égalité de deux variables, on utilise ==, l'usage de = étant réservé à l'affectation d'une valeur à une variable (par exemple `a=2`).

Exercice 10 Écrire une fonction `maximum` qui prend en paramètres deux nombres (entiers ou flottants) `a` et `b` et renvoie le plus grand des deux.

Exercice 11 Écrire une fonction `mention` qui prend comme paramètre la moyenne au bac et retourne une chaîne de caractère contenant la mention correspondante (on considérera qu'entre 10 et 12, on obtient la mention «Passable»).

Exercice 12
1. Écrire une fonction `nbe_racines` qui prend en paramètres trois réels `a`,`b` et `c` avec $a \neq 0$, et retourne le nombre de solutions réelles de l'équation $ax^2 + bx + c = 0$.
2. Écrire une fonction similaire qui retourne la ou les racines **réelles** de $ax^2 + bx + c = 0$ s'il y en a.

► Pour aller plus loin

Exercice 13 Écrire une fonction `bissextile` qui prend comme paramètre un entier correspondant au numéro d'une année, et retourne un booléen qui vaut `True` si cette année est bissextile et `False` sinon. AD
On rappelle qu'une année bissextile est une année qui est divisible par 4*, sauf si elle est divisible par* 100*, auquel cas elle est bissextile si et seulement si elle est divisible par* 400*.*

Exercice 14 D
1. Réécrire la fonction `maximum` sans utiliser **if**. *Indication : trouver une relation liant* $\max(x, y)$, $|x - y|$, x *et* y.
2. À l'aide de la fonction `maximum`, écrire une fonction `minimum` sans utiliser la commande **if**.
3. Toujours à l'aide de `maximum` écrire une fonction `max4` qui prend en paramètres quatre nombres et retourne le plus grand des 4.
4. Toujours sans utiliser de **if**, écrire une fonction `sort3` qui prend en paramètres trois nombres et retourne un triplet formé des mêmes nombres, mais rangés dans l'ordre croissant.

Exercice 15 Écrire une fonction `anniversaires`, qui prend comme paramètres 6 entiers `j_naiss`, `m_naiss`, `a_naiss`, `jour`, `mois`, `annee` où TD

- les trois premières représentent la date de naissance d'une personne
- les trois dernières représentent sa date de mort (ou la date actuelle dans le cas de personnes encore vivantes)

La fonction `anniversaires` doit alors retourner le nombre de fois où cette personne a fêté son anniversaire entre les deux dates. On considérera qu'une personne décédée le jour de son anniversaire a fêté ce dernier anniversaire.
⚠ La fonction ne doit pas retourner l'âge du décès, mais bien le nombre d'anniversaires. La principale difficulté venant des années bissextiles et des personnes nées un 29 février. Par exemple, le compositeur italien Rossini (29 février 1792 – 13 novembre 1868), bien que décédé à 76 ans n'a fêté que 18 fois son anniversaire, car le 29 février ne se produit pas tous les ans...

Exercice 16 Écrire une ligne de commande permettant de déterminer le dernier chiffre de 2017^{2019} sans avoir à afficher ce nombre en entier. D
D'ailleurs, combien de chiffres possède 2017^{2019} ?

Correction des exercices du TP 1

Solution de l'exercice 4

Après les deux premières lignes, a vaut 2 et b vaut 3.
Après la troisième, b vaut $3+1=4$. Donc après la quatrième, a vaut $2\times 4=8$.
Et à la fin, a vaut $8^2=64$. □

Solution de l'exercice 5

Remarquons que `print(n,p)` permet d'afficher en même temps la valeur de deux variables.
La différence entre ces deux programmes tient au fait que lors d'une affectation, Python calcule d'abord la valeur du membre de droite, puis l'affecte aux variables concernées.
Ainsi, dans le premier programme, il commence par calculer le couple $(p+2,n)$, qui vaut donc $(-1,5)$.
Et donc la valeur -1 est affectée à n et la valeur 5 est affectée à p.
Dans le second programme, la valeur de n est changée à la ligne 2, puis c'est cette nouvelle valeur qui est utilisée à la ligne 2 pour définir p.
Donc à la fin n et p ont la même valeur. □

Solution de l'exercice 6

1. C'est sûrement la première méthode qui vient à l'esprit : on veut commencer par donner à v la valeur contenue dans u. Mais si l'on fait ceci, on va effacer la valeur stockée dans v. Un moyen d'y remédier est de commencer par stocker cette valeur dans une autre variable.

```
1 w = v
2 v = u
3 u = w
```

2. Il s'agit de remarquer que si on connaît les valeurs de u+v et v, alors on connaît celle de u puisque $u=(u+v)-v$.
Et de même, on peut retrouver v à partir de u+v et u.

```
1 u = u+v
2 v = u-v
3 u = u-v
```

Notons que cette méthode[1] ne fonctionne que si u et v contiennent des valeurs numériques, et pas des chaînes de caractères.

[1] Contrairement aux deux autres.

3. La plus simple, utilisant pleinement les fonctionnalités de Python :

```
1 u,v = v,u
```

□

Solution de l'exercice 7

1.a. Un degré est $\frac{2\pi}{360}$ radians, une minutes est $\frac{2\pi}{360\times 60}$ radians et une seconde est $\frac{2\pi}{360\times 60^2}$ radians.

```
1 from math import pi
2 deg = 37
3 minutes = 18
4 secondes = 26
5 radians =2*pi/360*(deg + minutes/60 + secondes/60**2)
6 print(radians)
```

On obtient alors 0.651134 radians.

1.b. Inversement, un radian vaut $\frac{360}{2\pi}$ degrés, soient $\frac{360}{2\pi}60^2$ secondes.
Le nombre entier de secondes de $\frac{8\pi}{17}$ est donc $\left\lfloor \frac{8\pi}{17}\frac{360}{2\pi}60^2 \right\rfloor$.
Il faut ensuite déterminer combien de minutes entières sont contenues dans ces secondes, ce qui est obtenu à l'aide d'une division entière (opérateur //).
Puis une fois le nombre de minutes obtenues, il reste à voir combien de degré entiers sont contenus dans ces minutes.

```
from math import floor,pi
sec = floor(8*pi/17*360*60**2/(2*pi))
min = sec//60
sec = sec % 60 #Le nombre de secondes restantes
deg = min // 60
min = min % 60 # Le nombre de minutes restantes
print('Notre angle vaut ' + str(deg) + '°' + str(min) + "'" + str(sec) + "''")
```

L'emploi de `//` et `%`, bien que très pratique n'est pas obligatoire.
On peut par exemple remplacer `min = sec // 60` par `min = floor(sec/60)`.
Et alors on peut remplacer `sec =sec %60` par `sec = sec - 60*min`.

2. C'est le même principe que précédemment : une journée contient $26 \times 60 \times 60 = 86400$ secondes. Avec peut-être une nuance à apporter au calcul du nombre d'années : si on prend 12 mois de 30 jours, une année ne comporte que 360 jours, ce qui sur un grand nombre d'années peut engendrer une erreur conséquente.
Nous commencerons donc par calculer le nombre d'années.

```
from math import floor
jours = floor(1e9/86400)
annees = jours//365
jours = jours-annees*365
mois = jours//30
jours = jours % 30
mois = mois % 12
print('Dans un milliard de secondes, il y a environ ' + str(annees) + '
ans, ' + str(mois) + ' mois, ' + str(jours) + ' jours.')
```

Ce qui nous donne environ 31 ans, 8 mois et 19 jours.

Remarque

L'âge exact que vous aurez en fêtant votre milliard de secondes dépend tout de même de votre date de naissance (combien de mois de 28,29,30 ou 31 jours seront passés ?).
Sans parler de l'heure où vous avez vu le jour.

□

Solution de l'exercice 8

Rappelons que le volume d'une sphère est $\frac{4}{3}\pi R^3$ où R est le rayon.

```
from math import pi
def volume_sphere(R) :
        return(4/3*pi*R**3)
```

De même, le volume d'un cylindre de hauteur h et de rayon r est $\pi r^2 h$.

```
from math import pi
def volume_cylindre(h,r) :
        return(pi*r**2*h)
```

□

Solution de l'exercice 9

L'énoncé comporte un petit problème, car il supposait que la mémoire de Python était remise à zéro avant chaque programme (et notamment que les variable a et b étaient effacée de la mémoire). C'est probablement l'hypothèse qu'on fait si on essaie de réfléchir sur le papier, mais en pratique, on a tendance à essayer les trois programmes à la suite...
Dans les trois cas, x vaut 2, le fait de faire appel à la fonction `carre` (dont le paramètre s'appelle pourtant x dans le code) ne change pas la valeur de x.

Variable muette

Il est évident que si notre fonction était `def carre(t) : ...`, alors nous ne nous poserions pas la question de la valeur de x.

De la même manière, la variable y n'est pas affectée par l'appel à `carre` dans le premier programme.
Dans le second, c'est pire, puisque Python va nous dire qu'il n'a jamais entendu parler d'une variable nommée y (ou plus exactement qu'une telle variable n'est pas définie), puisque le y que nous avons utilisé ne désigne qu'une variable locale.
Enfin, dans le troisième programme, la variable y étant déclarée comme globale, l'appel à `carre` va modifier la valeur de y, qui va donc prendre la valeur 16 ($=4^2$). □

Et si $a = b$?

Si $a = b$, alors le maximum vaut a, mais il vaut aussi b. Donc que la fonction contienne des inégalités larges ou strictes n'a aucune importance.

Solution de l'exercice 10

Il suffit de tester si a est plus grand que b. Si c'est le cas, a est le maximum, sinon, c'est b.

```
def maximum(a,b) :
        if a>b :
                return(a)
        else :
                return(b)
```

Notons que Python possède déjà une fonction qui a cet effet, elle se nomme `max`. □

Solution de l'exercice 11

```
def mention(moyenne) :
        if moyenne>=16 :
                return('Très bien')
        elif moyenne>=14 :
                return('Bien')
        elif moyenne>=12 :
                return('Assez bien')
        elif moyenne>=10 :
                return('Passable')
        else :
                return('Try again !')
```

□

Solution de l'exercice 12

1. Bien entendu, tout dépend du signe du discriminant.

```
def nb_racines(a,b,c) :
        delta = b**2 - 4*a*c
        if delta>0 :
                return 2
        elif delta==0 :
                return 1
        else :
                return 0
```

2. Notre fonction va donc, suivant les cas, retourner un couple de flottants, un flottant, ou rien !

```
from math import sqrt
def racines(a,b,c) :
        delta = b**2 - 4*a*c
        if delta > 0 :
                return ((-b + sqrt(delta))/(2*a),(-b - sqrt(delta))/(2*a))

        elif delta==0 :
                return -b/2/a
```

□

Solution de l'exercice 13

Il faut tester si le numéro de l'année est divisible par 4.
Si ce n'est pas le cas, l'année n'est pas bissextile.
Si c'est le cas, on teste si elle est divisible par 100. Si ce n'est pas le cas, elle est bissextile, sinon il faut encore vérifier si elle est divisible par 400 ou non.

Donnons trois options. La première est la plus naïve, et la plus intuitive :

```
def bissextile(annee) :
        if annee%4==0 :
                if annee%100==0 :
                        if annee%400==0 :
                                return(True)
                        else :
```

```
                                return(False)
                        else :
                                return(True)
                else :
                        return(False)
```

La seconde, un peu plus courte, repose sur le fait que l'emploi de **return** interrompt l'exécution de la fonction.
Donc le tout dernier `return` ne sera exécuté que si aucune instruction `return` n'a été précédemment exécutée, donc si et seulement si l'année n'est pas bissextile.

> **Est-ce bien utile ?**
> Si cette astuce nous a permis d'économiser 3 lignes, le programme n'est pas forcément plus facile à comprendre en première lecture...

```
def bissextile(annee) :
        if annee %4 == 0 :
                if annee %100==0 :
                        if annee%400==0 :
                                return(True)
                        else :
                                return(True)
        return(False)
```

La dernière est bien plus astucieuse et repose sur le fait qu'on peut ajouter des booléens, et qu'on peut convertir un nombre en booléen à l'aide de `bool`.

```
def bissextile(annee) :
        return(bool((annee%4==0) - (annee%100==0) + (annee%400==0)))
```

Le tableau suivant résume les différents cas possible, et explique donc pourquoi cette fonction renvoie le bon résultat.

annee	annee%4==0	annee%100==0	annee%400==0	Somme des trois	bool(....)
Pas div. par 4	False	False	False	0	False
Div. par 4 pas par 100	True	False	False	1	True
Div. par 100 pas par 400	True	True	False	0	False
Div. par 400	True	True	True	1	True

□

Solution de l'exercice 14

1. Si x et y sont deux réels, alors le plus grand des deux est $\frac{1}{2}(x+y+|x-y|)$.
En effet, si ce maximum vaut x, alors $x-y \geqslant 0$ et donc $x+y+|x-y| = x+y+x-y = 2x$.
Et si le plus grand des deux vaut y, alors $x+y+|x-y| = x+y-x+y = 2y$.

```
def maximum(a,b) :
        return(1/2*(a+b+abs(a-b)))
```

2. Le minimum de deux nombres est la différence de leur somme et de leur maximum.

```
def minimum(a,b) :
        return(a+b-maximum(a,b))
```

3. On peut calculer le maximum des deux premiers nombres, le maximum des deux derniers, et le maximum de ces deux maxima.

```
def max4(a,b,c,d) :
        return(maximum(maximum(a,b),maximum(c,d)))
```

4. Nous savons déjà comment isoler le maximum, et comment isoler le minimum.
Le dernier nombre est donc la somme des trois, moins le plus grand, moins le plus petit.

```
def sort3(a,b,c) :
        M = maximum(maximum(a,b),c)
        m = minimum(minimum(a,b),c)
        milieu = a+b+c-m-M
        return m,milieu,M
```

Remarquons qu'on peut s'en tirer en deux (grandes) lignes, mais que c'est bien moins lisible[2] !

[2] Et donc pas forcément souhaitable.

```
def sort3(a,b,c) :
        return minimum(minimum(a,b),c),a+b+c - minimum(minimum(a,b),c) -
maximum(maximum(a,b),c),maximum(maximum(a,b),c)
```

□

Solution de l'exercice 15

Clarifions tout de suite un point : dans l'ensemble $[\![a,b]\!]$, il y a $b-a+1$ éléments (et non $b-a$).
Pour les personnes qui ne sont pas nées le 29 février, il n'y a pas grand chose à dire, il suffit de compter le nombre d'années écoulées[3] entre la naissance et le décès, et si la date d'anniversaire a été dépassée, de rajouter un.
En revanche, pour les personnes nées un 29 février, il faut trouver combien d'années bissextiles ont eu lieu entre l'année de naissance et l'année de décès.
À cet effet, nous proposons d'utiliser une fonction `nb_multiples` qui compte le nombre de multiples de `n` qu'il y a dans $[\![a,b-1]\!]$.

Détails
Si vous n'êtres pas convaincu, essayez avec $a=0$ et $b=5$, puis avec $a=1$.

[3] Année de naissance non comprise, puisqu'on ne fête traditionnellement pas son premier anniversaire l'année de sa naissance.

```
from math import ceil,floor
def nb_multiples(a,b,n) :
    return(floor((b-1)/n)-ceil(a/n)+1)
```

Notons que cette fonction fait bien ce qui avait été annoncé : `ceil(a/n)` est le premier multiple de n après a (éventuellement égal à a si ce dernier est divisible par n) ,`floor((b-1)/n)` est le multiple de n avant $b-1$.
Et donc il suffit de compter combien il y a d'entiers entre ces deux entiers, ce qui nécessite d'ajouter un à la différence en vertu de la remarque ci-dessus.
On en déduit alors facilement le nombre d'années bissextiles entre a et $b-1$.

```
def nb_bissextiles(a,b) :
    n = nb_multiples(a,b,4) - nb_multiples(a,b,100) + nb_multiples(a,b,400)
    return(n)
```

De là, il est facile de conclure, en prenant garde, pour les personnes nées un 29 février et décédées une année bissextile, de vérifier si le 29 février est déjà passé ou non.

```
def anniversaires(j_naiss,m_naiss,a_naiss,jour,mois,annee) :
    if (j_naiss != 29) or (m_naiss !=2) :
        if (m_naiss<mois) or ((m_naiss==mois) and (j_naiss<=jour)) :
            return(annee-a_naiss)
        else :
            return(annee-a_naiss-1)
    else :
        if bissextile(annee) and ((mois >2) or ((jour==29) and (mois == 2))) :
            return(nb_bissextiles(a_naiss,annee))
        else :
                        return(nb_bissextiles(a_naiss,annee)-1)
```

□

Solution de l'exercice 16

Il s'agit de se rappeler que le dernier chiffre (en base 10) d'un entier n est le reste de sa division euclidienne par 10. Et donc la simple commande `(2017**2019) %10` nous informe que le dernier chiffre de 2017^{2019} est un 3.
Pour déterminer le nombre de chiffres de 2017^{2019}, ayons en tête le fait qu'un nombre x possède n chiffres si et seulement si $10^{n-1} \leqslant x < 10^n$.
Ce qui donne encore $(n-1)\ln(10) \leqslant \ln(x) < n\ln(10) \Leftrightarrow n-1 \leqslant \log(x) < n$.
Et donc $n-1 = \lfloor \log(x) \rfloor \Leftrightarrow n = \lfloor \log x \rfloor + 1$.
Donc le nombre de chiffres nécessaires pour écrire 2017^{2019} est obtenu à l'aide de

```
floor(log10(2017**2019))+1
```

Il possède donc 6673 chiffres. □