Exercices d'Algorithmique et de Programmation avec Python

Guillaume Le Blanc Jean-Pierre Vallon

31 août 2018

Résumé

Ce cahier est destiné aux élèves de Seconde.

1. Ce cahier est à utiliser en mathématiques. Les entrées de l'index sont celles du programme de mathématiques de seconde. Cependant Il est conseillé d'utiliser ce cahier à partir du moment où les quatre premiers TP ont été effectués **afin de laisser aux élèves le temps de se familiariser avec le pseudo-code et le langage Python**
2. Les entrées de l'index concernent les mathématiques, mais les titres de chapitres concernent les concepts informatiques mis en jeu. L'ordre de difficulté des notions suit celui des TP, c'est à dire, séquence, répétition, fonction, variable, test, boucle tant que et finalement expressions logiques

Table des matières

1 Séquence **3**

2 Répéter n fois **6**

3 Notion de variable **8**

4 Fonction et Variable **12**

5 Test **14**

6 Boucle Tant Que **17**

7 Expressions logiques **22**

Chapitre 1

Séquence

Ex 1
Que fait ce programme ?

```
left(90)
left(90)
left(90)
left(90)
```

Ex 2
Que trace ce programme ?

```
left(-72)
forward(72)
left(-72)
forward(72)
left(-72)
forward(72)
left(-72)
forward(72)
left(-72)
forward(72)
```

Ex 3 Une suite de nombres En partant du nombre entier 7 on construit une suite de nombres de la manière suivante :

1. Multiplier le nombre avec le nombre qui le précède
2. Rajouter 1 au résultat

Utiliser l'interpréteur Python comme une calculatrice pour faire les premiers calculs, à partir du nombre 3263443 utiliser le copier-coller pour éviter de faire une erreur dans l'entrée des nombres dans l'interpréteur

1. On constate que le chiffre des unités est soit 7 soit 3. Prouver que tout entier se terminant par 3
2. Conjecturer une loi pour le nombre des chiffres des nombres de cette suite
3. On constate que la somme des chiffres pour chaque nombre vaut toujours 7
   Par exemple pour 43 on a évidemment 7, pour 1807 on a $1+8+0+7=16$ et $1+6=7$

```
>>> 7*6+1
43

>>> 43*42+1
1807

>>> 1807*1806+1
3263443

>>> 3263443*3263442+1
10650056950807

>>> 10650056950807*10650056950806+1
113423713055421844361000443

>>> 2739245030860303142341023429167468628119436436758091462794736794160869202622699363433211840458
243863492954873728399236975848797430631773058075388342946034495641007703476133047601673945464982838555415002139208067+1
750346333909286311464218348364293017384724140073732363176684391768374238237200233203724274839819736227493060107386942069521875902258281351952761393460726027774387698896086030486687796275661995019983548441838410309689949952466600707329879785293212787692398334049744823196004883309419542523184647878503560233926114995356472937133791777338667013341358153749078802023126509321031022439709564437114889326128420161145361044367+1
```

On vérifie cette propriété avec Python sur quelques exemples en utilisant la commande % 9 qui calcule le reste de la division 9 (On démontre en mathématiques que calculer le reste de la division par 9 équivaut à faire la somme des chiffres jusqu'à obtenir un nombre inférieur ou égal à 9

```
>>> 750346333909286311464218348364293017384724140073732363176684391768374238237200233203724274839819736227493060107386942069521875902258281351952761393460726027774387698896086030486687796275661995019983548441838410309689949952466600707329879785293212787692398334049744823196004883309419542523184647878503560233926114995356472937133791777338667013341358153749078802023126509321031022439709564437114889326128420161145361044367 % 9
7

>>> 43 % 9
7

>>> 1807 % 9
7

>>> 3263443 % 9
7
```

Ex 4 Exécuter à la console l'opération suivante

```
>>> 0.1*3
0.30000000000000004
```

Etonnant non ? on s'attend à obtenir 0.3. Il faut avoir conscience que nous programmons des machines et donc **la représentation des nombres à virgule conduit parfois à des valeurs approchées**

Pour avoir une idée de l'approximation on va utiliser la fonction valeur absolue `abs()` (voir sur internet la fonction valeur absolue) La différence est "très petite" de

```
>>> abs(0.1*3-0.3)
5.551115123125783e-17
```

l'ordre de 10^{-17}, on verra dans les chapitres suivants comment tenir compte de cette approximation dans certains programmes

Ex 5

Importer le module math dans l'interpréteur en entrant `from math import *`

On a vu en maths que la racine carrée de 2, le nombre noté $\sqrt{2}$ vérifie $\sqrt{2}^2 = 2$.

Là encore avec Python on a une valeur approchée de $\sqrt{2}$.

Donner un ordre de grandeur de l'écart entre $\sqrt{2}^2$ et 2. (`sqrt` signifie square root c'est à dire racine carrée en anglais)

```
>>> from math import *

>>> sqrt(2)
1.4142135623730951

>>> sqrt(2)**2
2.0000000000000004
```

Ex 6

1. Compléter le programme suivant pour qu'il trace un parallélogramme

```
forward(100)
left(60)
forward(70)
# à compléter
```

2. Faire évoluer le programme précédent pour qu'il trace cette fois-ci un losange de 100 pixels de côté et tel que l'un des angles vaut 70 degrés
3. Ecrire un programme pour qu'il trace cette fois-ci un rectangle de longueur 100 pixels et de largeur 70 pixels

Chapitre 2

Répéter n fois

Ex 1 Corriger ce programme, afin qu'il trace un carré

```
for i in range(4):
    forward(50)
left(90)
```

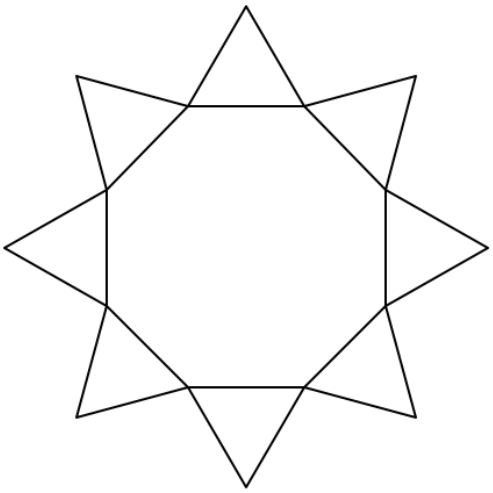
Que trace-t-il d'ailleurs ?

Ex 2

1. Pour trouver de quel angle x en degrés, faut il tourner avec la tortue pour dessiner un pentagone régulier il faut résoudre $5x = 360$. Que vaut x ? Est ce un **entier** ?
2. Quelle équation résoudre pour un heptagone régulier (7 côtés) ? La solution trouvée est elle entière ?Est elle rationnelle ?Qu'observez vous de remarquable dans l'écriture décimale de la solution ?

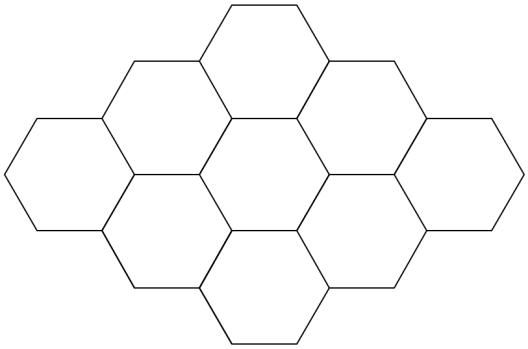
Ex 3

Faire un algorithme puis un programme python pour reproduire le dessin suivant :



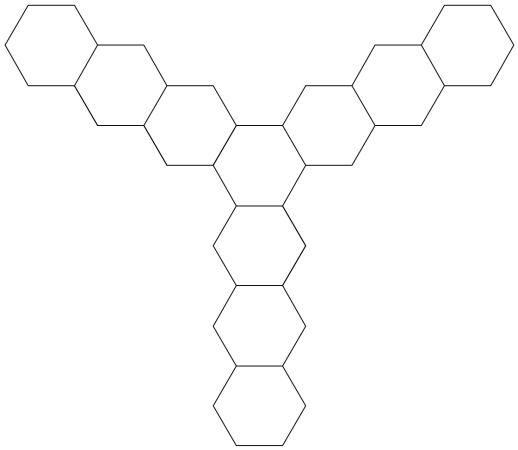
Ex 4

Faire un algorithme puis un programme python pour reproduire le dessin suivant :



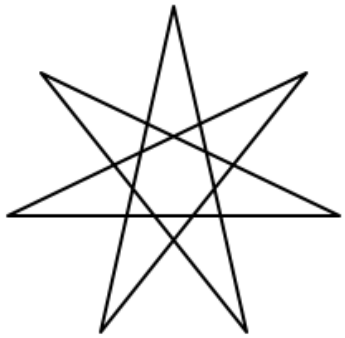
Ex 5

Faire un algorithme puis un programme python pour reproduire le dessin suivant :



Ex 6

Faire un algorithme puis un programme python pour reproduire un dessin analogue au suivant :



Chapitre 3

Notion de variable

Ex 1
Compléter et corriger le programme suivant :

1. x est la variable contenant la longueur d'un côté du polygone régulier. Renommer cette variable avec un nom **explicite**
2. Modifier l'instruction x = 50 de telle sorte que l'utilisateur puisse entrer une valeur **décimale** au clavier

```
nbCotes = int(input("Combien de côtés a le polygone régulier?"))
x  = 50
for indice in range(......):
    forward(x)
    left(.....)
```

Ex 2 (*Formules d'aire*) Compléter pour que l'algorithme calcule :

1. l'aire d'un carré de côté donné par l'utilisateur
2. l'aire d'un cercle de rayon donné par l'utilisateur. Renommer la variable côté

---

début
  coté ← entrer("Quel est la longueur du côté ?")
  aire ← .....
fin

Ex 3 (*Formules de volume*)
Compléter pour que l'algorithme calcule (renommer les variables) :

1. le volume d'un cylindre
2. le volume d'un cône

---

début
  r ← entrer("Que vaut le rayon ?")
  h ← .....
  v ← .....
fin

Ex 4

1. Faire évoluer "à la main" l'algorithme suivant à l'aide d'un tableau d'évolution de variables :

```
début
    entier ← 2
    entier ← entier + 1
    entier ← entier*entier
fin
```

   A la fin de l'algorithme qu'elle est la valeur de la variable `entier` ?

2. Faire évoluer "à la main" l'algorithme suivant à l'aide d'un tableau d'évolution de variables :

```
début
    entier ← 2
    entier ← entier*entier
    entier ← entier + 1
fin
```

   A la fin de l'algorithme qu'elle est la valeur de la variable `entier` ?

Ex 5
On veut **échanger** les valeurs des variables `entier1` et `entier2`.
Est ce que l'algorithme suivant convient ?

```
début
    entier1 ← entier2
    entier2 ← entier1
fin
```

Ex 6
Quel est le résultat des 3 instructions suivantes ?

```
début
    entier1 ← entier 1 + entier2
    entier2 ← entier1 − entier 2
    entier1 ← entier 1 − entier2
fin
```

Ex 7 Au cours de ce trimestre vous avez trois notes en mathématiques

1. Donner un algorithme qui affiche la moyenne de ces trois notes
2. Traduire en Python ce dernier algorithme

Ex 8
Définir les noms suivants et donner des exemples :
instruction , **expression**, **variable**, **affectation**.

Ex 9

Quelle est la valeur de la variable entier à la fin de l'algorithme ?

---

début
  entier ← 1
  répéter 4 fois
    entier ← entier + 2
fin

Ex 10

Donner un algorithme où M est un entier entré au clavier

1. Pour calculer la somme $S = 1^2 + 2^2 + 3^2 + ... + M^2$?
2. Pour calculer la somme $S = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{M}$?

Ex 11

1. Développer et simplifier les expressions algébriques suivantes :
   (a) $(x-1)(2x+3)$
   (b) $(y-1)(2y+3)+y(1+y)$
   (c) $(t+1)^2$
   (d) $(1-3z)^2$
   (e) $(a+b+c)(a+b-c)(a-b+c)(a-b-c)$
2. Dans l'interpréteur Python entrer les instructions suivantes (importer une partie de la bibliothèque `sympy` disponible dans l'environnement EduPython ou Pyzo)

```
>>> from sympy import symbols,expand

>>> x = symbols('x')

>>> expand((1-3*x)**2)
9*x**2 - 6*x + 1

>>> expand((x-1)*(2*x+3)+x*(1+x))
3*x**2 + 2*x - 3
```

Ex 12

1. Factoriser les expressions suivantes :
   (a) $(t-1)(t+2)+(2t-3)(t-1)$
   (b) $(x-1)(x+2)-(2x-3)(x-1)$
   (c) y^2-1
   (d) $4z^2-1$
   (e) $9-16x^2$
   (f) t^2+1
   (g) $2-5x^2$

(h) $9 - 16x^2$

(i) $9 - 16x^2$

(j) $(a^2 + b^2 + c^2)^2 - 4a^4b^4$

2.

3. Dans l'interpréteur Python entrer les instructions suivantes (importer une partie de la bibliothèque `sympy` disponible dans l'environnement EduPython ou Pyzo)

```
>>> from sympy import symbols,factor

>>> x,y,z,t = symbols('x y z t')

>>> factor((t-1)*(t+2)+(2*t-3)*(t-1))
(t - 1)*(3*t - 1)
```

Chapitre 4

Fonction et Variable

Ex 1 On rappelle que la pente d'un segment (ou le coefficient directeur) ou d'une droite définie par deux points A et B de coordonnées $(x_A; y_A)$ et $(x_B; y_B)$ relativement à un repère $(O; I; J)$ est définie par $\dfrac{y_B - y_A}{x_B - x_A}$

On suppose dans tout l'exercice que $x_A \neq x_B$

1. Est ce que $\dfrac{y_A - y_B}{x_A - x_B} = \dfrac{y_B - y_A}{x_B - x_A}$?
2. L'algorithme suivant défini une fonction `pente(`x_A`,`y_A`,`x_B`,`y_B`)` qui retourne la pente de (AB) et fait afficher à l'écran la pente du segment défini par A et B de coordonnées (1 ;2) et (2 ;0) Ecrire l'appel de la fonction dans l'algorithme :

---

début
  pente *(x_A,y_A,x_B,y_B)*
    retourner $\dfrac{y_A - y_B}{x_A - x_B}$
  .....
fin

---

3. Traduire en Python

Ex 2 Etant donné deux points A de coordonnées $(x_A; y_A)$ et B de coordonnées $(x_B; y_B)$ relativement à un repère (O, I, J)

1. Exprimer les coordonnées du vecteur $\overrightarrow{AB}$ en fonction des coordonnées de A et B relativement au repère (O, I, J)
2. Ecrire une fonction en pseudo-code `coordonnees(` x_A`,` y_A`,`x_B`,` y_B`)` qui retourne les coordonnées du vecteur $\overrightarrow{AB}$ en fonction des coordonnées de A et B
3. Traduire cette fonction en Python

Ex 3 Le déterminant de deux vecteurs $\vec{u_1}$ de coordonnées $(x_1; y_1)$ et $\vec{u_2}$ de coordonnées $(x_2; y_2)$ relativement à un repère (O, I, J) est défini par
$\det(\vec{u_1}\ ,\ \vec{u_2}) = x_1y_2 - x_2y_1$

1. Créer une fonction en pseudo-code `det(`x_1`,`y_1`,`x_2`,`y_2`)` qui retourne le déterminant
2. Traduire en Python

3. On montre en mathématiques que deux vecteurs sont colinéaires si et seulement si le déterminant de ces deux vecteurs est nul

   Créer une fonction en Python `colineaires(`x_1`,`y_1`,`x_2`,`y_2`)` qui retourne `True` si les vecteurs sont colinéaires

4. Que devrait retourner `colineaires(3,1,0.3,0.1)` ? Est ce le cas ?Sinon corriger en tenant compte de l'exercice 4 du chapitre 1

Ex 4

1. Ecrire en pseudo-code une fonction `appartientDroite(a,b,x,y)` qui retourne vrai si le point de coordonnées (x,y) appartient à la droite d'équation réduite $y = ax + b$
2. Traduire en Python
3. Que devrait retourner `appartientDroite(3,0,0.1,0.3)` ? Est ce le cas ?Sinon corriger en tenant compte de l'exercice 4 du chapitre 1

Ex 5

1. Ecrire une fonction `milieu(`x_A`,`y_A`,`x_B`,`y_B`)` qui retourne les coordonnées du milieu du segment $[AB]$
2. Traduire cette fonction en Python

Ex 6

1. Etant donné une fonction polynôme du second degré f définie sur $\mathbb{R}$ par : $f(x) = ax^2 + bx + c$. Quelles sont les coordonnées du sommet de la parabole associée à f ?
2. Ecrire une fonction `coordonneesSommetParabole(a,b,c)` en Python

Chapitre 5

Test

Ex 1

On reprend l'exercice 1 du chapitre précédent sur la pente d'un segment.
Il s'agit maintenant de tenir compte du cas où $x_A = x_B$.
Faire un test au début de la fonction ainsi
Si $x_A == x_B$ alors afficher ('la pente est infinie') Sinon .....

Ex 2

Pour simuler le lancer d'un dé à six faces on va utiliser la fonction `randint` de la bibliothèque `random`

`randint(1,6)` retourne un entier "au hasard" entre 1 et 6 inclus

1. Faire un programme qui simule le lancer de 10 fois un dé et affiche les 10 lancers
2. Compléter le programme suivant pour pouvoir calculer la fréquence d'apparition du six pour 1000 lancers

```
from random import randint
nbSix = 0
for i in range(1000):
    de = randint(1,6)
    if  .......:
        .......
print("la fréquence pour 1000 lancers est ",........)
```

Ex 3

1. Ecrire une fonction en pseudo-code `max(x,y)` qui retourne le plus grand de deux nombres x et y
2. Traduire la fonction en Python.
3. Utiliser cette fonction pour écrire une autre fonction `max(x,y,z)` qui retourne le plus grand des trois nombres x,y et z
4. Cette fonction existe déjà et fait partie des fonctions de base de Python en dehors de toute bibliothèque importée (on parle de fonction built-in ou native) aller voir ici `https://docs.python.org/fr/3.5/library/functions.html` sa documentation et comparer votre documentation à celle de Python

   On veut généraliser la fonction à n'importe quel ensemble fini de nombres

```
début
    max (liste)
        maximum ← premier élément de la liste
        pour chaque élément x de la liste faire
            si x > maximum alors
                .....
            fin
        fin
        retourner maximum
fin
```

puis le programme

```
def max(liste):
#on prend le premier élément de la liste
    maximum = liste[0]
    #on parcourt la liste ou on itère sur la liste
    for x in liste:
        if x > maximum:
            ..........
    return maximum
```

Attention notre fonction est moins souple que celle de Python on appellera cette fonction ainsi

max((1,2,3)) ou max([1,2,3]) et non pas max(1,2,3) sinon il y aura un message d'erreur

Ex 4

1. Faire une fonction Python `nbSix(n)` qui retourne le nombre de fois où on a obtenu six lorsqu'on a lancé n fois un dé
2. Faire une fonction Python `auMoinsUnSix()` qui retourne vrai si on a obtenu **au moins** un six lorsqu'on a lancé 4 fois un dé
3. Sur lequel des deux évènements suivants est-il préférable de parier :

   A :"obtenir au moins un six lorsqu'on lance 4 fois un dé" ou l'évènement contraire $\overline{A}$?

   (Simuler 1000 répétitions de 4 lancers d'un dé et calculer la fréquence d'obtenir au moins un six)

Ex 5 Ecrire une fonction `sensVariationsAffine(a)` qui retourne la chaîne de caractères 'croissante sur R' si la fonction affine définie par $f(x) = ax + b$ est croissante sur R et qui retourne la chaîne de caractères 'décroissante sur R' si la fonction affine définie par $f(x) = ax + b$ est décroissante sur R

Ex 6 Ecrire une fonction `sensVariationsTrinome(a,b,c)` qui retourne le sens de variations de la fonction trinôme f définie par $f(x) = ax^2 + bx + c$

Ex 7 La fonction valeur absolue d'un nombre réel est une fonction native de Python. Redéfinir cette fonction en complétant la fonction suivante :

```
def valeurAbsolue(x):
    if x >= 0:
        return  ........
    else:
        return .........
```

Ex 8

1. Ecrire en pseudo code une fonction `positionRelative(a,b,x,y)` qui retourne la chaîne de caractère 'au-dessus' si le point de coordonnées (x ;y) est au-dessus de la droite d'équation réduite $y = ax + b$, qui retourne 'appartient' si le point de coordonnées (x ;y) appartient la droite d'équation réduite $y = ax + b$ et qui retourne 'en-dessous' si le point de coordonnées (x ;y) est en dessous de la droite d'équation réduite $y = ax + b$
2. Traduire la fonction en Python et visualiser son fonctionnement sur PythonTutor

Ex 9

1. Ecrire une fonction en pseudo-code :

   `estUnParallelogramme(`x_A`,` y_A`,`x_B`,` y_B`,` x_C`,` y_C`,`x_D`,` y_D`)` qui retourne Vraie si le quadrilatère $ABCD$ est un parallélogramme

   Les paramètres de la fonction sont les coordonnées des points A,B,C et D relativement à un repère (O, I, J)
2. Traduire cette fonction en Python

Ex 10(Problème du duc de Toscane) On lance 3 dés et on étudie la somme des trois chiffres obtenus.

Lequel des deux évènements suivants est le plus probable ?

A :" La somme des trois chiffres vaut 9 "

B :" La somme des trois chiffres vaut 10 "

Faire un programme qui simule 1000 lancers de 3 dés et qui calcule la fréquence de chaque évènement puis conclure

Chapitre 6

Boucle Tant Que

Ex 1

On lance un dé à six faces autant de fois que l'on veut tant que le chiffre six n'est pas sorti. On compte le nombre de fois où on a lancé le dé. On utilise une variable `nbLancers` puis on affiche le contenu de `nbLancers` à la fin de l'algorithme.

1. Compléter l'algorithme suivant :

```
début
    de ← entierAleatoire(1,6)
    nbLancers ← .....
    tant que ..... faire
        .....
        .....
    fin
    afficher(.....)
fin
```

2. Transformer cet algorithme en fonction `nombreLancers(chiffre)` qui retourne le nombre de lancers de dé effectués jusqu'à ce que le chiffre `chiffre` soit sorti
3. Appeler 100 fois cette fonction avec le paramètre six et calculer le nombre moyen de lancers.

   Introduire pour cela une variable `nbLancersMoyen` qu'il faudra correctement initialiser puis mettre à jour

   Afficher le contenu de `nbLancersMoyen` à la fin de l'algorithme
4. Traduire en python et exécuter le programme

Ex 2

1. Ecrire en pseudo-code une fonction `sylvester(x)` qui retourne l'image de x par la fonction f définie par $f(x) = x(x-1)+1$
2. On appelle suite de Sylvester la suite de nombres définie ainsi :

   On part de 7 puis on calcule $f(7) = 43$ puis on calcule $f(43)$ etc...

   Ecrire une fonction `suiteSylvester(n)` qui calcule les n premiers termes de la suite de Sylvester et les affiche à l'écran

3. Ecrire une fonction `suiteSylvester(seuil)` qui calcule les termes x de la suite de Sylvester tel que $x \leqslant$ seuil et les affiche à l'écran

Ex 3

Voici une autre suite de nombres :

Etant donné un nombre au départ par exemple 3, tant qu'on obtient pas 1 on effectue la suite d'instructions suivantes, si le nombre est pair on le divise par 2 sinon on lui multiplie 3 et on rajoute 1.

Ainsi en partant de 3 on obtient :

3, 10, 5, 16, 8, 4, 2, 1

1. Ecrire une fonction `syracuse(nombre)` en pseudo-code puis un programme Python qui affiche à l'écran la suite des nombres obtenue en partant du nombre `nombre`

   Est on sûr que la boucle Tant que se termine toujours quelque soit le nombre `nombre` ?

2. Modifier la fonction précédente de telle sorte qu'elle retourne le nombre de termes de la suite sans tenir compte de 1

3. Modifier encore la fonction de telle sorte que cette fois ci elle retourne la plus grande valeur atteinte par la suite

Ex 4 (La multiplication egyptienne)

Pour calculer le produit de deux nombres entiers a et b avec $a \geqslant b$, les anciens Egyptiens procédaient ainsi :

On introduit une variable `resultat` initialisé à 0

Si b est **pair** alors on remplace a par $2a$ et on remplace b par $\frac{b}{2}$ en effet $ab = (2a)\frac{b}{2}$

Si b est **impair** alors on remplace b par $b-1$ et on rajoute a à `resultat`

Puisque b diminue à chaque répétition on est sûr qu'au bout d'un moment b sera égal à 1, par conséquent on répète le processus **tant que** $b > 2$

A la fin on rajoute $2a$ à `resultat` et on retourne `resultat`

1. Ecrire une fonction `multEgyptienne(a,b)` qui retourne le produit de deux entiers a et b

2. Est que si on entre comme données deux nombres à virgule aura-t-on quand même le produit de ces deux nombres ?

3. Que se passe-t-il si on entre pour a la chaîne de caractères 'Hello' et pour b le nombre 2.5 ?

Ex 5

La Suite de Fibonnacci est la suite de nombres entiers 0 1 1 2 3 5 8 13, etc... un terme quelconque est la somme des deux termes qui le précèdent

Les deux termes initiaux sont 0 et 1

1. Calculer encore quelques termes

2. Ecrire une fonction en pseudo-code `fibonnacci(seuil)` qui affichent à l'écran, tous les termes x de la suite tel que $x \leqslant seuil$

   (Introduire deux variables `terme1` et `terme2` initialisés respectivement à 0 et à 1)

Ex 6 On définit le nombre d'or ϕ par $\phi = \dfrac{1+\sqrt{5}}{2}$

1. Vérifier que le nombre d'or est solution de l'équation $x^2 - x - 1 = 0$
2. Vérifier en développant que $x^2 - x - 1 = (x - \frac{1}{2})^2 - \frac{5}{4}$
3. Factoriser $(x - \frac{1}{2})^2 - \frac{5}{4}$ et retrouver que ϕ est solution de $x^2 - x - 1 = 0$. Vérifier que l'autre solution est $\overline{\phi} = \dfrac{1-\sqrt{5}}{2}$
4. On montre que le quotient de deux termes consécutifs de la suite de Fibonnacci (le plus grand sur le plus petit) "se rapproche indéfiniment" de ϕ

   Ecrire une fonction `nombreOr(seuil)` qui affichent la suite des rapports obtenus avec deux termes consécutifs de la suite de Fibonnacci inférieurs à `seuil` (en faisant le rapport du plus grand sur le plus petit)

Ex 7

Voici un jeu entre vous et l'ordinateur.

L'ordinateur choisit un nombre entier au hasard entre 1 et 1000. Ce nombre vous est inconnu.

On affecte la variable `nombreInconnu` par ce nombre.

```
from random import randint
nombreInconnu = randint(1,1000)
```

Vous devez en un minimum de coups deviner ce nombre.

Comment ?

Tant que vous n'avez pas trouvé ce nombre vous pouvez suggérer un nombre et l'ordinateur vous répond 'Trop grand' si le nombre que vous avez proposé est strictement plus grand que le nombre solution, et 'Trop petit' si le nombre que vous avez proposé est strictement plus petit que le nombre solution

Compléter le programme suivant :

```
from random import randint
nombreInconnu = randint(1,1000)
nombrePropose =  int(input("Quel nombre proposez vous ?  ")
nbCoups = .......
while ..................:
    if ................:
        ...................
    else:
        ...................
    nombrePropose = int(input("Quel nombre proposez vous ?
")
    nbCoups = ............
print("Vous avez trouvé le nombre ",............,"  en ",.........,"
coups")
```

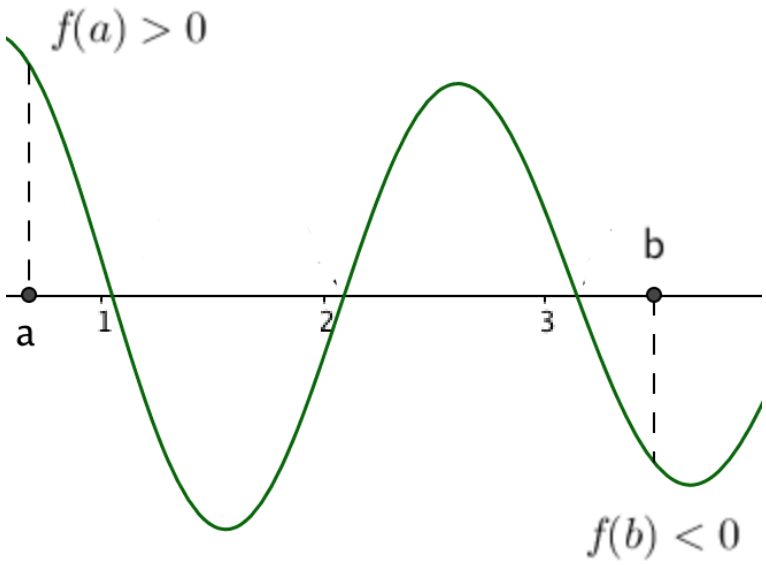
Quelle stratégie adopter pour minimiser le nombre de coups ?

Ex 8 On ne cherche pas forcément à résoudre des équations de manière exacte.

Aussi on va développer dans cet exercice une méthode de calcul approché pour résoudre une équation

1. Résoudre dans l'ensemble des nombres réels l'équation $2x - 3 = 0$
2. Vérifier que $4x^2 - 8x + 1 = 4((x-1)^2 - \frac{3}{4})$
3. En déduire les solutions de l'équation $4x^2 - 8x + 1 = 0$
4. Maintenant on veut résoudre $f(x) = 0$ de manière approchée parce que la fonction f est plus "compliquée" que les deux fonctions précédentes (tout en restant relativement "régulière", on parle de fonction continue)

   Un théorème de mathématiques (théorème des valeurs intermédiaires) dit que si f est une fonction continue sur$[a;b]$ et si $f(a)$ et $f(b)$ sont de signes contraires alors l'équation $f(x) = 0$ a **au moins** une solution sur l'intervalle $[a;b]$



   (a) Graphiquement combien y-a-t-il de solutions à l'équation $f(x) = 0$ dans l'intervalle $[a;b]$?

   (b) Montrer que si $f(a)$ et $f(b)$ sont de signes contraires alors $f(a)f(b) < 0$ (et réciproquement)

   (c) Maintenant on se place dans le cas où f est **monotone** sur $[a;b]$ c'est à dire **croissante** ou **décroissante** sur $[a;b]$. Faire des dessins pour conjecturer le nombre de solutions à l'équation $f(x) = 0$

   (d) Nous allons maintenant étudier un algorithme d'approximation de la solution (méthode par dichotomie)

   L'idée est : Diviser par 2 l'intervalle I sur lequel f change de signe en deux intervalles I_1 et I_2. f change de signe sur l'un des deux intervalles donc recommencer avec ce nouvel intervalle jusqu'à ce que le diamètre de l'intervalle soit inférieure à un nombre `precision` donné à l'avance.

   Compléter la fonction suivante `dichotomie(a,b,f,precision)` qui retourne une valeur approchée de la solution $f(x) = 0$

f est définie dans une autre fonction, on prend pour f la fonction du 2)

```
début
    f (x)
        retourner 4*x*x-8*x+1
    dichotomie (a,b,f,precision)
        borneInf ← a
        borneSup ← b
        tant que borneSup-borneInf > precision faire
            milieu ← .....
            si f(borneInf)f(milieu) > 0 alors
                .....
            sinon
                .....
            fin
            .....
        fin
        retourner .....
    dichotomie(...,...,f,...)
fin
```

5. Traduire en Python et exécuter, puis comparer avec les valeurs exactes obtenues en 2)

Chapitre 7

Expressions logiques

Ex 1

1. Quel est l'intervalle des nombres réels x tel que $x > 1$?
2. Quel est le contraire de $x > 1$?
3. Quel est l'intervalle des nombres réels x tel que $x > 1$ et $x \leqslant 2$?
4. Quel est le contraire de $x > 1$ et $x \leqslant 2$?

Ex 2

1. Que signifie :"Obtenir **au moins un six** lorsqu'on a lancé 4 fois un dé" ?
2. Quel est le contraire de :"Obtenir **au moins un six** lorsqu'on a lancé 4 fois un dé" ?

Ex 3
Avant l'entraînement un entraineur de football pose la question :
"Est ce que **tous** les ballons de football sont gonflés ?"
Si vous répondez à cette question par la **négative**, est ce que vous répondez ?
"Tous les ballons ne sont pas gonflés"
Ex 4
`a` et `b` sont deux variables contenant soit Vrai soit Faux
Est ce que le contraire de `a et b` est `(non a) et (non b)` ?
Ex 5 On reprend l'exercice 4 du chapitre 5.

1. On veut écrire une fonction `auMoinsUnSix()` qui retourne vrai **dès que l'on a détecté un six au cours des 4 lancers**

   Première méthode :

   On répète 4 fois le lancer de dé et si on obtient un six on sort de la boucle en retournant Vrai

   Compléter la fonction puis l'algorithme dans lequel on appelle 1000 fois cette fonction pour ensuite afficher la fréquence de l'évènement "Obtenir au moins un six sur une série de 4 lancers d'un dé" dans une série de 1000 répétitions .

   Ensuite traduire en Python et exécuter.

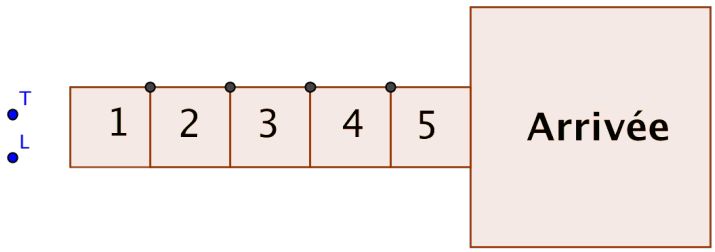
```
début
    auMoinsUnSix ()
        répéter 4 fois
            de ← entierAleatoire(1,6)
            si ..... alors
                retourner True
            fin

        .....
    nbSucces ← .....
    répéter 1000 fois
        si ..... alors
            ..... ;
        fin

    afficher(.....)
fin
```

2. Deuxième méthode :
   Tant que l'on n'a pas lancé 4 fois le dé **et** obtenu un six on lance le dé
   Mettre au point cette méthode
3. **Comparer** les deux méthodes

Ex 6



Un lièvre (L) et une tortue (T) s'affrontent dans une course aléatoire.
On lance un dé si on obtient six le lièvre gagne sinon la tortue avance d'une case
La tortue peut donc gagner lorsqu'elle se trouve à la case 5 et si au cours du lancer suivant le six ne sort pas
Nous supposons dans un premier temps que nous disposons de la fonction suivante :
`lievreEstVainqueur()` qui retourne vrai si le lièvre a gagné **une** course

1. Ecrire un algorithme où on répète 1000 courses et où on affiche la fréquence de gain du lièvre au cours de ces 1000 courses
2. Ecrire cette fonction `lievreEstVainqueur()`
3. Traduire en Python et exécuter. Qui a le plus de chances de gagner le lièvre ou la tortue ?

Ex 7 (Jeu du Crap)

Le Jeu du Crap est le jeu de dés suivant opposant la banque à un joueur :

On lance deux dés et on calcule la somme P (comme première somme) des deux chiffres obtenus : (Il est important pour comprendre la suite de bien saisir que le premier lancer , donc P joue un rôle important et conditionne les résultats du jeu)

Si $P = 7$ ou 11 le joueur a gagné, si $P = 2$ ou 3 ou 12 la banque a gagné, sinon pour les autres résultats on relance les deux dés autant de fois que nécessaire jusqu'à ce qu'on obtienne 7 comme somme dans ce cas la banque gagne ou on retombe sur P et dans ce cas le joueur gagne

1. Nous supposons dans un premier temps que nous disposons de la fonction suivante `laBanqueGagne()` qui retourne Vrai si la banque gagne **une** partie

   Ecrire un algorithme où on répète 1000 parties et où on affiche la fréquence de gain de la banque au cours de ces 1000 parties
2. Ecrire cette fonction `laBanqueGagne()`
3. Traduire en Python et exécuter. Qui a le plus de chances de gagner la banque ou le joueur ?

Ex 8

On lance une pièce autant de fois que nécessaire jusqu'à ce que pile (0) et face (1) sont apparus.

Par exemple on peut obtenir 01 ou 0001 ou 111110

1. Ecrire une fonction `nbLancers()` qui retourne le nombre de lancers de pièce effectué pour obtenir pile et face

   On va utiliser deux variables `faceNonSorti` et `pileNonSorti` initialisées à Vrai.

   (On dit que le type de ces variables est booléen)

   Dès que pile ou face sortira ces variables seront affectées en conséquence, ce qui permettra d'arrêter de lancer la pièce

```
début
    nbLancers ()
        faceNonSorti ← Vrai
        pileNonSorti ← Vrai
        nbLancers ← .....
        tant que ..... faire
            piece ← entierAleatoire(0,1)
            .....
            si ..... alors
                .....
            fin
            si ..... alors
                .....
            fin
        fin
        .....
fin
```

2. Répéter 1000 fois cette fonction pour calculer le nombre moyen de lancers pour 1000 répétitions
3. Conjecturer le nombre moyen de lancers
4. Traduire en python et exécuter.

Ex 9

On généralise l'exercice précédent avec un dé à 4 faces.

On lance le dé autant de fois que nécessaire jusqu'à ce que les chiffres 1, 2, 3 et 4 sont tous apparus.

Par exemple on peut obtenir 22432332441

On cherche le nombre moyen de lancers de dé pour 1000 répétitions

Ex 10

On généralise l'exercice cette fois ci à un dé à six faces et on va changer de méthode.

On va utiliser une liste `nonSorti` telle que `nonSorti[i]` est vrai si le chiffre `i` n'est pas encore sorti

On initialise la liste ainsi :

`nonSorti` ← `[Faux, Vrai, Vrai, Vrai, Vrai, Vrai, Vrai]`

Le premier élément de la liste `nonSorti[0]` n'est pas significatif et il est initialisé à Faux pour simplifier la suite

La condition pour continuer de lancer le dé est qu'il y ait au moins un Vrai dans cette liste

1. Ecrire une fonction en pseudo-code `auMoinsUnChiffreNonSorti(liste)` qui retourne Vrai si dans la liste de booléens `liste` il y a au moins un Vrai

```
début
    auMoinsUnChiffreNonSorti (liste)
        pour chaque élément x de la liste faire
            si ..... alors
                retourner .....
            fin
        fin
        retourner .....
fin
```

2. Utiliser cette fonction et finir l'exercice (on cherche le nombre moyen de lancers de dé pour 1000 répétitions)

Ex 11

Existe-t-il une relation entre le nombre de possibilités 2, 4 , 6, etc... et le nombre moyen de lancers de dé ?

Index

D

Droites dans le plan
- coefficient directeur **12**
- équation de droite **13**, **16**

E

Equations
- équation du premier degré **6**
- équation du second degré **19**
- résolution approchée par dichotomie . **19**

Expressions algébriques
- développer **10**
- factoriser **10**

F

Fonctions
- maximum **14**
- sens de variation **15**
- sommet de la parabole **13**
- valeur absolue **16**

G

Géométrie dans l'espace
- volume **8**

Géométrie plane
- aire **8**
- carré **6**
- losange **5**
- milieu d'un segment **13**
- parallélogramme **5**, **16**
- rectangle **5**

N

Nombres
- $\sqrt{2}$ **5**
- entier **3**
- nombre décimal **5**
- nombre rationnel **6**

P

Probabilités
- Problème du Duc de Toscane ... **16**
- Temps d'attente d'un évènement ... **24**
- Jeu du Chevalier de Méré .. **15**, **22**
- Jeu du Crap **24**
- Jeu du lièvre et de la tortue **23**

S

Statistiques
- moyenne **9**

V

Vecteurs
- colinéaires **13**
- coordonnées **12**
- déterminant **12**