Démonstration # 5

Exercice 1

La fonction ci-dessous est une fonction de test pour la fonction **myst**. À l'aide de cette fonction, déduisez la spécification de la fonction **myst** et écrivez une définition de cette fonction.

```
def testMyst():
    assert myst(11,42,13) == 42
    assert myst(-6,-5,13) == 13
    assert myst(-6,-5, 3) == -6
    assert myst(-6,-5, 6) == 6
```

Exercice 2

On cherche à écrire une fonction pour obtenir de l'usager un nombre dans un certain intervalle. Si le nombre entré n'est pas dans l'intervalle en question, il faut continuer à demander à l'usager d'entrer un nombre dans cet intervalle jusqu'à ce qu'un nombre valide soit entré.

Faire la conception de cette fonction en répondant aux 5 questions de conception de fonction vues en cours.

Exercice 3

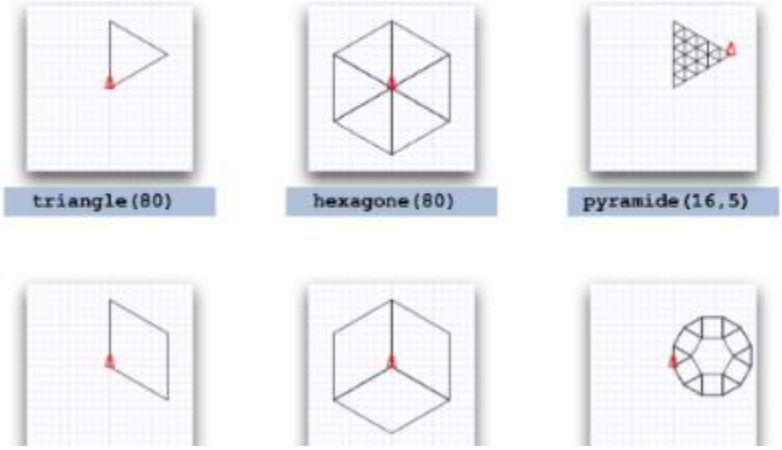
Voici la fonction **polygoneReg** qui permet de dessiner des polygones réguliers :

```
def polygoneReg(cote, nbCotes):
    for _ in range(nbCotes):
        fd(cote)
        rt(360/nbCotes)
```

Vous devez utiliser cette fonction, et possiblement définir d'autres fonctions, pour définir des fonctions qui dessinent les 6 figures géométriques suivantes :

- **triangle(cote)** : un triangle équilatéral ayant des côtés de longueur cote
- **losange(cote)** : un losange ayant des côtés de longueur cote et des angles de 60 et 120 degrés
- **hexagone(cote)** : un hexagone formé de 6 triangles équilatéraux ayant des côtés de longueur cote
- **cube(cote)** : un cube "3D" ayant des côtés de longueur cote
- **pyramide(cote,n)** : une pyramide formée de n couches de triangles équilatéraux
- **bague(cote)** : une bague formée de 6 triangles équilatéraux et 6 cubes qui s'alternent et qui ont des côtés de longueur cote

Voici le résultat de ces fonctions (si on a fait **`lt(90)`** au préalable pour pointer la tortue vers le haut) :

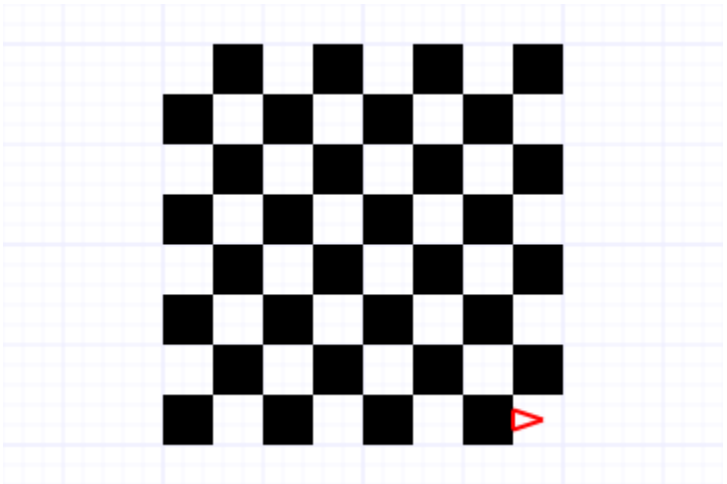


Exercice 4

1. Comparez les fonctions **`triangle`** et **`polygoneReg`**. Est-ce que l'une est plus générale que l'autre. Si oui laquelle?
2. Comparez les fonctions **`losange`** et **`polygoneReg`**. Est-ce que l'une est plus générale que l'autre. Si oui laquelle?
3. Comparez les fonctions **`triangle`** et **`losange`**. Est-ce que l'une est plus générale que l'autre. Si oui laquelle?

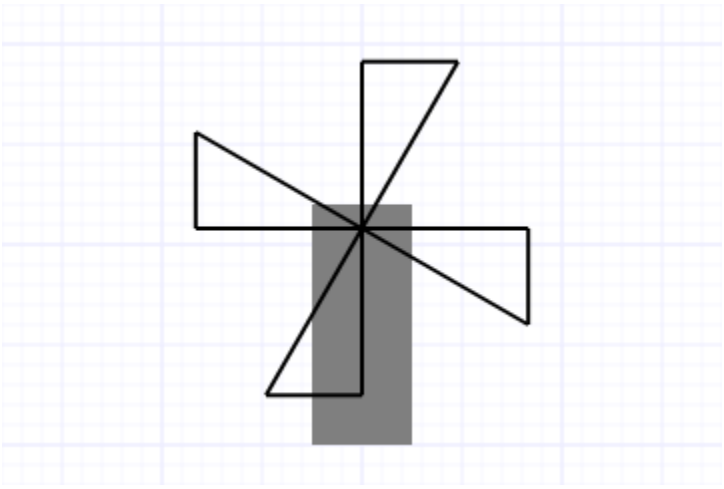
Exercice 5

Écrivez un programme qui affiche un échiquier **`nxn`** centré sur la fenêtre de dessin.



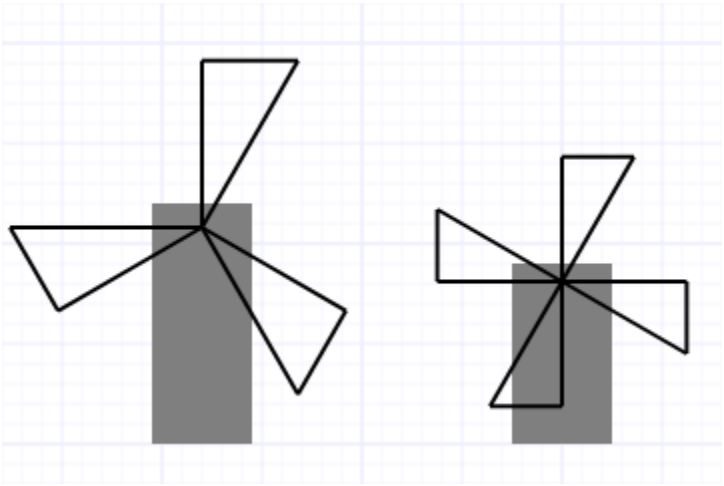
Exercice 6

Écrivez un programme qui affiche l'image du moulin à vent ci-dessous. Modifiez-le pour que le moulin soit animé avec une rotation de ses ailes.



Exercice 7

Modifiez le programme précédent pour qu'il fasse l'animation de deux moulins de taille différente qui tournent à des vitesses différentes.



Démonstration #5 Solution IFT1015

Exercice 1

```
# Solution

# La fonction maxabs prend 2 nombres en paramètres et retourne le
# nombre qui a la plus grande valeur absolue.  Si les deux paramètres
# ont la même valeur absolue, alors c'est le paramètre positif qui est
# retourné.

def maxabs(x, y):
    if abs(x) == abs(y):
        if x > y:
            return x
        else:
            return y
    elif abs(x) > abs(y):
        return x
    else:
        return y

# La fonction myst prend 3 nombres en paramètres et retourne le nombre
# qui a la plus grande valeur absolue.  Si deux paramètres ont la même
# valeur absolue, alors c'est le paramètre positif qui est retourné.

def myst(a, b, c):
    return maxabs(a, maxabs(b, c))
```

Exercice 2

```
# Solution

Q1 : Quelle est la tâche de la fonction?

La fonction obtient un nombre de l'usager dans un certain intervalle
précisé par une borne inférieure et une borne supérieure.  Une question
est posée répétitivement tant qu'un nombre valide n'est pas entré par
l'usager.

Q2 : Quels sont les paramètres de la fonction?

La fonction prend 3 paramètres : un texte qui est la question posée à
l'usager, et 2 nombres flottants, la borne inférieure et la borne
supérieure de l'intervalle de nombres acceptables.  Ces bornes sont
inclusives.

Q3 : Quels est le type du résultat?

Le résultat est un nombre flottant.
```

```
Q4 : Quel est le meilleur nom pour la fonction?

promptFlottantIntervalle

Q5 : Quels sont des exemples d'utilisation de la fonction?

taux = promptFlottantIntervalle("pourcentage de solde?", 0.0, 100.0)

prix = promptFlottantIntervalle("prix de l'article?", 0.0, float("inf"))


Codage de la fonction :

# La fonction promptFlottantIntervalle obtient un nombre de l'usager
# dans un certain intervalle précisé par une borne inférieure et une
# borne supérieure.  La fonction prend 3 paramètres : un texte qui est
# la question posée à l'usager, et 2 nombres flottants, la borne
# inférieure et la borne supérieure de l'intervalle de nombres
# acceptables.  Ces bornes sont inclusives.

def promptFlottantIntervalle(question, borneInf, borneSup):
    while True:
        nombre = float(prompt(question))
        question = ('Un nombre de ' + str(borneInf) + ' à ' +
                    str(borneSup) + ' SVP.')
        if nombre >= borneInf and nombre <= borneSup:
            return nombre
```

Exercice 3

```
# Solution

def polygoneReg(cote, nbCotes):
    for _ in range(nbCotes):
        fd(cote)
        rt(360/nbCotes)

def triangle(cote):
    polygoneReg(cote, 3)

def losange(cote):
    for _ in range(2):
        fd(cote)
        rt(120)
        fd(cote)
        rt(60)

def hexagone(cote):
    for _ in range(6):
        triangle(cote)
        rt(60)

def cube(cote):
    for _ in range(3):
```

```
        losange(cote)
        rt(120)

def pyramide(cote, n):
    pu()
    for couche in range(n, 0, -1):
        for _ in range(couche):
            pd(); triangle(cote); pu()
            fd(cote)
        bk(cote*couche); rt(60); fd(cote); lt(60)
    pd()

def carre(cote):
    polygoneReg(cote, 4)

def bague(cote):
    for i in range(12):
        if i%2 == 0:
            triangle(cote)
        else:
            carre(cote)
        pu(); fd(cote); pd()
        rt(30)

lt(90)
#triangle(80)
#losange(80)
#hexagone(80)
#cube(80)
#pyramide(16, 5)
#bague(25)
```

Exercice 4

```
# Solution

1. La fonction polygoneReg est plus générale que triangle car
avec polygoneReg on peut faire le même dessing que triangle, mais
l'inverse n'est pas vrai.

2. Les fonctions polygoneReg et losange ne sont pas comparables.
On ne peut pas se servir de l'une des fonctions pour faire le
même travail que l'autre fonction.

3. Les fonctions triangle et losange ne sont pas comparables.
On ne peut pas se servir de l'une des fonctions pour faire le
même travail que l'autre fonction.
```

Exercice 5

```
# Solution

def echiquier(n, cote):
    pensize(cote)
```

```
    for i in range(n):
        for j in range(n):
            if i%2 != j%2:  # tuile noire?
                pu()
                # positionner à la bonne coordonnée
                # avec une tuile blanche en haut à gauche
                goto((j-n/2)*cote, -((i-n/2)*cote+cote/2))
                pd()
                fd(cote)

echiquier(8, 25)
```

Exercice 6

```
# Solution

def rad(degres):
    # convertir degrés en radians
    return degres*math.pi/180

def aile(longueur):
    # dessiner une aile du moulin à vent
    angle = 30
    fd(longueur*math.cos(rad(angle)))
    rt(90)
    fd(longueur*math.sin(rad(angle)))
    rt(90+angle)
    fd(longueur)
    rt(180-angle)

def ailes(longueur, nbAiles):
    # dessiner les ailes du moulin à vent
    for _ in range(nbAiles):
        aile(longueur)
        lt(360/nbAiles)

def moulin(hauteur, nbAiles, rotation):
    # dessiner un moulin à vent avec une certaine
    # rotation de ses ailes
    pd()
    pensize(50)
    pencolor(0.5, 0.5, 0.5) # gris
    fd(hauteur)
    pu()
    bk(hauteur*0.1)
    pensize(2)
    pencolor(0,0,0) # noir
    pd()
    rt(rotation)
    ailes(hauteur*0.8, nbAiles)
    rt(-rotation)
    pu()
    bk(hauteur*0.9)

def animation():
```

```
    rotation = 0
    while True:
        clear()
        pu(); lt(90); bk(100)
        moulin(120, 4, rotation)
        rotation += 0.5
        ht()
        sleep(0.02)

animation()
```

Exercice 7

```
# Solution

def animation():
    rotation1 = 0
    rotation2 = 0
    while True:
        clear(); pu()
        bk(80); lt(90); bk(100)
        moulin(120, 3, rotation1)
        rt(90); fd(180); lt(90)
        moulin(90, 4, rotation2)
        rotation1 += 5
        rotation2 += 0.5
        ht()
        sleep(0.02)

animation()
```