Lycée Raymond Queneau – 9-10 novembre 2017

Utilisation de la bibliothèque graphique `turtle`

La bibliothèque (ou module) `turtle` regroupe des fonctions permettant de créer des dessins en faisant bouger une tortue (ou plutôt une flèche) sur l'écran. Cette flèche fonctionne comme un crayon qui s'abaisse pour écrire (`down()`) ou se lève (`up()`), qui se déplace vers l'avant (`foward()`) ou vers l'arrière (`backward()`), qui tourne vers la gauche (`left()`) ou vers la droite (`right()`). On trouvera dans le Memento une liste (non exhaustive mais suffisante) des instructions pour travailler avec des élèves au lycée.

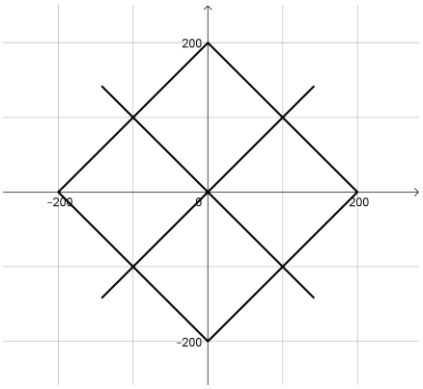
En tête des programmes de ces exercices devra être inscrite l'instruction d'importation de ce module :

```
from turtle import *
```

Pour afficher les graphiques sur l'écran, la dernière instruction du programme sera toujours `mainloop()`.

Exercice 1 Un premier dessin pour comprendre

Recopier le programme ci-contre et l'exécuter. La figure qui se trace doit ressembler au dessin suivant.




```
from turtle import *
from math import *

up()
goto(200,0)
left(90+45)
down()

for k in range(4):
    forward(200*sqrt(2))
    left(90)

up()
goto(0,0)
for k in range(4):
    down()
    forward(200)
    up()
    goto(0,0)
    left(90)

mainloop()
```

Le graphique est muni d'un repère invisible, orthonormé et centré. La fenêtre graphique mesure par défaut 800 pixels sur 600 pixels.

La tortue-crayon possède une direction (elle aussi invisible) et des coordonnées. En début de programme, ses coordonnées sont $(0\,;0)$ et sa direction vaut 0, c'est-à-dire qu'elle se déplace vers l'Est. Les valeurs prises par les coordonnées sont mesurées en pixels, la direction est un angle en degrés se référant au cercle trigonométrique orienté (Nord$\equiv 90\ (360)$, Ouest$\equiv 180\ (360)$, …).

En s'aidant du Memento, relire les instructions une par une et les relier à chaque étape de construction.

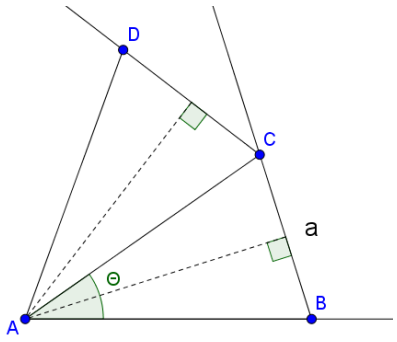
Exercice 2 Polygones réguliers

Géométrie

1°) La fonction $carre(a)$ ci-dessous permet de tracer un carré dont les côtés mesurent a pixels. On remarquera qu'elle ne possède pas d'instruction `return()`, puisqu'elle n'a pas de valeur à retourner.
En s'en inspirant, construire une fonction qui trace un triangle équilatéral dont les côtés mesurent a pixels, puis concevoir quelques lignes pour dessiner des triangles de ce type.

```
from turtle import *

def carre(a):
    for k in range(4):
        left(90)
        forward(a)

carre(100)

up()
goto(0,100)
down()
for k in range(1,11):
    carre(10*k)

mainloop()
```

Il s'agit maintenant de construire un polygone régulier à n côtés. La figure ci-contre représente le début de construction d'une telle figure. On appelle θ l'angle $\widehat{BAC}$ et a la longueur d'un côté.



2°) Compléter le programme ci-dessous pour qu'il construise un polygone régulier à n côtés de longueur $a = 200$ pixels. Le tracé se fait à partir du point B, puis en tournant dans le sens trigonométrique.
Attention : `turtle` utilise des angles en degrés et les fonctions trigonométriques de `math` des angles en radians.

```
from turtle import *
from math import *

t=360/n
left(…)
for k in range(n):
    forward(…)
    left(…)

mainloop()
```

3°) Adapter cette suite d'instructions pour les intégrer dans la fonction `polygone(n,a)` qui dessine un polygone régulier à n côtés de longueur a. Utiliser cette fonction pour effectuer des tracés.

```
def polygone(n,a):
    …
    …

# utilisation de la fonction polygone(n,a)
polygone(8,50)
polygone(7,100)
```

Exercice 3 Construire un triangle à partir des formules d'Al-Kashi

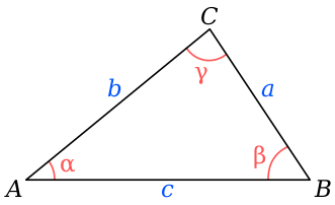
Géométrie

En respectant les notations de la figure ci-contre, les formules d'Al-Kashi sont données par :

$$a^2 = b^2 + c^2 - 2bc \times \cos(\alpha)$$

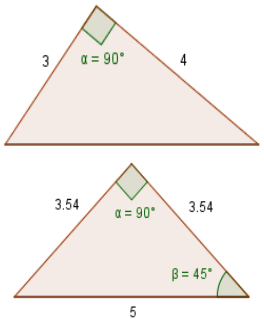
$$b^2 = a^2 + c^2 - 2ac \times \cos(\beta)$$

$$c^2 = a^2 + b^2 - 2ab \times \cos(\gamma)$$



L'objectif de l'exercice est de construire un triangle ABC dont on connait les trois longueurs. A partir d'un point A placé de manière aléatoire dans le rectangle $[-200; 0] \times [-200; 0]$, on parcourt le triangle en passant par les sommets B, puis C, puis de nouveau A. Le point B est placé à droite de A, à la même hauteur, comme sur la figure ci-dessus.

Ce programme nécessite l'utilisation des modules `turtle`, `math` et `random`.



1°) Ecrire la fonction $angle(a, b, c)$ qui retourne la valeur en degrés de l'angle opposé au côté de longueur a (c'est-à-dire l'angle α, si on se réfère à la figure). Vérifier la fonction pour des triangles connus.

2°) Ecrire les trois instructions demandant les longueurs des trois segments, puis calculer les angles $\hat{B}$ et $\hat{C}$.

```
#demande des longueurs
AB=int(input('longueur AB'))
BC=…
AC=…

#calcul des angles
angleB=…
angleC=…
```

3°) Inscrire en bas de programme l'instruction `mainloop()` qui permettra d'afficher la fenêtre graphique de `turtle`. Cette instruction sera toujours la dernière écrite en bas de programme.

4°) Pour tracer le triangle, compléter et exécuter les instructions ci-dessous.

`up()`	on relève le crayon
`goto(…,…)`	on se place au point A, point quelconque du plan (voir énoncé)
`down()`	on abaisse le crayon
`forward(…)`	on avance jusqu'au point B
`left(…)`	on oriente le crayon vers C puis on avance jusqu'au point C
…	
…	on oriente le crayon vers A puis on retourne au point A
…	

Exercice 4 Rantanplan (d'après Sesamath 2GT)

Géométrie

Averel est en train de manger avec Ma pendant que Joe, Jack et William se trouvent dans un champ et ne bougent pas. Ils forment donc un triangle. Rantanplan se trouve n'importe où dans le même champ (dans le triangle ou en dehors, peu importe).

Simultanément, les trois frères Dalton appellent Rantanplan. De manière équiprobable, le célèbre chien choisit un des trois lascars au hasard et se dirige vers lui. Arrivé à la moitié du chemin, il fait un trou et comme il a une mémoire de poisson rouge, il s'arrête car il ne sait plus où aller. Les trois frères l'appellent simultanément à nouveau, et à nouveau Rantanplan choisit un des trois loustics au hasard et se dirige vers lui. Arrivé à la moitié du chemin, il fait un trou et comme bla bla bla poisson rouge bla bla bla, il s'arrête. Et ainsi de suite un très grand nombre de fois.

Quelle figure les trous de Rantanplan réalisent-t-ils (vue du ciel) ?

Une simulation de la situation peut se faire grâce à `turtle`. Il faudra, dans ce programme, importer ce module ainsi que le module `random`.

1°) Ecrire la dernière ligne du programme : `mainloop()`.

2°) Rédiger la fonction $milieu(x1, y1, x2, y2)$ qui retourne les coordoonées du milieu du segment dont les extrémités ont pour coordonnées $(x1; y1)$ et $(x2; y2)$.

```
def milieu(x1,y1,x2,y2) :
    …
    return(…,…)
```

Dans un premier temps, les trois Dalton sont positionnés aux points $A(200; 200)$, $B(-200; 200)$, $C(0; -200)$ et Rantanplan se trouve au départ en $R(100; 100)$.

3°) Initialiser les variables correspondant aux coordonnées des trois Dalton et tracer le triangle formé par eux.

4°) Initialiser les variables xR et yR correspondant à la première position de Rantanplan. En ayant pris soin de lever le crayon, placer celui-ci sur la position occupée par Rantanplan sur le terrain.

5°) On commence par programmer un déplacement de Rantanplan. La trace qu'il laisse sur le sol lorsqu'il s'arrête sera matérialisée par un cercle de rayon 1 pixel. Compléter le programme proposé et le tester.

```
xA,yA=200,200
xB,yB…,…
xC,yC=…,…

up()
goto(xA,yA)
down()
goto(…,…)
…
```

```
choix=randint(…,…)
if choix==… : xR,yR=milieu(…,…,…,…)
elif choix==… : …
else : …

goto(xR,yR)
down()
circle(1)
up()
```

6°) Pour obtenir 10 déplacements du chien étourdi, il suffit d'inscrire les lignes de la question 5 dans une boucle bornée.

```
for k in range(10) :
    # recopier ici les instructions
```

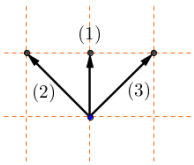
7°) Le tracé peut être long si l'on augmente le nombre de déplacements… Pour obtenir un tracé plus rapide, ajouter l'instruction `speed(0)` avant les premiers tracés. Puis augmenter le nombre de déplacements de Rantanplan.

8°) Modifier quelques instructions pour que les positions des trois Dalton et de Rantanplan au départ soient aléatoires. Par défaut, la fenêtre graphique a pour dimensions $[-400;400] \times [-300;300]$.

Exercice 5 Le pont entre deux rives (d'après IREM de Lorraine)

Probabilités conditionnelles

Après s'être longuement attardé à la brasserie « Les Deux Rives », M. Heine Ken décide de rentrer chez lui. Il doit pour cela emprunter un pont sans garde-corps de vingt pas de long et quatre pas de large. Il se trouve au milieu du pont, sur une rive, au début de la traversée.



La démarche de Ken est particulière : il avance d'un pas en avant (1) ou en avant en diagonale vers la gauche (2) ou en avant en diagonale vers la droite (3). On suppose que ces trois déplacements sont aléatoires et équiprobables.

On souhaite connaître la probabilité que Ken puisse traverser le pont.

La simulation graphique de plusieurs traversées du pont se fera avec le module `turtle` et `random`.

Les instructions correspondant à la mise en place du décor (les deux rives et le pont) sont regroupées dans la fonction $decor()$. On considère qu'un pas mesure 10 pixels. La rive Sud est matérialisée par la droite d'équation $y = 0$, la rive Nord par la droite $y = 150$ et les deux bords du pont sont des segments des droites $x = -25$ et $x = 25$.

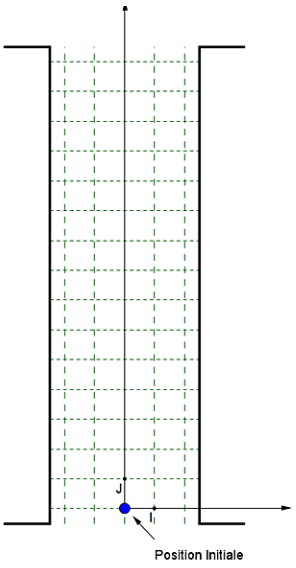
La construction étant répétitive (2 rives et 2 bords de ponts), deux formulations de cette fonction sont possibles.

formulation 1

```
def decor():
    up()
    goto(-400,0)
    down()
    goto(400,0)

    up()
    goto(-400,150)
    down()
    goto(400,150)

    up()
    goto(-25,0)
    down()
    goto(-25,150)

    up()
    goto(25,0)
    down()
    goto(25,150)
```

formulation 2

```
def decor():
    for y in [0,150]:
        up()
        goto(-400,y)
        down()
        goto(400,y)
    for x in [-25,25]:
        up()
        goto(x,0)
        down()
        goto(x,150)
```



1°) Recopier le programme ci-contre en utilisant une des deux formulations de la fonction $decor()$

Exécuter le programme pour visualiser la scène.

```
def decor() :
    …

decor()

mainloop()
```

La traversée du pont s'effectue en quinze pas maximum. La position de Ken est donnée par ses coordonnées $(x; y)$. Au départ, $(x; y) = (0; 0)$. Tant que Ken a fait moins de 15 pas $(y < 150)$ et qu'il est sur le pont $(-25 < x < 25)$, il peut faire un pas supplémentaire dans l'une des trois directions. On remarquera que lorsque Ken fait un pas en avant, le choix aléatoire de la direction modifie l'abscisse x, la valeur de y sera, dans tous les cas, augmentée de 10.

2°) Compléter les instructions suivantes qui simule une traversée du pont. Elles s'écrivent entre les instructions `decor()` et `mainloop()`.

```
up()
goto(0,0)
x,y=0,0
down()
while …<… and …<…<…:
    a=randint(1,3)
    if a==1: x=…
    elif a==3: x=…
    y=…
    goto(x,y)
```

A la fin de l'exécution de la boucle non bornée, on peut savoir si Ken a traversé le pont ou est tombé à l'eau. En effet, à ce moment, si y vaut 150, on est assuré que Ken est sur l'autre rive (Si Ken est tombé à l'eau pendant la traversée, la boucle s'est arrêtée parce que $x \notin]-25; 25[$ et y est inférieur à 150).

3°) Ajouter et compléter les instructions suivantes après la boucle non bornée pour savoir si Ken a traversé le pont.

```
if …:
    print('Ken a traversé')
else:
    print('Ken est tombé')
```

On peut à présent simuler plusieurs traversées et compter le nombre de fois où Ken a réussi la traversée. On répète donc toutes les instructions correspondant à une traversée dans une boucle bornée. On utilise une variable *succes* qui sera initialisée à 0 avant la première traversée (soit avant la première ligne de la boucle non bornée) et qui sera incrémenté d'une unité pour chaque traversée réussie.

4°) Modifier le programme comme ci-dessous et ajouter l'instruction incrémentant la variable *succes* en cas de traversée réussie. L'instruction `mainloop()` doit rester la dernière ligne du programme.

```
decor()

speed(0)    # pour accélérer les tracés
N=100       # nombre de traversées
succes=0
for k in range(N):
    # insérer ici les lignes
    # correspondant à une traversée :
    # lignes allant de up()
    # à print('Ken est tombé')

print('nombre de traversées réussies : ',succes)
print('frequence :',succes/N)

mainloop()
```