6 TP pour apprendre le langage Python

I. TP n°1 : Objets de type simple

Ce premier TP permet de découvrir quelques objets et syntaxes élémentaires du langage de programmation Python. Il est mené en activité dialoguée. Vous conserverez précieusement tous les TP réalisés dans votre classeur.

1) Différents types

Les langages de programmation traitent toute sorte d'objets et de données : des nombres aux mots, des images aux fichiers, jusqu'aux sons, tous convertis en dernier ressort en une suite de 0 et de 1 . Le but de ce TP est de découvrir un certain nombre d'objets que peut manipuler python : nombres, booléens, chaînes de caractères ; et les opérations que l'on peut effectuer avec les nompbres. On effectuera les saisies dans la console.
Tout langage possède ses règles. On appelle syntaxe l'ensemble de ces règles. Chaque objet manipulé par python a un type, qui s'écrit avec une syntaxe particulière.
Ainsi, ce qui ne ferait a priori pas grande différence pour nous en français dans les notations du nombre 1 : `1`, `1.0`,`1==0+1`, `' 1 '`, `␣[1]`, `␣'un'`, recouvrent sous python des réalités différentes. Une fonction prédéfinie essentielle pour savoir à quel type d'objet on a affaire est la fonction : `type`.

Exercice 1 Que renvoie la console Python quand on saisit ?

`type(1)`		`type(1.0)`		`type(1+1==2)`	
`type(-2020)`		`type(3.14)`		`type(5<4)`	

Quels sont les 3 types d' « *objets* » découverts à travers ces exemples ?

.....

> **Remarque** On remarque dès à présent la différence entre deux symboles « *égal* » à la suite : ==, et un seul symbole =. La succession des deux symboles = est le test d'égalité classique. Un seul symbole = signifie qu'une valeur est affectée à une variable. Voir paragraphe 5.

2) Les nombres

a) Nombres en Mathématiques et pour la machine

Python peut traiter les nombres comme une calculatrice peut le faire. Attention, une machine ne peut garder en mémoire qu'un **nombre fini** de chiffres. Quand elle traite par exemple la fraction $\frac{1}{3}$ qui a une écriture décimale illimitée, elle ne considère que des **valeurs décimales approchées** appelées **nombres à virgule flottante** (**float** en anglais).
En classe de Mathématiques, nous apprendrons à classer les nombres dans des familles emboîtées comme le zoologue classerait les animaux.
Par exemples les nombres $0,3$, $\frac{1}{3}$, et $\sqrt{2}$ ne sont pas de même nature.

Le nombre $0,3$ est un nombre dont la partie décimale est finie. C'est un nombre .....

Le nombre $\frac{1}{3}$ est un nombre dont la partie décimale contient une infinité de chiffres qui se répètent :

$$\frac{1}{3} \approx 0,\underline{3}3333333333\ldots$$

Le nombre $\frac{1}{3}$ est un nombre à l'**écriture décimale périodique illimitée**. C'est un nombre ..... non décimal.

Le nombre $\sqrt{2}$ est un nombre à l'**écriture décimale illimitée** non périodique.
Voici une valeur approchée de $\sqrt{2}$ fournie par Python : 1.4142135623730951

C'est un nombre .....

Pour le langage Python tous les nombres précédents sont des « *flottants* » (présence d'une « *virgule* » notée avec un point).

La console Python renvoie True lorsqu'on teste l'égalité entre $\frac{1}{3}$ et 0,3333333333333333 . Qu'en pensez-vous ?

Début de la Console Python

```
>>> 1/3==0.33333333333333333333333333333333333333
True
```

Fin de la Console Python

.....

b) Nombres et opérations

Addition, soustraction, multiplication, division, exponentiation (puissance), valeur arrondie au nombre de décimales fixé près, quotient et reste dans la division euclidienne de deux entiers sont disponibles dans Python. Les exercices suivants permettent de les découvrir.

Exercice 2 Que renvoie la console Python lorsqu'on saisit les opérations suivantes ?

`20+3`		`20-3`		`20*3`		`20/3`	
`20**3`		`20//3`		`20%3`		`round(20/3,2)`	

Retrouver la syntaxe Python des opérations arithmétiques élémentaires sur les nombres en complétant le tableau suivant :

Opérations	Syntaxe Python
Addition de a et b	
Multiplication de a par b	
Division de a par b	
Exponentiation a^b	
Quotient de la division euclidienne de a par b	
Reste de la division euclidienne de a par b	
Valeur arrondie de a à p décimales près	

Les règles de priorité des opérations pour Python sont les mêmes qu'en Mathématiques. Vous pouvez les mémoriser grâce à l'acronyme PEMDAS :

- P pour « *Parenthèses* » : ce sont elles qui ont la plus haute priorité. Elles permettent de forcer l'évaluation d'une expression dans l'ordre que vous voulez.
- E pour « *Exposant* » : Les exposants sont évalués ensuite, avant les autres opérations.
- M pour « *Multiplication* » et D pour « *Division* » : elles sont évaluées avant l'addition et la soustraction, ont égales priorité et sont traité de la gauche vers la droite.
- A pour « *Addition* » et S pour « *Soustraction* » : elles ont égales priorité et sont traité de la gauche vers la droite.

Exercice 3 Quels sont les résultats des opérations suivantes dans la console Python ?

`2+3-5`		`2+3*5`		`2*3+5`	
`(2+3)*5`		`2*3/5`		`2+3/5`	
`(2+3)/5`		`2/(3+5)`		`5+2**3`	
`5*2**3`		`(5+2)**3`		`(5*2)**3`	

3) Les booléens

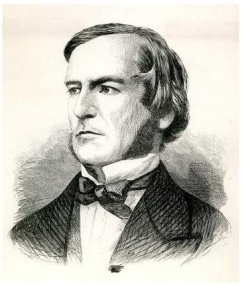
Un booléen ne peut prendre que deux valeurs : vrai ou faux. Dans le vocabulaire de la logique, cela correspond à une assertion. Une assertion est une phrase, un énoncé pouvant être soit vrai, soit faux, et c'est tout. C'est le type de « *phrases* », d'affirmations auxquelles s'attaquent les Mathématiques, finalement très simples : Tel triangle est rectangle ou non, ces trois points sont alignés ou pas, cette égalité est vraie ou fausse.

Dans la vie de tous les jours, nous nous exprimons sur le monde qui nous entoure, les propriétés des objets, des êtres, etc. Les phrases que nous énonçons sont souvent subjectives, ce sont de simples opinions, et n'ont pas le caractère binaire du booléen concernant la vérité.

Par exemple, considérons la phrase : « *Pau est une grande ville* ». Pour une personne habitant à Affoux, sans doute, pour un parisien ou un New-yorkais certainement pas. Cette phrase n'est pas assez précise pour la reconnaître comme une assertion mathématique.

En revanche, la phrase : « *Pau a moins d'habitants que Paris* » peut être considérée comme une assertion mathématique. Il se trouve que cette assertion est vraie.

« *Pau a plus d'habitants que Paris* » peut être aussi considérée comme une assertion mathématique. Il se trouve que cette assertion est fausse. En Mathématiques, nous nous intéressons à ce dernier type d'affirmations, que nous appelons assertions ou propositions, et qui n'ont que deux valeurs de vérités qui s'excluent l'une l'autre. C'est que l'on appelle un booléen en informatique.



Georges Boole, logicien anglais (1815-1864)

Exercice 4 Que renvoie la console Python quand on saisit les booléens suivants ?

`1==0`		`1!=0`		`5>4`	
`-5>-4`		`2**3==6`		`6%2==3`	
`6/2==3`		`2021%2==0`		`3**2==9`	

Symbole mathématique	Egal	Différent
Syntaxe Python	==	! =

4) Variables et affectation

L'essentiel du travail effectué par un programme d'ordinateur consiste à manipuler des données. Ces données peuvent être très diverses (tout ce qui est numérisable en fait). Pour pouvoir accéder aux données, un programme fait abondamment usage d'un grand nombre de variables de différents types. Une variable apparaît dans un langage de programmation sous un nom (une lettre, un mot) mais pour l'ordinateur il s'agit d'une référence désignant une adresse mémoire, c'est-à-dire un emplacement précis dans la mémoire vive. À cet emplacement, est stockée une valeur bien déterminée.

En résumé, une variable (en informatique) est un espace mémoire dans lequel il est possible de mettre une valeur. Cette opération est appelée : ..... .

Si la variable affectée n'existe pas, l'ordinateur la crée, sinon il écrase sa valeur. En effet, il est permis de réaffecter une nouvelle valeur à une même variable, autant de fois qu'on le souhaite.

Quand on décrit un algorithme en langage naturel, et que l'on affecte, par exemple, le nombre 1 à la variable a, voici comment on note cette affectation :

$$a \leftarrow 1$$

La flèche indique le « *mouvement* » effectué et le sens de lecture : une affectation se lit de droite à gauche. On évalue ce qui est à droite de la flèche et on vient le « *loger* » dans la variable se situant à gauche de la flèche. Considérons le petit programme de calcul suivant :

Pseudo-Code

```
a ← 1
b ← 2
a ← a+3
b ← a \times 2
a ← a+b
```

Il y a 5 affectations, voir réaffectations successives. Les variables changent de valeurs au cours des affectations. Compléter le tableau d'états des variables a et b au cours de ces 5 affectations :

Ligne du programme	Contenu de la variable a	Contenu de la variable b
1		
2		
3		
4		
5		

Python note l'égalité, nous l'avons vu, par un double signe égal ==, et il note par un simple égal = l'affectation. Ainsi, l'algorithme ci-dessus s'écrirait :

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

	Affectation	Signification
Symbole algorithmique	a $\leftarrow$ 1	La variable a reçoit la valeur 1
Syntaxe Python	a=1	La variable a reçoit la valeur 1

On rappelle les différences essentielles entre l'égalité au sens mathématique et l'affectation au sens informatique :

- L'égalité est **symétrique** : On peut incrire indifféremment $2+1=3$ ou $3=2+1$, tandis que l'affectation ne l'est pas. On évalue au cours d'une affectation d'abord le membre de droite que l'on vient loger dans la variable située à gauche. Par exemple, pour l'affectation : a=1+2. On calcule 1+2, on obtient 3 , puis a prend la valeur 3 . L'écriture : 1=a n'a aucun sens pour Python car 1 est reconnu comme un entier qui ne peut être un nom de variable.
- Si a est un nombre entier, l'égalité $a = a+3$ n'a pas de sens car elle conduirait à : $3=0$. Il n'existe pas de nombre entier a vérifiant : $a = a+3$. En revanche, dans la séquence Python :

  Code Python
  ```
  a=1
  a=a+3
  ```

  a prend successivement la valeur 1 , puis 4.

Exercice 5 Donner le contenu des variables a et b à la fin des 3 programmes suivants :

Programme n°01

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

$a =$ . . . . . .

$b =$ . . . . . .

Programme n°02

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

$a =$ . . . . . .

$b =$ . . . . . .

Programme n°03

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

$a =$ . . . . . .

$b =$ . . . . . .

Remarque
Les 3 programmes présentent les mêmes 5 affectations mais dans un ordre différent. Cela affecte-t-il le résultat final des contenus des variables a et b ?

.....

II. TP n°2 : Objets de type composé

Après avoir évoqué les objets de type simple, les nombres et les booléens, et quelques règles opératoires, ainsi que la notion primordiale d'affectation, on présente des objets de type composé, vus comme une collection d'objets de type simple : les chaînes de caractère et les listes. La présentation se fait en activité dialoguée avec le professeur et les exercices en autonomie à l'aide de l'ordinateur.

1) Chaînes de caractère

Une chaîne de caractères (de type string) est une suite ordonnée de caractères (caractères unicode du clavier : lettres, symboles) délimitée par des ..... éventuellement par des .....
Quand un programme travaille avec des données sous forme de mots par exemple, les chaînes de caractères interviennent.

Code Python
```
motdepasse= 'Pythagore'
```

Il est possible d'accéder au caractère individuel d'une chaîne de caractères. Pour accéder à un élément de la chaîne `C` , il suffit d'écrire `C[i]` où i est le rang de l'élément souhaité.

Attention, il y a un piège. On commence à compter à partir du rang 0 . Le premier élément de la liste est donc celui de rang 0 , le second de rang 1 , etc. Par exemple, compléter ce que renvoie la console dans les 4 cas suivants :

- `>>>motdepasse[0]` : .....
- `>>>motdepasse[1]` : .....
- `>>>motdepasse[3]` : .....
- `>>>motdepasse[6]=='g'` : .....

2) Exercices

On proposera une réponse sans l'aide de l'ordinateur, puis on la contrôlera grâce à la console.

Exercice 6 Que renvoie la console quand on saisit

`Prenom='Albert'`	`Prenom[0]`	
`Prenom='Albert'`	`Prenom[0]+'.'+'Einstein'`	
`Prenom='Albert'`	`'T' in Prenom`	

> **Remarque** Le mot réservé **in** permet de tester si un élément (de type simple) est dans un objet de type composé.

III. TP n °3 : Fonctions

Les fonctions sont un concept central des Mathématiques, comme du langage de programmation Python. Le plus souvent, à partir de paramètres divers pris en **entrée(s)**, une fonction Python renvoie des résultats (sous la forme d'un nombre, d'une liste, d'un booléen, etc.) en **sortie(s)**.
Les paragraphes 1. et 2. sont menés en activité dialoguée avec le professeur pour découvrir la notion de fonction Python et sa syntaxe (règles d'écriture). Le paragraphe 3. est traité en autonomie à l'aide de l'ordinateur. On enregistrera son travail (dossier TP).

1) Premiers exemples

a) Moyenne trimestrielle

On peut écrire une fonction qui prend en entrées les trois notes en Mathématiques d'un élève (affectés d'un même coefficient) à un trimestre, et renvoie sa moyenne.

Code Python

```
def moyenne(a,b,c):
    M = ( a + b + c ) / 3
    return M
```

Que renvoie `moyenne(10,9,14)` dans la console ? .....

b) Test de divisibilité

On peut écrire une fonction qui prend en entrée un entier et renvoie « *vrai* » s'il est divisible par 7 et « *faux* » sinon.

Code Python

```
def test(n):
    r=n%7
    return r==0
```

Que renvoie `test(63)` dans la console ? ..... . Et `test(22)` ? .....

c) Prix hors taxe et toutes taxes comprises

On peut écrire une fonction qui prend en entrée un prix hors-taxe (H.T.) en euros et renvoie la liste contenant : le prix H.T. pris en entrée, le montant d'une T.V.A. à 20% (taxe sur la valeur ajoutée qui correspond à 20% du prix), et le prix T.T.C. (toutes taxes comprises).

Code Python

```
        def prix(p):
        t=0.2*p
        TTC=p+t
        return [p,t,TTC]
```

Que renvoie `prix(60)` dans la console ? .....

2) Fonctions somme, produit et opposé

a) Fonction somme

1) Ecrire dans l'éditeur la fonction :

Code Python

```
def somme(a,b):
    return a+b
```

2) Que renvoie dans la console `somme(3,4)` ? .....

Et `somme(120,45)` ? .....

3) Quel est le rôle de cette fonction ? .....

b) Fonction produit

1) Ecrire une fonction Python produit qui prend en entrées deux entiers et renvoie leur produit.

2) Tester-la dans la console en observant le résultat renvoyé par `produit(3,4)`.

3) Avant d'utiliser Python, quel est le chiffre des unités du résultat de la multiplication de 1789 par 1515 ? ..... Noter le résultat renvoyé par `produit(1789,1515)`. .....

4) A-t-on besoin de la fonction `produit` pour connaître le résultat de `produit(1234,1)` ?

.....

Et de `produit(1234,0)` ? .....

c) Fonction opposé

1) Ecrire une fonction Python oppose qui prend en entrée un entier relatif et renvoie son opposé.

2) Tester-la dans la console sur des entiers relatifs positifs et négatifs.

3) Que renvoie dans la console `oppose(oppose(1789))` ? ..... Pouvait-on s'y attendre ? .....

d) Fonctions emboîtées

1) Pourquoi obtient-on le même résultat avec `produit(2,somme(3,4))` et `somme(produit(2,3),produit(2,4))` ?

.....

2) Compléter le tableau suivant. Vous utiliserez la console Python pour contrôler vos résultats.

Fonctions Python	Ecriture mathématique	Résultat
`somme(oppose(2),3)`		
`oppose(somme(2,3))`		
`somme(produit(2,3),4)`		
`produit(2,somme(3,4))`		
	$-(3+4)$	
	$4+(-3)$	
	$(5+2)\times 3$	
	$3\times 5+3\times 7$	

IV. TP n°4 : Structure conditionnelle

1) Premiers exemples

a) Statut d'un candidat au Bac

On rappelle qu'à l'issue des épreuves du Bac, un candidat est :

- admis si sa moyenne générale est supérieure ou égale à 10
- convoqué à l'oral de rattrapage si sa moyenne générale est inférieure à 10 mais supérieure ou égale à 8
- recalé si sa moyenne générale est inférieure à 8

On a écrit ci-dessous une fonction Python incomplète qui prend en entrée la moyenne générale à l'examen d'un candidat au Bac et renvoie son statut : « *admis* », « *oral* », « *recalé* ».

Code Python

```
def Bac(m):
    if ..................:
        return "admis"
    if ............... and ...............:
        return "oral"
    if ..................... :
        return "recalé"

```

1) Compléter les pointillés.

2) Que renvoie la console lorsqu'on saisit `Bac(11.5)` ? .....

3) Que renvoie la console lorsqu'on saisit `Bac(8.2)` ? .....

4) Que renvoie la console lorsqu'on saisit `Bac(7.8)` ? .....

Il est possible d'écrire une autre fonction, avec des instructions un peu différentes, mais qui renvoie les mêmes résultats :

Code Python

```python
def Bac2(m):
    if m>=10:
        return "admis"
    else:
        if m>=8:
            return "oral"
        else:
            return "recalé"

```

5) Que signifie le mot réservé **else** ? .....

6) Après le premier **else**, on se place dans le cas où m est ..... ?

7) Après le second **else**, on se place dans le cas où m est ..... ?

2) Facture

Le but du TP est d'aider un vigneron du Beaujolais à programmer une facturation. Connaissant le prix d'une bouteille, le nombre de bouteilles achetées, les éventuelles remises effectuées en fonction du nombre de bouteilles achetées, le programme doit renvoyer le montant total à payer pour le client.

Données :

- Le vigneron met à la vente deux types de bouteilles : celles de la cuvée « *Automne* » à 9,20 € la bouteille, et celles de la cuvée « *Prestige* » à 13,85 € la bouteille.
- A partir de 12 bouteilles achetées, et ce quel que soit la cuvée, le vigneron effectue une remise de 10% sur le montant total de la facture.

1) Un client achète 5 bouteilles de la cuvée « *Automne* ». Quel est le montant de la facture ?

.....

2) Un client achète 7 bouteilles de la cuvée « *Prestige* ». Quel est le montant de la facture ?

.....

3) Un client achète 3 bouteilles de la cuvée « *Automne* » et 6 bouteilles de la cuvée « *Prestige* ». Quel est le montant de la facture ?

.....

4) Un client achète 6 bouteilles de la cuvée « *Automne* » et 10 bouteilles de la cuvée « *Prestige* ». Quel est le montant de la facture ?

.....

5) Ecrire une fonction Python qui prend en entrées le nombre de bouteilles achetées de la cuvée « *Automne* » et le nombre de bouteilles achetées de la cuvée « *Prestige* », et qui renvoie le montant de la facture.

On contrôlera que la fonction est correcte en vérifiant les résultats qu'elle propose sur les 4 premières questions.

Montrer son code au professeur.

V. TP n°5 : Boucle bornée

1) Premiers exemples

a) Répéter des instructions

Il est possible, grâce à un langage de programmation, de répéter un bloc d'instructions un nombre donné de fois. Pour Python, c'est la boucle for qui permet cette répétition. Il est pénible de répéter la même action. Soit la citation suivante donnée à copier 5 fois par un professeur avec Python :
ńLes Mathématiques sont la reine des Sciencesż

1) Première manière de faire :

Code Python
```
def citation():
    print "Les Mathématiques sont la reine des Sciences"
    print "Les Mathématiques sont la reine des Sciences"
    print "Les Mathématiques sont la reine des Sciences"
    print "Les Mathématiques sont la reine des Sciences"
    print "Les Mathématiques sont la reine des Sciences"
```

2) Autre manière de faire :

Code Python
```
def citation2():
    for k in range(5):
        print "Les Mathématiques sont la reine des Sciences"
```

On rappelle que la fonction range(5) contient les entiers : 0, 1, 2, 3, et 4.

L'instruction **for** est une instruction composée, c'est-à-dire une instruction dont l'en-tête se termine par deux-points « : » , suivie d'un bloc ..... qui constitue ..... de la boucle (dans notre exemple, il ne contient qu'une seule instruction).

On dit que l'on réalise une ..... de la boucle à chaque fois que le corps de la boucle est exécuté.

Dans l'en-tête de la boucle, on précise après le mot-clé **for** le nom d'une variable (k dans l'exemple ci-dessus) qui prendra successivement toutes les valeurs qui sont données après le mot-clé **in**. On dit souvent que cette variable (ici k) est un ..... (ou variable d'itération) car elle sert à numéroter les itérations de la boucle.

3) Si l'on demandait d'écrire n fois la citation (n entier naturel non nul), comment modifier très simplement la fonction précédente pour qu'elle écrive effectivement n fois la phrase si elle prend n en entrée. Compléter les pointillés de la fonction citation3.

Code Python
```
def citation3(...........):
    for k in range(...........):
        print "Les Mathématiques sont la reine des Sciences"
```

b) Tables de multiplication

1) Compléter,sans calculatrice, les résultats des multiplications suivantes :

$9 \times 0 =$ ..... $9 \times 1 =$ ..... $9 \times 2 =$ ..... $9 \times 3 =$ ..... $9 \times 4 =$ ..... $9 \times 5 =$ .....

$9 \times 6 =$ ..... $9 \times 7 =$ ..... $9 \times 8 =$ ..... $9 \times 9 =$ ..... $9 \times 10 =$ .....

2) Identifier ce qui reste fixe et ce qui varie dans ces multiplications. .....

3) Combien de multiplications ont été écrites ? .....

4) Compléter les pointillés de la fonction Python suivante pour qu'elle affiche les multiplications précédentes. Attention, contrairement aux exemples précédents, chaque produit dépend du compteur k.

Code Python

```
def table9():
    for k in range(.......):
        print ("9x",.........,"=",.............)

```

5) Modifier la fonction précédente en écrivant une fonction table pour qu'elle affiche les résultats des multiplications $n \times 0, n \times 1$, ..., $n \times 9$ lorsqu'elle prend en entrée l'entier naturel n.

2) Premiers pas avec la tortue



Dans Python, on peut importer des modules qui contiennent un grand nombre de fonctions prédéfinies permettant de calculer, de dessiner, de simuler des expériences aléatoires. Nous découvrons le module turtle. Le principe est simple. Une petite tortue peut se déplacer en répondant à des ordres du type :

- avancer ou reculer d'une distance donnée,
- tourner sur place d'un angle donné, vers la droite ou la gauche,
- se déplacer vers un point donné par ses coordonnées cartésiennes,
- abaisser ou lever son stylo afin de laisser une trace, ou non
- à cela s'ajoute la possibilité de modifier l'épaisseur du trait, la couleur de tracé, de remplir un polygone fermé par une couleur

a) Memento turtle

Mouvements

Action précise	Fonction turtle
Faire avancer la tortue de n pixels (en ligne droite)	`forward(n)`
Faire reculer la tortue de n pixels (en ligne droite)	`backward(n)`
Faire tourner la tortue sur elle-même vers la gauche d'un angle de a degrés	`left(a)`
Faire tourner la tortue sur elle-même vers la droite d'un angle de a degrés	`right(a)`
Déplacer la tortue vers le point de coordonnées (x, y)	`goto(x,y)`

Tracés

Action précise	Fonction turtle
Lever le crayon pour que la tortue ne laisse pas de trace	`up()`
Baisser le crayon pour que la tortue laisse une trace à nouveau	`down()`
Déterminer la couleur du tracé	`color('red')`
Choisir l'épaisseur n du tracé (n est compté en nombre de pixels)	`width(n)`

Remplir un contour fermé à l'aide d'une couleur

Choisir une couleur de remplissage	`color('blue')`
Encadrer le script du contour par	`begin_fill()` et `end_fill()`

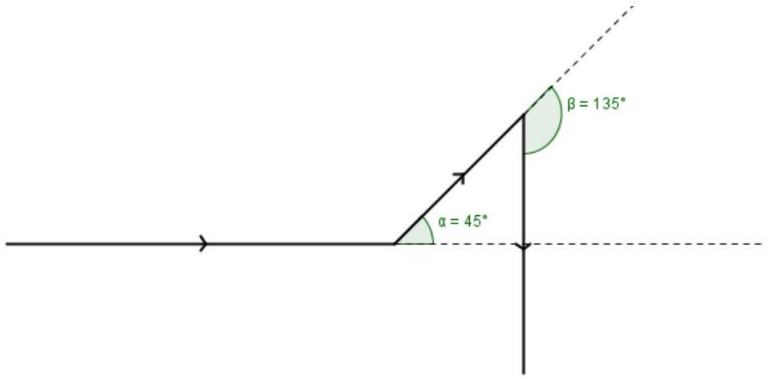
b) Exemples

1) Ce programme :

```python
from turtle import *

#importe toutes les fonctions du module turtle
forward(60)
left(45)
forward(28)
right(135)
forward(40)
mainloop()
#permet d'éviter que la fenêtre graphique ne se ferme toute seule à la fin du programme
```

fait déplacer la tortue de la manière suivante :



2) Associer chaque fonction de commande de tracé au bon tracé de la tortue.

```python
def trace1():
    for k in range(4):
        forward(100)
        left(90)
```

```python
def trace2():
    for k in range(4):
        forward(100)
    left(90)
```



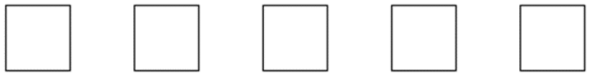
c) Exercices

1) Ecrire une fonction Python qui fait dessiner à la tortue un triangle équilatéral de côté 100 pixels.

2) Ecrire une fonction Python qui fait dessiner à la tortue un carré de côté c pixels, le nombre c de pixels étant pris en entrée.

3) Ecrire deux fonctions Python qui dessinent les frises suivantes :

a) Le côté de chaque carré mesure 50 pixels.



b) Le côté de chaque segment mesure 40 pixels.



VI. TP n°6 : Boucle non bornée

À partir de quelle année l'argent déposé sur un livret rémunéré à 3% par an aura doublé ?
Au bout de combien de temps une population de bactéries aura triplé si sa taille augmente de 10% par mois ?
Si l'épaisseur d'un glacier baisse de 4% par an, dans combien d'années son épaisseur aura-t-elle diminué de moitié ?
Voici des situations où l'on cherche une date représentant un seuil. Le but du TP est de programmer la recherche du seuil.

1) Premier exemple

Exemple : La population d'un pays donné est de 50 millions d'habitants. On admet qu'elle augmente de 5% par an.
On aimerait savoir dans combien d'années la population de ce pays va dépasser 65 millions d'habitants.

Modélisation mathématique

1) Combien le pays comptera d'habitants l'année suivante ? ..... dans deux ans ? .....

2) Justifier la phrase : « *Augmenter de* 5% *revient à multiplier par 1,05* ».

.....

Programme Dans l'éditeur, on a écrit la fonction suivante :

Code Python

```
def annee_seuil():
    n=0
    P=50
    while P<=65:
        n=n+1
        P=1.05*P
    return n
```

Dans l'exécuteur, on obtient :

Début de la Console Python

```
>>> annee_seuil()
6
```

Fin de la Console Python

Expliquer pourquoi cette fonction répond au problème en complétant le tableau d'états des variables suivant :

n	P	P<100
0	50	vrai

2) Application

A vous de jouer Vous déposez 1 000 € sur un livret rémunéré à 3% par an. A partir de quelle année l'argent déposé aura-t-il doublé ?