Gymnase de Burier

1OS 2018 — Python

Table des matières

Chapitre 1

Introduction

1.1 Éléments primitifs

1.1.1 Démontrer que la somme des n premiers entiers vaut

$$\frac{n \cdot (n+1)}{2}$$

1.1.2 Faire calculer la somme des nombres entiers 8, 9 et 10.

1.1.3 Faire calculer le produit des nombres entiers 8, 9 et 10.

1.1.4 Faire calculer le nombre de secondes dans une année bancaire.

1.1.5 Faire calculer le nombre de pouces dans un mile. Sachant qu'un pouce vaut 2.54 cm, exprimer un mile en km.

1.1.6 Trouver le nombre de poignées de mains que doivent échanger toutes les personnes de votre classe de façon à ce que chaque personne serre la main de toute autre personne une et une seule fois.

1.1.7 Trouver l'âge moyen des élèves de la classe à l'aide d'une division entière.

1.1.8 Trouver l'âge moyen des élèves de la classe à l'aide d'une division à virgule flottante.

1.1.9 Faire calculer le volume d'une sphère de rayon 1 en utilisant la formule

$$V = 4/3\,\pi\,r^3$$

1.1.10 Multiplier 0.3 par 10. Diviser ensuite 0.3 par 0.1. Que dire des deux résultats ?

1.1.11 La galaxie d'Andromède se trouve à environ 2.55 millions d'années lumières du Soleil. Sachant qu'une année lumière compte environ 9 460.730 km, calculer la distance qui sépare Andromède du Soleil.

1.1.12 Combien d'années faudrait-il pour se déplacer jusque dans la galaxie d'Andromède à la vitesse constante de 125 km/h ?

1.1.13 Faire calculer la factorielle de 13. On rappelle que la factorielle de $n \in \mathbb{N}$ est définie par la formule

$$n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$

1.1.14 Faire calculer la 120ème puissance de 2.

1.1.15 Sachant que l'univers est vieux de 15 milliards d'années, calculer le nombre de secondes que cela représente.

1.1.16 Si tous les Zurichois se serrent la main, combien cela représente-t-il de poignées de mains ? Et si tous les suisses se serrent la main ?

1.2 Nommer des objets

1.2.1 Considérons les instructions Python ci-dessous :

```
a = 79
b = a
a = 89
```

a) Dessiner un diagramme de références montrant les liens entre étiquettes et objets juste après les deux premières instructions.

b) Faire de même pour montrer la situation après la troisième instruction.

1.2.2 Parmi les noms proposés ci-dessous, lesquels sont des noms de variable acceptés par le langage Python ?

a) `_abc123`

b) `123abc`

c) `abc123_`

d) `_123`

1.2.3 Considérons les instructions ci-dessous :

```
a = 10
b = 20
c = a * b
d = a + b
```

Dessiner un diagramme montrant tous les noms et objets après exécution de ces quatre instructions.

1.2.4 Quelles sont les valeurs de `a` et `b` après évaluation des quatre instructions ci-dessous ?

```
a = 10
b = 20
a = b
b = 15
```

1.2.5 Considérons les instructions ci-dessous :

```
nbre_entier = 0
nbre_entier = nbre_entier + 1
nbre_entier = nbre_entier + 2
```

Quelle est la valeur référencée par la variable `nbre_entier` après l'évaluation des trois lignes ci-dessus ?

1.3 L'abstraction

1.3.1 Créer une tortue qui s'appelle `pikachu`. Donner à `pikachu` l'ordre d'avancer de 100. Quelle est maintenant la position de `pikachu` ?

1.3.2 Créer une tortue qui s'appelle `miew`. Donner à `miew` l'ordre de tourner à droite de 45 degrés et celui d'avancer de 50. Observer qu'il y a maintenant deux tortues dans la même fenêtre.

1.3.3 Ecrire une fonction `dessineCarre` prenant en paramètres un objet de type `Turtle` et un nombre entier qui fait tracer un carré par la tortue passée en paramètre.

1.3.4 Modifier la fonction `dessineCarre` de sorte à ce qu'elle fasse dessiner à la tortue un rectangle dont la longueur est le double de la largeur. Renommer cette fonction `dessineRectangleParticulier`.

1.3.5 Créer une nouvelle fonction nommée `dessineRectangle` qui prend trois paramètres : une tortue, une longueur et une largeur et qui dessine le rectangle correspondant.

1.3.6 Créer un fichier polygones.py dans lequel on enregistrera le code des trois fonctions `dessineCarre`, `dessineRectangleParticulier` et `dessineRectangle`. Dans un autre fichier nommé dessin.py, importer le module `polygones` à l'aide de l'instruction `from polygones import *` et faire dessiner un carré, un rectangle dont la largeur mesure le double de la longueur et un rectangle quelconque.

1.3.7 Importer le module créé à l'exercice précédent dans un nouveau fichier nommé dessin_2.py, cette fois à l'aide de l'instruction `import polygones`. Faire dessiner trois polygones comme à l'exercice précédent.

1.4 Répétition

1.4.1 Créer une fonction `carre` qui permet de faire tracer un carré de côté donné, à une tortue donnée. On veillera à utiliser une boucle `for` dans le code.

1.4.2 En utilisant une boucle `for`, créer une fonction `triangle` qui permet de faire tracer un triangle équilatéral de côté donné, à une tortue donnée.

1.4.3 Faire dessiner 10 triangles ayant chacun un sommet placé en $(0, 0)$. Le côté du premier triangle mesure 20 unités. À chaque étape, la longueur du côté du triangle augmente de 30 unités.

1.5 Solutions des exercices

1.1.1 –

1.1.2

```
>>> 8 + 9 + 10
27
```

1.1.3

```
>>> 8 * 9 * 10
720
```

1.1.4

```
>>> 60 * 60 * 24 * 365
31536000
```

1.1.5

```
>>> 12 * 3 * 1760
63360
>>> 2.54 * 63360 / 100
1609.344
```

1.1.6

```
>>> 1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23
276
>>> 23 * 24 // 2
276
```

1.1.7 –

1.1.8 –

1.1.9

```
>>> 4 / 3 * 3.141592653589793238462
4.1887902047863905
```

1.1.10 –

1.1.11

```
>>> 9.46073e+6 * 2.55e+6
24124861500000.0
```

1.1.12

```
>>> 24124861500000.0 / 125.0 / 24 / 365.25
22016757.01574264
```

1.1.13

```
>>> 13*12*11*10*9*8*7*6*5*4*3*2*1
6227020800
```

1.1.14

```
>>> 2**120
1329227995784915872903807060280344576
```

1.1.15

```
>>> 15*10**9*365*24*60*60
473040000000000000
```

1.1.16

```
>>> 415682 * 415683 // 2
86395970403
>>> 8327126 * 8327127 // 2
34670517873501
```

1.2.1 –

1.2.2 Les noms proposés en a), c) et d).

1.2.3 –

1.2.4 La variable `a` se réfère à l'objet `20` et la variable `b` à l'objet `15`.

1.2.5 La variable `nbre_entier` est une référence à l'objet 3.

1.3.1 `(100.00,0.00)`

1.3.2

```
import turtle
pikachu = turtle.Turtle()
pikachu.forward(100)
miew = turtle.Turtle()
miew.right(45)
miew.forward(50)
```

1.3.3

```
import turtle

def dessineCarre(tortue, cote):
    tortue.forward(cote)
    tortue.right(90)
    tortue.forward(cote)
    tortue.right(90)
    tortue.forward(cote)
    tortue.right(90)
    tortue.forward(cote)
    tortue.right(90)

onyx = turtle.Turtle()
dessineCarre(onyx, 200)
```

1.3.4

```
import turtle

def dessineRectangleParticulier(tortue, largeur):
```

```
        tortue.forward(2 * largeur)
        tortue.right(90)
        tortue.forward(largeur)
        tortue.right(90)
        tortue.forward(2 * largeur)
        tortue.right(90)
        tortue.forward(largeur)
        tortue.right(90)

    onyx = turtle.Turtle()
    dessineRectangleParticulier(onyx, 100)
```

1.3.5

```
    def dessineRectangle(uneTortue, longueur, largeur):
        uneTortue.forward(longueur)
        uneTortue.right(90)
        uneTortue.forward(largeur)
        uneTortue.right(90)
        uneTortue.forward(longueur)
        uneTortue.right(90)
        uneTortue.forward(largeur)
        uneTortue.right(90)
```

1.3.6 Voir le fichier polygones_1.zip

1.3.7 Voir le fichier polygones_1.zip

1.4.1

1.4.2

1.4.3

Chapitre 2

πthon

2.1 Le module `math`

2.1.1 Faire exécuter la commande `help("turtle")`. Que se passe-t-il si l'on écrit, sans les guillemets, `help(turtle)` ?

2.1.2 Importer le module `turtle` et refaire l'exercice précédent.

2.1.3 Appliquer la commande `help` à la fonction `math.sin`.

2.1.4 Explorer les possibilités du module `random` en utilisant l'aide intégrée de Python. Que fait la fonction `randrange` ? Que fait la fonction `randint` ?

2.2 L'approche d'Archimède

2.2.1 Soit P_n le polygone à n côtés inscrit dans un cercle de rayon r. Notons α l'angle au centre de l'arc déterminé par l'un des côtés de P_n. Montrer que l'on a

$$\pi \simeq n \cdot \sin\big(\alpha/(2n)\big)$$

si n est assez grand.

2.2.2 En utilisant les fonctions `math.sin` et `math.radians`, programmer une fonction `archi_pi` qui prend un nombre entier en paramètre et qui renvoie une valeur approchée de π.

2.2.3 Quelle valeur de `n` faut-il choisir pour que `archi_pi(n)` renvoie la même valeur que `math.pi` ?

2.3 The accumulator pattern

2.3.1 Faire calculer la somme des 100 premiers nombre pairs à l'aide de « l'accumulator pattern ».

2.3.2 Faire calculer la somme des 50 premiers nombres impairs.

2.3.3 Faire calculer la moyenne des 100 premiers nombres impairs.

2.3.4 Ecrire une fonction qui renvoie la moyenne des n premiers entiers, n étant passé en paramètre.

2.3.5 Ecrire une fonction `ma_factorielle` qui calcule le produit des n premiers nombres entiers, n étant passé en paramètre.

2.3.6 Chaque nombre de la suite de Fibonacci est la somme des deux nombres précédents. Les deux premiers nombres de cette suite sont 1 et 1. Calculer le dixième nombre de Fibonacci.

2.3.7 Ecrire une fonction `fibonacci` prenant un paramètre entier $n \geq 3$ et qui renvoie le $n^{\text{ème}}$ nombre de Fibonacci.

2.3.8 Ecrire une fonction `leibniz` qui prend un paramètre entier n et qui renvoie une approximation de π basée sur la formule ci-dessous :

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \dots$$

2.3.9 Faire tourner la fonction `leibniz` avec 10 000 termes, puis 100 000 termes.

2.3.10 Faire afficher l'approximation de π donnée par `math.pi`. Peut-on obtenir en un temps raisonnable le même nombre de décimales avec notre fonction `leibniz` ?

2.3.11 Comparer le résultat donné par la fonction `leibniz` pour n termes avec celui donné par la fonction `archi_pi` pour n côtés.

2.3.12 Ecrire une fonction `wallis` qui prend un paramètre entier n et qui renvoie une approximation de π basée sur la formule ci-dessous :

$$\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \dots$$

2.3.13 Faire tourner la fonction `wallis` avec 10 000 termes, puis 100 000 termes.

2.3.14 Faire afficher l'approximation de π donnée par `math.pi`. Peut-on obtenir en un temps raisonnable le même nombre de décimales avec notre fonction `wallis` ?

2.3.15 Comparer le résultat donné par la fonction `wallis` pour n termes avec celui donné par la fonction `archi_pi` pour n côtés et celui donné par la fonction `leibniz` pour n termes.

2.4 Une simulation de type « Monte Carlo »

2.4.1 Quelle résultat donne l'évaluation de l'expression booléenne `False or True` ?

2.4.2 Quelle résultat donne l'évaluation de l'expression booléenne `True and False` ?

2.4.3 Quelle résultat donne l'évaluation de l'expression booléenne `not 7 > 3` ?

2.4.4 Soit l'expression booléenne `not (True or False)`. Quel résultat donne son évaluation par le moteur d'interprétation de Python ?

2.4.5 Soit l'expression booléenne `(not True) and (not False)`. Quel résultat donne son évaluation par le moteur d'interprétation de Python ?

2.4.6 Écrire une expression booléenne équivalente à `(not (True and False))`.

2.4.7 Écrire une expression booléenne composée qui renvoie `True` si la valeur de la variable `count` est comprise entre 1 non compris et 10 y compris.

2.4.8 Comment fait Python pour savoir à quelle instruction `if` appartient une instruction `else` subséquente ?

2.4.9 Écrire une instruction conditionnelle qui attribue la valeur 1 à une variable `answer` si une variable `resultat` vaut 100. La variable `answer` se verra affecter la valeur 2 si ce n'est pas le cas.

2.4.10 Écrire des instructions conditionnelles emboîtées qui attribuent la valeur 4 à la variable `bonus` si la valeur de la variable `score` est plus grande ou égale à 90, la valeur 3 si `score` est compris entre 80 et 89, la valeur 2 si `score` est entre 70 et 79, la valeur 1 si `score` est entre 60 et 69 et la valeur 0 sinon.

2.4.11 Si l'instruction `elif` n'a pas été utilisée dans le code écrit pour l'exercice précédent, réécrire ce code en utilisant cette instruction.

2.4.12 Une année est bissextile si elle est divisible par 4 sauf si c'est un siècle qui n'est pas divisible par 400. Écrire une fonction `annee_bissextile` qui prend une année en paramètre et qui renvoie `True` si l'année est bissextile et `False` sinon.

2.4.13 Un fournisseur de primeurs vend des oranges à 85 ct. le kg. Les frais de port s'élèvent à 15 fr. par commande. Pour une commande de plus de 50 kg, les frais de ports sont réduits à 3 fr. 50 ct. Écrire une fonction `calculer_prix` qui prend un nombre de kg en paramètre et qui renvoie le prix total de la commande.

2.4.14 Écrire une fonction qui prend trois entiers en paramètre et qui renvoie le plus grand des trois.

2.4.15 Écrire une fonction `paye_de_la_semaine` qui prend deux paramètres : le nombre d'heures et le salaire horaire ; cette fonction doit renvoyer la paye en fonction du nombre d'heures faites en une semaine comptant 42h30 de travail. Toute heure supplémentaire est rétribuée une fois et demie la valeur de l'heure « standard ».

2.4.16 Écrire une fonction `monteCarlo` qui prend en paramètre un nombre entier et qui renvoie une approximation de π. Cette fonction génère n couples de nombres réels appartenant à $[0;1] \times [0;1]$ et calcule l'approximation de π en comptant ceux qui se trouvent dans le quart de disque inclus dans le carré $[0;1] \times [0;1]$.

2.4.17 Écrire un programme qui fait se déplacer une tortue aléatoirement sur l'écran, dans un carré de 200 pixels de côté, dont le coin inférieur gauche est placé à l'origine. La tortue se déplace continuellement, tant que le programme n'a pas été interrompu par l'utilisateur.

2.4.18 Modifier le programme de l'exercice précédent de manière à ce que la tortue marque un point à l'écran à la fin de chaque déplacement. Le point sera de couleur bleue si l'emplacement est dans le disque de rayon 200 pixels centré à l'origine et de couleur rouge sinon.

2.4.19 Faire en sorte que le programme de l'exercice précédent garde un décompte du nombre total de points marqués par la tortue et également du nombre de points bleus. En déduire une approximation de π.

2.4.20 Écrire une fonction `monteCarlo` qui prend en paramètres une tortue, le côté d'un carré exprimé en pixels, un nombre de points à marquer au total, et qui renvoie une approximation de π.

2.4.21 Faire tourner la fonction `monteCarlo`, avec ou sans tortue, en augmentant le nombre de points. Tester les fonctions pour 10 000 points, 100 000 points et plus si possible.

2.4.22 Comparer l'approximation de π donnée par la fonction `monteCarlo` avec celles données par les fonctions `wallis` ou `leibniz`.

2.4.23 Écrire une fonction `estDansDisque` qui prend un point du plan et un rayon en paramètres. La fonction renvoie `True` si le point se trouve dans le disque et `False` sinon.

2.4.24 Modifier la fonction `monteCarlo` de sorte à ce qu'elle utilise la fonction auxiliaire `estDansDisque`.

2.5 Exercices de programmation

2.5.1 Écrire une fonction qui permet de calculer la circonférence d'un cercle de rayon r. Passer r en paramètre. Utiliser la valeur de π du module `math`.

2.5.2 Écrire une fonction qui permet de calculer l'aire d'un cercle de rayon r. Passer r en paramètre. Utiliser la valeur de π du module `math`.

2.5.3 Écrire une fonction qui permet de calculer le volume d'une sphère de rayon r. Passer r en paramètre. Utiliser la valeur de π du module `math`.

2.5.4 Écrire une fonction qui renvoie une valeur approchée de π en évaluant l'expression

$$\pi = 16 \cdot \arctan(1/5) - 4 \cdot \arctan(1/239)$$

dans laquelle arctan désigne la fonction arc tangente du module mathématique de python.

2.5.5 Écrire une fonction qui renvoie une valeur approchée de π en évaluant l'expression

$$\pi = \frac{\ln(640320^3 + 744)}{\sqrt{163}}$$

dans laquelle ln désigne la fonction `log` du module mathématique de python.

2.5.6 Ecrire une fonction `estPair` qui prend un nombre en paramètre et renvoie `True` si le nombre est paire et `False` si ce n'est pas le cas.

2.5.7 Écrire une fonction qui renvoie une valeur approchée de π en évaluant l'expression

$$\pi = \sqrt{12}\left(1 - \frac{1}{3 \cdot 3} + \frac{1}{5 \cdot 3^2} - \frac{1}{7 \cdot 3^3} + \dots\right)$$

Utiliser un paramètre nommé n pour désigner le nombre de termes à inclure dans la somme.

2.5.8 Nous avons utilisé la fonction racine carrée du module mathématique de python. Ecrire votre propre fonction qui calcule une approximation de la racine carrée d'un nombre n en utilisant la suite donnée par récurrence ci-dessous :

$$X_{k+1} = \frac{1}{2} \cdot \left(X_k + \frac{n}{X_k}\right)$$

où $X_0 = 1$. Cette définition exprime le fait que le nombre X_k s'approche de plus en plus de $\sqrt{n}$ lorsque k devient très grand. La fonction prendra deux nombres en paramètre, le nombre n dont il s'agit d'extraire la racine carrée et le nombre $k \in \mathbb{N}$ d'étapes à réaliser.

2.5.9 Le problème de l'aiguille de Buffon est un problème célèbre du dix-huitème siècle. Il s'agit d'une expérience qui permet de trouver une approximation de π. Trouver l'énoncé de ce problème et créer une simulation à l'aide du module turtle de python. Cela donne-t-il une bonne approximation ?

2.6 Solutions des exercices

2.1.1 –

2.1.2

2.1.3

2.1.4

2.2.1

2.2.2

2.2.3

2.3.1

2.3.2

2.3.3

2.3.4

2.3.5

2.3.6

2.3.7

2.3.8

2.3.9

2.3.10

2.3.11

2.3.12

2.3.13

2.3.14

2.3.15

2.4.1

2.4.2

2.4.3

2.4.4

2.4.5

2.4.6

2.4.7

2.4.8

2.4.9

2.4.10

2.4.11

2.4.12

2.4.13

2.4.14

2.4.15

2.4.16

2.4.17

2.4.18

2.4.19

2.4.20

2.4.21

2.4.22

2.4.23

2.4.24

2.5.1

2.5.2

2.5.3

2.5.4

2.5.5

2.5.6

2.5.7

2.5.8

2.5.9

Chapitre 3

Codes et autres secrets

3.1 Opérations sur les chaînes de caractères

3.1.1 Créer une chaîne de caractères qui contiendra votre prénom et votre nom, séparés par un espace.

3.1.2 En utilisant l'opérateur d'indiçage (« slice operator » en anglais), faire afficher à la console votre prénom.

3.1.3 En utilisant l'opérateur d'indiçage, faire afficher à la console votre nom.

3.1.4 En utilisant l'opérateur « slice » et la concaténation, faire afficher à la console votre nom complet sous la forme suivante :

```
Nom, Prenom
```

3.1.5 Faire afficher la longueur de votre prénom.

3.1.6 On déclare deux variables comme ci-dessous :

```
s = "s"
p = "p"
```

En utilisant la concaténation et la répétition, soit les opérateurs `+` et `*`, écrire une expression dont l'évaluation donne la chaîne de caractères `mississippi`.

3.1.7 On déclare une chaîne de caractères comme ci-dessous :

```
mot = "SUPERCALIFRAGILISTIC"
```

En utilisant l'opérateur « slice » et une boucle `for`, faire afficher à la console les lignes ci-dessous :

```
S
SU
SUP
SUPE
SUPER
SUPERC
SUPERCA
SUPERCAL
SUPERCALI
SUPERCALIF
SUPERCALIFR
SUPERCALIFRA
SUPERCALIFRAG
SUPERCALIFRAGI
SUPERCALIFRAGIL
SUPERCALIFRAGILI
SUPERCALIFRAGILIS
SUPERCALIFRAGILIST
SUPERCALIFRAGILISTI
SUPERCALIFRAGILISTIC
```

3.1.8 En utilisant la méthode `count`, trouver le nombre de `"s"` qu'il y a dans le mot `"mississippi"`.

3.1.9 En utilisant la méthode `count`, trouver le nombre de fois que la lettre `"é"` apparaît dans le mot `hétérogénéité`.

3.1.10 En utilisant la méthode `count`, trouver le nombre de fois que la lettre `"e"`, accentuée ou non, apparaît dans le texte ci-dessous :

```
Un chemin creux s'enfonçait. Tout disparut. L'homme avait
 à droite une palissade, quelque mur de grosses planches
fermant une voie ferrée ; tandis qu'un talus d'herbe s'él
evait à gauche, surmonté de pignons confus, d'une vision
de village aux toitures basses et uniformes. Il fit envir
on deux  cents pas. Brusquement, à un coude du chemin, le
s feux reparurent près de lui, sans qu'il comprît davanta
ge comment ils brûlaient si haut dans le ciel mort, parei
ls à des lunes fumeuses. Mais, au ras du sol, un autre sp
ectacle venait de l'arrêter. C'était une masse lourde, un
 tas écrasé de constructions, d'où se dressait la silhoue
tte d'une cheminée d'usine ; de rares lueurs sortaient de
s fenêtres encrassées, cinq ou six lanternes tristes étai
ent pendues dehors, à des charpentes dont les bois noirci
s alignaient vaguement des profils de tréteaux gigantesqu
es ; et, de cette apparition fantastique, noyée de nuit e
t de fumée, une seule voix montait, la respiration grosse
 et longue d'un échappement de vapeur, qu'on ne voyait po
int.
```

3.1.11 Dans la chaîne `mississippi`, remplacer toutes les occurences de `"ss"` par `"ox"`.

3.1.12 Trouver l'indice de la première occurence de `"p"` dans `mississippi`.

3.1.13 Faire afficher le mot `"python"` en lettres majuscules et centré dans une chaîne de caractères de longueur 20.

3.1.14 À l'aide d'une boucle, des méthodes `center`, `ljust`, `rjust` et de l'opérateur de concaténation, faire afficher les dix premières lignes du livret 13 comme ci-dessous :

```
 1 * 13 = 13
 2 * 13 = 26
 3 * 13 = 39
 4 * 13 = 52
 5 * 13 = 65
 6 * 13 = 78
 7 * 13 = 91
 8 * 13 = 104
 9 * 13 = 117
10 * 13 = 130
```

3.1.15 Quelle est la différence entre `ord('A')` et `ord('a')`.

3.1.16 Quelle est la différence entre `ord('1')` et `int('1')`.

3.1.17 On donne le code ci-dessous

```
def lettreVersNumero(car):
    alph = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    i = alph.find(car)
    if i < 0:
        print("Le caractère {} n'est pas
               une majuscule de l'alphabet.".format(car))
    return i


def numeroVersLettre(i):
    alph = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    if i > 25:
        print("Erreur: {} est trop grand.".format(i))
        lettre = ""
    elif i < 0:
        print("Erreur: {} est inférieur à 0.".format(i))
        lettre = ""
    else:
        lettre = alph[i]
    return lettre
```

Tester ce code et l'analyser ligne par ligne.

3.1.18 Quelles sont les fonctions prédéfinies qui permettent d'éviter d'écrire les fonctions de l'exercice précédent ?

3.1.19 Ecrire une fonction python `transpo2(message)` qui répond au cahier des charges suivant :

a) Tous les caractères d'indice pair dans la chaine de caractères `message` sont « collés » ensemble pour former une nouvelle chaîne de caractères nommée `car_pairs`.

b) Tous les caractères d'indice impair dans la chaine de caractères `message` sont « collés » ensemble pour former une nouvelle chaîne de caractères nommée `car_impairs`.

c) La fonction renvoie la chaîne formée des deux chaînes définies précédemment et mises bout-à-bout.

Exemple « d'exécution » :

```
transpo2("ILYACEUXQUIONTUNPISTOLETCHARGE")

car_pairs :   IYCUQINUPSOECAG
car_impairs : LAEXUOTNITLTHRE

IYCUQINUPSOECAGLAEXUOTNITLTHRE
```

3.1.20 Ecrire une fonction qui permet de déchiffrer un message qui aura été chiffré à l'aide de la fonction `transpo2` de l'exercice précédent.

Chapitre 4

Un peu de cryptologie

4.1 Calculer modulo un nombre entier

4.1.1 On calcule *modulo 5*, c'est à dire avec les entiers compris entre 0 et 4. L'ensemble de ces nombres peut être noté

$$\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}.$$

Établir la table d'addition de $\mathbb{Z}_5$.

4.1.2 On calcule *modulo 3*, c'est à dire avec les entiers compris entre 0 et 2. L'ensemble de ces nombres peut être noté

$$\mathbb{Z}_3 = \{0, 1, 2\}.$$

Établir la table de multiplication de $\mathbb{Z}_3$.

4.1.3 Donner tous les nombres qui sont dans l'ensemble $\mathbb{Z}_{26}$. Faire calculer la table de multiplication de $\mathbb{Z}_{26}$. Dans un tableur, mettre en évidence les résultats 0 et 1.

Écrire l'ensemble des inversibles de $\mathbb{Z}_{26}$, noté $\mathbb{Z}_{26}^*$.

4.1.4 Effectuer les calculs ci-dessous et réduire le résultat modulo 26.

a) $12 + 3$ e) $13 + 13$ i) $24 + 3$ m) $12 - 15$

b) $7 + 15$ f) $5 + 20$ j) $21 + 7$ n) $20 - 25$

c) $11 + 11$ g) $21 + 22$ k) $8 + 19$ o) $8 - 19$

d) $15 + 8$ h) $17 + 16$ l) $5 + 25$ p) $5 - 25$

4.1.5 Effectuer les calculs ci-dessous et réduire le résultat modulo 26.

a) $27 + 33$ d) $72 + 100$

b) $41 + 28$ e) $132 - 150$

c) $35 + 36$ f) $53 - 12$

g) $30 - 8$

h) $-5 - 12$

4.1.6 Pour chaque calcul donné à l'exercice 4.1.5, réduire les nombres modulo 26 avant d'effectuer l'opération. On veillera à réduire le résultat modulo 26 également, après avoir fait le calcul. Cela modifie-t-il le résultat final ?

4.1.7 Effectuer les calculs ci-dessous et réduire le résultat modulo 26.

a) $6 \cdot 3$

b) $5 \cdot 4$

c) $2 \cdot 12$

d) $8 \cdot 4$

e) $5 \cdot 6$

f) $10 \cdot 4$

g) $15 \cdot 3$

h) $5 \cdot 7$

i) $12 \cdot 3$

j) $7 \cdot 6$

k) $8 \cdot 9$

l) $20 \cdot 30$

m) $14 \cdot 5$

n) $25 \cdot 2$

o) $4 \cdot 11$

p) $6 \cdot 6$

4.1.8 Effectuer les calculs ci-dessous et réduire le résultat modulo 26.

a) $27 \cdot 33$

b) $41 \cdot 28$

c) $35 \cdot 36$

d) $72 \cdot 100$

e) $132 \cdot 150$

f) $53 \cdot 12$

g) $30 \cdot 8$

h) $(-5) \cdot 12$

4.1.9 Pour chaque calcul donné à l'exercice 4.1.8, réduire les nombres modulo 26 avant d'effectuer l'opération. On veillera à réduire le résultat modulo 26 également, après avoir fait le calcul. Cela modifie-t-il le résultat final ?

4.1.10 Soit m et n deux nombres entiers, avec n positif.
On sait qu'il existe q et r, avec $0 \leq r < b$ tels que

$$m = n \cdot q + r$$

et que cette écriture est unique.

On dit que r est le reste de la division de m par n et on note

$$r = m \mod n$$

Soit n un entier positif, comme ci-dessus et a et b deux entiers quelconques.

a) Démontrer que

$$(a + b) \mod n = ((a \mod n) + (b \mod n)) \mod n$$

b) Démontrer que

$$(a \cdot b) \mod n = ((a \mod n) \cdot (b \mod n)) \mod n$$

4.1.11 Soit a et b dans $\mathbb{Z}$. On dit que b divise a, noté $b \mid a$, s'il existe k dans $\mathbb{Z}$ tel que $a = k \cdot b$.

Soit encore n dans $\mathbb{N}$. On dit encore que a est congru à b modulo n, noté $a \equiv b \mod n$, si $n \mid (b - a)$.

a) Démontrer que $a \equiv b \mod n$ si est seulement si a et b ont même reste après division par n.

b) Soit c dans $\mathbb{Z}$. Montrer que si $a \equiv b \mod n$ et que $b \equiv c \mod n$, alors $a \equiv c \mod n$.

4.2 Le chiffre de César

4.2.1 On numérote les lettres majuscules de l'alphabet à partir de 0. Traduire le message

`ILYACEUXQUIONTUNPISTOLETCHARGE`

à l'aide d'une liste de nombres.

4.2.2 On donne la liste de nombres

$$M = \{4, 19, 2, 4, 20, 23, 16, 20, 8, 2, 17, 4, 20, 18, 4, 13, 19\}$$

Trouver le message correspondant à M.

4.2.3 Chiffrer le message M de l'exercice 4.2.2 à l'aide du code de César. Donner le chiffre sous forme d'une liste de nombres.

4.2.4 Chiffer le message

`ILYACEUXQUIONTUNPISTOLETCHARGE`

en utilisant le code de César.

4.2.5 J'ai chiffré un message avec le code de César et le chiffre résultant est

`HWFHXATXLFUHXVHQW`

Quel était le message secret ?

4.2.6 Chiffrer le message donné ci-dessous à l'aide du décalage 7 :

$$M = \{19, 14, 8, 19, 20\}$$

Donner le résultat sous forme d'une liste de nombres.

Plutôt que dire que le décalage vaut 7, on dira que la *clef de chiffrement* vaut 7.

4.2.7 Trouver une technique pour déchiffrer le message

$$C = \{9, 24, 11, 1, 25, 11, 25\}$$

sachant qu'il a été codé à l'aide d'un décalage de 7.

4.2.8 Combien y a-t-il de décalages de « type César » possibles ?

4.2.9 Donner une technique générale pour déchiffrer un message codé à l'aide d'un décalage de César quelconque. Si l'on note $n \in \mathbb{Z}_{26}$ le décalage utilisé, donner une formule qui donne le décalage permettant de déchiffrer le message ou *clef de déchiffrement*.

4.2.10 J'ai chiffré un message avec le décalage 13. Trouver la clef de déchiffrement.

4.2.11 On a pu intercepter le chiffre ci-dessous :

`ARZTLVZCCZTVSIZEUVSILPVIV`

On sait que l'expéditeur a chiffré le message clair à l'aide du code de César, mais on ignore la clef. Comment faire pour trouver le message clair ? Cela revient à *casser le code de César*.

4.2.12 Chiffrer le message m ci-dessous à l'aide du chiffrement de César et la clef 24.

`JAMAISPERSONNENAVAITETEAUSSITROPFORT`

Donner toutes les étapes du chiffrement :

a) écrire M, la liste des nombres qui correspondent aux lettres du message ;

b) chiffrer M, c'est à dire donner la liste C des nombres calculés à l'aide de la formule

$$z' = (z + 24) \mod 26;$$

c) écrire c le chiffre formé de la suite des lettres qui correspondent aux nombres de C.

4.2.13 On donne ci-dessous le chiffre c obtenu à l'aide du chiffrement de César et la clef 4 :

`PEXSTMWWMXYHIIXPEWIHYGXMZMXI`

Donner toutes les étapes du déchiffrement :

a) écrire C, la liste des nombres qui correspondent aux lettres du message ;

b) déchiffrer C, c'est à dire donner la liste M des nombres calculés à l'aide de la formule

$$z = (z' + (26 - 4)) \mod 26;$$

c) écrire m le message formé de la suite des lettres qui correspondent aux nombres de M.

4.2.14 On a pu intercepter le chiffre ci-dessous :

`SNBDRBUJVJDEJRBNQNAKNKAJENBPNWBKAJENBPNWB`

Trouver le texte clair.

4.3 Le chiffrement affine

4.3.1 Soit z un nombre de $\mathbb{Z}_{26}$. On a vu que z peut représenter une lettre de l'alphabet majuscule. On sait que pour le chiffrement de type César on choisit une clef n et que le chiffrement de z est calculé à l'aide de la formule

$$z' = z + n \mod 26$$

On peut définir une autre méthode de chiffrement, nommée *chiffrement affine* en suivant la procédure ci-dessous :

a) On choisit a et b deux nombres de Z_{26} ;

b) On chiffre la lettre de la façon suivante :

$$z' = (a \cdot z + b) \mod 26$$

Cette façon de procéder soulève deux questions :

a) Peut-on choisir a « sans réfléchir » ?

b) Qu'en est-il pour le choix de b ?

4.3.2 Chiffrer le message ci-dessous à l'aide du chiffrement affine avec $a = 5$ et $b = 3$.

```
JAMAISPERSONNENAVAITETEAUSSITROPFORT
```

4.3.3 On a chiffré un message à l'aide du chiffrement affine avec $a = 15$ et $b = 7$. Le chiffre est donné ci-dessous :

```
QHGJYXRRXGVAPPGQHRPAVLGXKXGP
```

Déchiffrer ce message.

4.3.4 On a chiffré $z \in \mathbb{Z}_{26}$ à l'aide de la formule

$$z' = a \cdot z + b$$

pour a et b choisis dans Z_{26}.

Trouver une formule permettant de retrouver z à partir de z', a, et b.

4.4 Puissances modulo un nombre entier

4.4.1 Effectuer les calculs ci-dessous et réduire le résultat modulo 7.

a) 5^6 b) 2^6 c) 4^6 d) 5^6

4.4.2 Effectuer les calculs ci-dessous et réduire le résultat modulo 17.

a) 8^{16} b) 2^{16} c) 11^{16} d) 5^{16}

4.4.3 Un nombre supérieur à 1 est dit premier s'il a exactement deux diviseurs : 1 et lui-même. Trouver tous les nombres premiers inférieurs à 100.

4.4.4 Trouver un facteur de 4841.

4.4.5 Montrer que 521 est un nombre premier.

4.4.6 Factoriser les nombres ci-dessous.

a) 200 e) 35 i) 713 m) 4897

b) 150 f) 77 j) 1147 n) 6319

c) 128 g) 143 k) 473 o) 1591

d) 164 h) 323 l) 493 p) 901

4.4.7 Effectuer les calculs ci-dessous :

a) $3^5 \mod 5$ d) $5^7 \mod 7$ g) $11^{13} \mod 13$ j) $8^{19} \mod 19$

b) $2^5 \mod 5$ e) $6^7 \mod 7$ h) $10^{19} \mod 19$ k) $12^{17} \mod 17$

c) $3^7 \mod 7$ f) $5^{13} \mod 13$ i) $2^{19} \mod 19$ l) $10^{17} \mod 17$

Que peut-on en déduire ?

4.4.8 Établir la table de multiplication de $\mathbb{Z}_{15}$.

4.4.9 Trouver l'inverse de 13 dans $\mathbb{Z}_{15}$.

4.4.10 Donner la liste de tous les nombres inversibles de $\mathbb{Z}_{15}$.

4.4.11 Expliquer pourquoi si a est inversible dans $\mathbb{Z}_{15}$, alors

$$a^8 \equiv 1 \mod 15$$

4.4.12 Établir la table de multiplication de $\mathbb{Z}_{21}$.

4.4.13 Trouver l'inverse de 13 dans $\mathbb{Z}_{21}$.

4.4.14 Donner la liste de tous les nombres inversibles de $\mathbb{Z}_{21}$.

4.4.15 Expliquer pourquoi si a est inversible dans $\mathbb{Z}_{21}$, alors

$$a^{12} \equiv 1 \mod 21$$

4.4.16 Effectuer les calculs ci-dessous :

a) $25^5 \mod 133$

b) $100^{65} \mod 133$

c) $107^5 \mod 133$

d) $46^{65} \mod 133$

e) $234^{65} \mod 667$

f) $447^{65} \mod 667$

g) $99^{417} \mod 667$

h) $664^{65} \mod 667$

4.4.17 Soit $p = 23$ et $q = 31$. On donne encore $e = 17$.

a) Calculer d, l'inverse de e modulo $(p-1)\cdot(q-1) = 22 \cdot 30 = 660$.

b) Ecrire la clef privée correspondante.

c) Ecrire la clef publique correspondante.

d) Chiffrer le « message » $m = 333$ en utilisant la formule

$$c = m^e \mod n$$

si $n = p \cdot q$.

e) Retrouver le message à partir du chiffre c en utilisant la formule

$$m = c^d \mod n$$

f) Chiffrer $m = 555$.

g) Déchiffrer $c = 100$.

4.4.18

4.4.19

4.4.20

4.4.21

4.4.22

4.4.23

4.5 Solutions des exercices

3.1.1 –

3.1.2 –

3.1.3 –

3.1.4 –

3.1.5 –

3.1.6 –

```
mot = "SUPERCALIFRAGILISTIC"
for i in range(len(mot)):
    print(mot[0:i + 1])
```

3.1.7 3.1.8 –

3.1.9 –

3.1.10 –

3.1.11 –

3.1.12 –

3.1.13 –

3.1.14 –

3.1.15 –

3.1.16 –

3.1.17 –

3.1.18 –

3.1.19

```
def transpo2(message):
    car_pairs = ""
    car_impairs = ""
    i = 1
    for lettre in message:
        if i % 2 == 0:
            car_pairs += lettre
        else:
            car_impairs += lettre
        i += 1
    return car_pairs + car_impairs
```

3.1.20

```
def dech_transpo2(chiffre):
    taille = len(chiffre)
    moitie = taille // 2
    message = ""
    for i in range(moitie):
        message += chiffre[i + moitie] + chiffre[i]
    if taille % 2 == 1:
        message += chiffre[-1]
    return message
```

4.1.1 —

4.1.2 —

4.1.3 $\mathbb{Z}_{26} = \{0, 1, 2, 3, 4, 5, \ldots, 20, 21, 22, 23, 24, 25\}$

4.1.4 —

4.1.5 —

4.1.6 —

4.1.7 —

4.1.8 —

4.1.9 —

4.1.10 —

4.1.11 —

4.2.1 —

4.2.2 —

4.2.3 —

4.2.4 LOBDFHXATXLRQWXQSLVWROHWFKDUJH

4.2.5 ETCEUXQUICREUSENT

4.2.6 —

4.2.7 —

4.2.8 Il y a 25 décalages si l'on excepte le « décalage 0 ».

4.2.9 $26 - n$

4.2.10 —

4.2.11 —

4.2.12 —

4.2.13 —

4.2.14 —

4.3.1 —

4.3.2 —

4.3.3 —

4.3.4 —

4.4.1 —

4.4.2 —

4.4.3 —

4.4.4 $4841 = 103 \cdot 47$

4.4.5 —

4.4.6 —

4.4.7 —

4.4.8 —

4.4.9 —

4.4.10 —

4.4.11 —

4.4.12 —

4.4.13 —

4.4.14 —

4.4.15 —

4.4.16 —

4.4.17 —

4.4.18 —

4.4.19 —

4.4.20 —

4.4.21 —

4.4.22 —

4.4.23 —