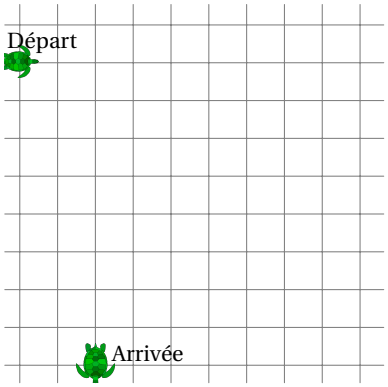
La tortue : Fiche professeur

I Trace ton chemin...

1. L'algorithme ci-dessous permet de déplacer une tortue :

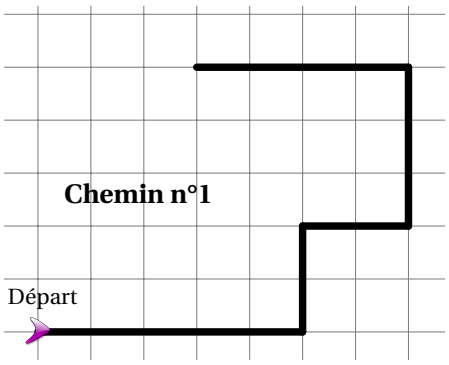
```
tortue.avance(7)
tortue.droite(90)
tortue.avance(5)
tortue.droite(90)
tortue.avance(5)
tortue.gauche(90)
tortue.avance(3)
```



Pour définir correctement les instructions « droite(90) » et « gauche(90) ».

Dessiner le chemin parcouru par la tortue.

2. Écrire un algorithme qui permet de déplacer la tortue comme dans la figure ci-dessous :

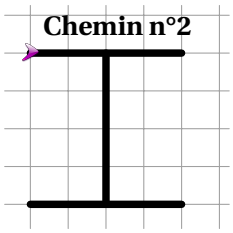


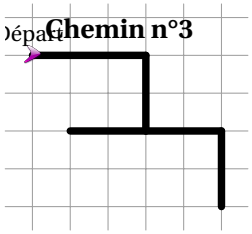
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

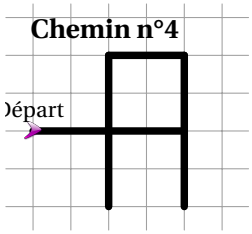
Permet de définir le travail futur à savoir : Écrire un algorithme permettant de retracer le chemin.

Ensuite, distribution des chemins n°2, 3 ou 4 (deux groupes côte à côte n'ont pas le même chemin) et des cartes codes vierges pour écrire l'algorithme correspondant (cf p.7 et p.10 du doc élève).

Par groupe, les élèves doivent écrire un algorithme sur la carte vierge permettant à un autre groupe de reproduire le chemin.
Il est préférable de **donner en double** le chemin à décrire afin que les quatre élèves d'un même groupe comment en même temps à proposer un algorithme sur leurs cahiers. Une fois le groupe d'accord, ils peuvent mettre au propre leur algorithme sur la carte vierge.



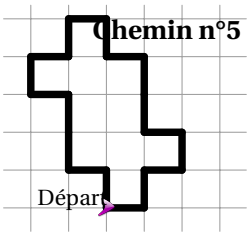


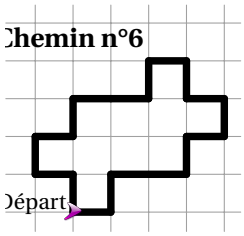


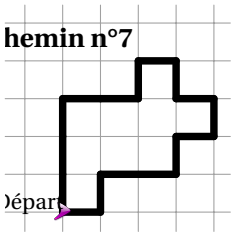
Chemin n°.....

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

Tous les groupes ne finissent pas en même temps ces premières cartes, on peut donc commencer à distribuer les chemins n°5, 6 ou 7 aux groupes qui ont fini de décrire leur chemin précédent.
Ces trois autres chemins vont poser un problème car il faut définir un nouveau mouvement pour que sa description rentre sur la carte d'instructions.
Pour les groupes qui ont l'idée de créer un nouveau mouvement, on peut leur proposer d'envoyer un élève le définir au tableau (ou pas, cela peut aussi être intéressant de montrer la nécessité de diffuser une définition...).
Réflexions d'élèves en vidéo : Vidéo : la tortue







Après avoir mis en place la nécessité de créer de nouveaux mouvements, on peut donner cette fiche de synthèse pour une mise en commun.

On peut définir un « zigzag » ou un autre mouvement créé par les élèves qui permet de décrire le chemin correspondant avec toutes les contraintes.

Une définition des procédures proche de celle de Python sera donner dans la partie II.

Un exemple avec ajout de parenthèses sur l'appel de la procédure. Sinon, les parenthèses peuvent attendre...

- - - - - - - - - - - - - - - - ✂ -

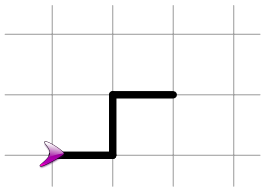
Il est possible d'inventer de nouvelles commandes pour déplacer la tortue !

Nom de la nouvelle commande :

...`zigzag()`...

```
...tortue.avance(1)...
...tortue.gauche(90)...
...tortue.avance(1)...
...tortue.droite(90)...
...tortue.avance(1)...
```

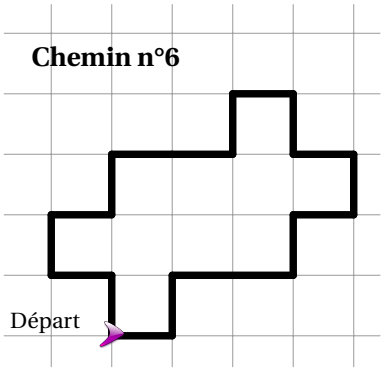
Dessin de la nouvelle commande :



Votre programme :

```
...zigzag()...
...zigzag()...
...tortue.gauche(90)...
...zigzag()...
...tortue.gauche(90)...
...zigzag()...
...zigzag()...
...tortue.gauche(90)...
...zigzag()...
```

Le résultat :



- - - - - - - - - - - - - - - - ✂ -

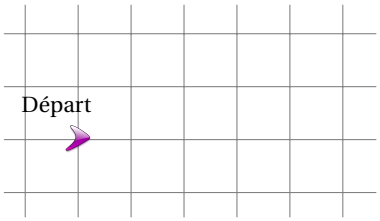
II Définir et appeler une procédure

Cette partie est consacrée à l'utilisation et la définition de nouvelles procédures sans paramètres.
On donne une syntaxe proche de celle de Python. une mise en commun s'impose pour détailler tous les élèments (def, parenthèses et peut-être l'indentation mais elle n'est pas encore fondamentale sur papier ici).

Une idée, pour alléger le code, est de créer de nouvelles **procédures** offrant à la tortue un nouveau déplacement. Voici ci-contre une façon de définir une **procédure** : « *zigzag* ».

```
def zigzag() :
    tortue.avance(1)
    tortue.gauche(90)
    tortue.avance(1)
    tortue.droite(90)
    tortue.avance(1)
```

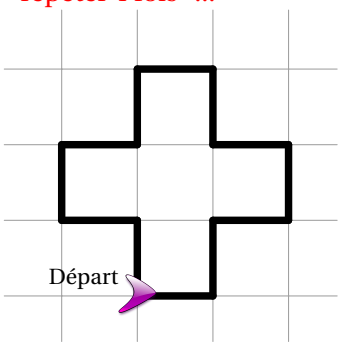
1. Que va dessiner la tortue avec l'algorithme ci-dessous ?




```
zigzag()
zigzag()
```

2. En utilisant la **procedure** **zigzag**(), écrire un algorithme permettant de dessiner la figure ci-dessous :

Pour cette figure, de nombreux élèves essayent de décrire un « répéter 4 fois »...

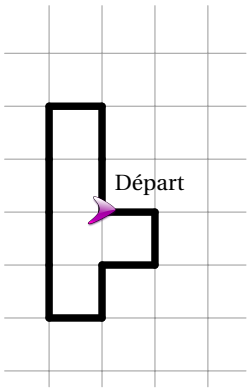


3. Définir une procédure permettant de réaliser le chemin ci-dessous en un minimum de lignes de codes :

On voit apparaître ici des noms de procédures comme : « `carretroiscotes()` » ou encore « `U()` »

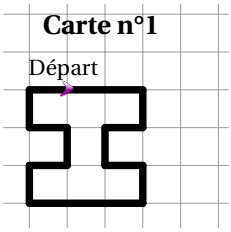
Votre procédure :

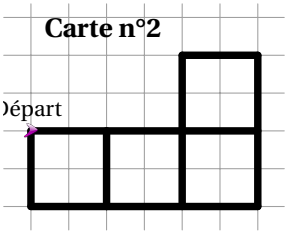
Votre programme :



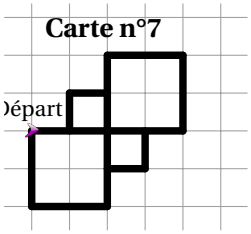
4. Demander une carte à votre professeur. Écrire un algorithme permettant de reproduire le dessin de la carte. Les **procédures** suivantes peuvent être utiles :

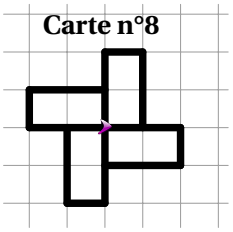
On distribue ici une des cartes n°1 à 8 et une carte d'instructions vierge (cf p.8 et p.10 du doc élève).
La consigne est la même que dans la partie I mais cette fois, ils peuvent utiliser les procédures ci-dessous car elles sont connues de tous ou alors en inventer de nouvelles.
Les cartes 6, 7 et 8 vont peut-être demander la création de nouvelles procédures...





.....






```
def demiTourGauche() :
    tortue.avance(1)
    tortue.gauche(90)
    tortue.avance(1)
    tortue.gauche(90)
    tortue.avance(1)
```

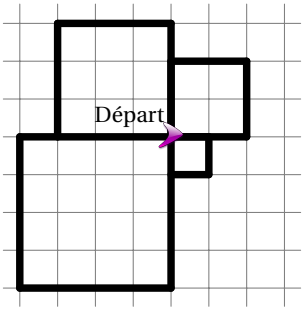
```
def demiTourDroite() :
    tortue.avance(1)
    tortue.droite(90)
    tortue.avance(1)
    tortue.droite(90)
    tortue.avance(1)
```

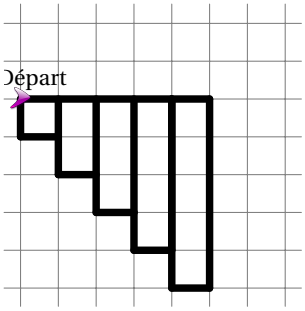
```
def doubleZigzag() :
    zigzag()
    tortue.gauche(90)
    tortue.avance(1)
    tortue.droite(90)
    tortue.avance(1)
```

```
def carre() :
    tortue.avance(2)
    tortue.droite(90)
    tortue.avance(2)
    tortue.droite(90)
    tortue.avance(2)
    tortue.droite(90)
    tortue.avance(2)
```

Vous pouvez aussi inventer vos procédures et venir les marquer au tableau !

À la fin de cette partie, avant d'introduire la possibilité d'insérer des paramètres dans les procédures, on donne à chaque groupe les figures suivantes (cf p.9 du doc élève) :





L'idée étant d'engendrer chez les élèves la nécessité d'introduire un paramètre dans les procédures.

III Procédure à paramètre

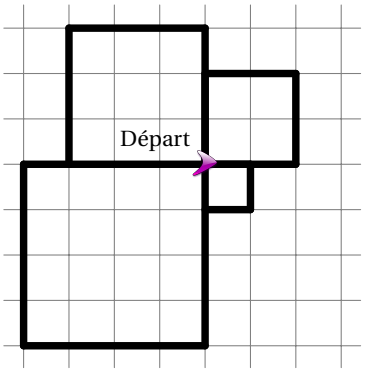
Trois questions pour s'entraîner et comprendre. La première question sert de mise en commun :

1. Nadia a écrit un algorithme permettant de dessiner la figure ci-dessous en utilisant une **procédure** **carre**(x). Compléter l'algorithme de Nadia.

```
def carre(x) :
    tortue.avance(x)
    tortue.droite(90)
    tortue.avance(x)
    tortue.droite(90)
    tortue.avance(x)
    tortue.droite(90)
    tortue.avance(x)
```

Le programme de Nadia :

```
carre(1)
.....
.....
.....
```

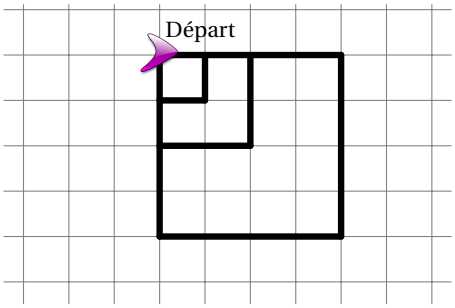


☞ La **procédure** **carre**(x) demande un **paramètre** x.

☞ Comme pour une fonction, on utilise cette **procédure** en remplaçant le **paramètre** x par une valeur.

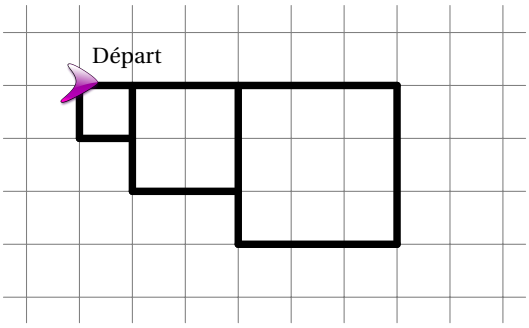
2. Écrire un algorithme qui permet de déplacer la tortue comme dans la figure ci-dessous :

```
.....
.....
.....
.....
.....
```

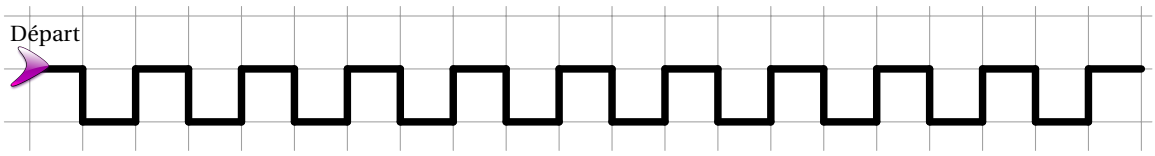


3. Même question avec la figure ci-dessous :

```
.....
.....
.....
.....
.....
.....
.....
```

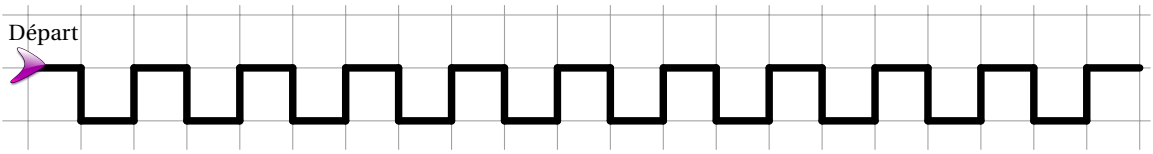


4. Écrire un algorithme qui permet de déplacer la tortue comme dans la figure ci-dessous :
Avant d'introduire les boucles, les élèves doivent réfléchir à la description de cette figure.
L'idée de boucle vient naturellement et tous les élèves ont créé une procédure « `trou()` » ou « `zagzig()` »...



IV Vers les boucles

1. Écrire un algorithme qui permet de déplacer la tortue comme dans la figure ci-dessous :
Mise en place de l'écriture d'une boucle en Python.
Le dernier (ou le premier) segment de la figure ne doit pas faire partie du contenu de la boucle.
On peut donc débattre de l'indentation lors de la mise en commun...




```
# (Répéter 10 fois :)
for i in range(10):
    ..................................
    ..................................
    ..................................
    ..................................
..........................................
..........................................
```

☞ « `Répéter 10 fois :` » n'existe pas en *Python*.

☞ On écrit : « `for i in range(10) :` »

☞ On peut le traduire par : « *Pour i allant de 0 à 9 faire :* »

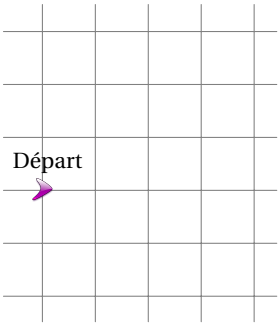
☞ Le **compteur** `i` parcourt les valeurs 0 ; 1 ; 2 ... jusqu'à 9.

☞ À la sortie de la **boucle** `for`, `i` aura la valeur 9.

2. Que va dessiner la tortue avec l'algorithme ci-dessous ?

```
tortue.avance(1)
for i in range(4):
    tortue.avance(2)
    tortue.gauche(90)
tortue.droite(90)
tortue.avance(1)
```

Travail sur les boucles et l'indentation.



La fin de l'**indentation** annonce la fin du contenu de la **boucle**.

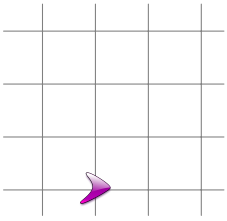
Voici deux **procédures** déjà utilisées :
Pour les questions qui vont suivre...

```
def zigzag() :
    tortue.avance(1)
    tortue.gauche(90)
    tortue.avance(1)
    tortue.droite(90)
    tortue.avance(1)
```

```
def carre(x) :
    tortue.avance(x)
    tortue.droite(90)
    tortue.avance(x)
    tortue.droite(90)
    tortue.avance(x)
    tortue.droite(90)
    tortue.avance(x)
```

3. Que va dessiner la tortue avec l'algorithme ci-dessous ?

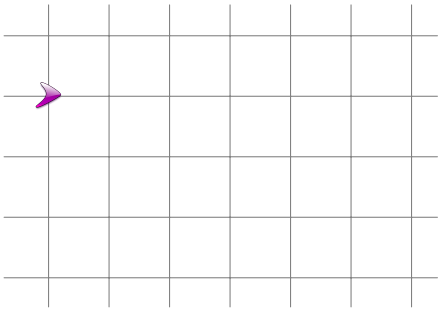
```
for i in range(4):
    zigzag()
    tortue.gauche(90)
```



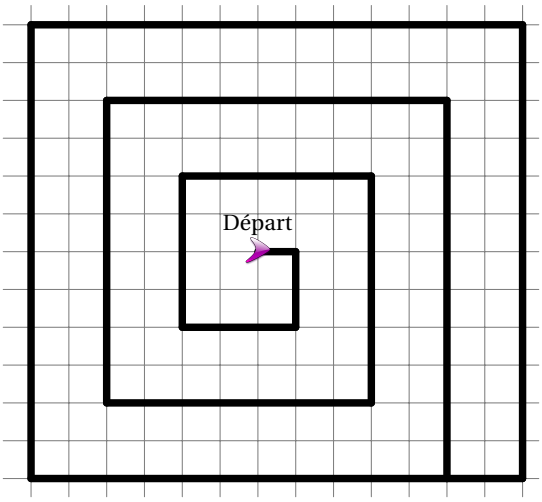
4. Que va dessiner la tortue avec l'algorithme ci-dessous ?

Mise en commun sur les bornes du compteur de boucle et possibilité d'utiliser ce compteur.

```
for i in range(1,4):
    carre(i)
    tortue.droite(90)
    tortue.avance(i)
```



5. Écrire un algorithme permettant de dessiner la figure ci-dessous :




```
.............................
.............................
.............................
.............................
.............................
.............................
.............................
```

V Sur machine

Passage sur machine et traduction des commandes avec un exemple de procédure.

1. Ouvrir *Pyzo* et tester le code suivant :

```
import turtle as tortue

tortue.reset()

#Vos procedures:
def zigzag():
    tortue.fd(10)
    tortue.left(90)
    tortue.fd(10)
    tortue.right(90)
    tortue.fd(10)

#Votre algorithme:
zigzag()
tortue.fd(10)
zigzag()

#Fenetre graphique:
tortue.mainloop()
```

Un exemple de code Python que les élèves peuvent copier (en ouvrant le document avec AdobeReader).

a) Que signifie la commande : `tortue.fd(10)` ?

.....

b) Que signifie la commande : `tortue.left(90)` ?

.....

c) Que signifie la commande : `tortue.right(90)` ?

.....

d) Que signifie la commande : `tortue.reset()` ?

.....

2. Écrire un programme permettant de dessiner les figures ci-dessous :

Les élèves peuvent réaliser ces figures dans le désordre et se servir des algorithmes vus en cours.

Figure n°1

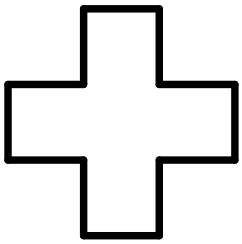


Figure n°2

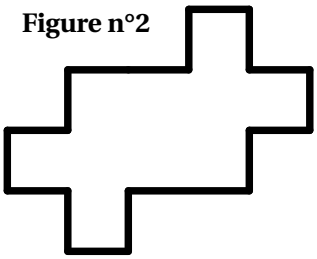


Figure n°3

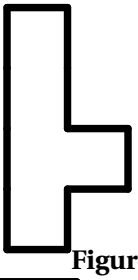


Figure n°4

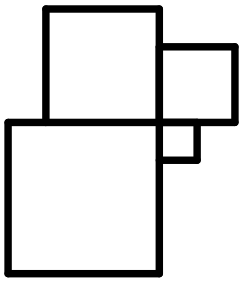


Figure n°5

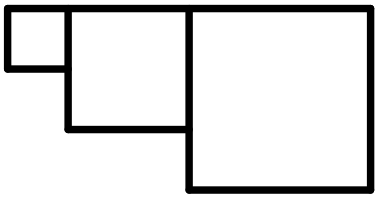


Figure n°6

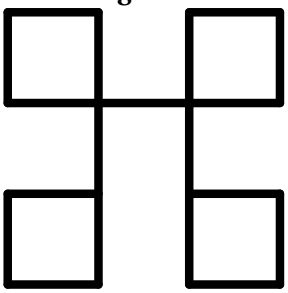


Figure n°7

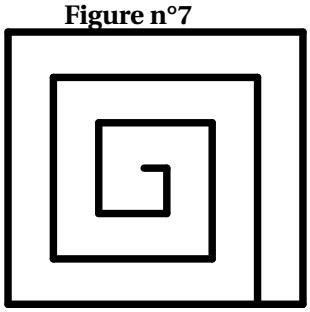
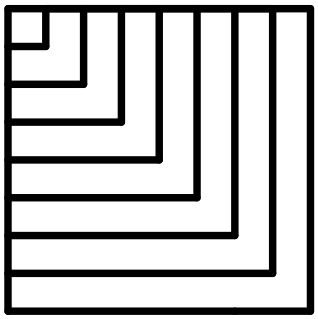


Figure n°8



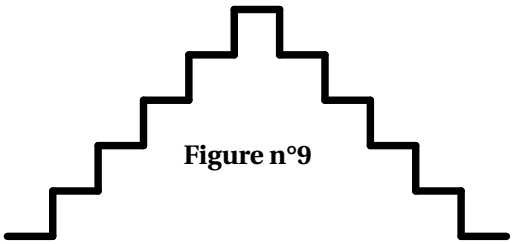
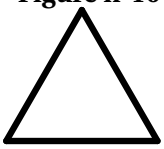


Figure n°9

Figure n°10



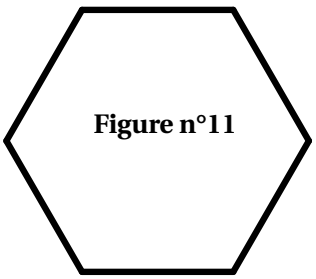


Figure n°11

Des questions se posent sur les angles et le fait de tourner.
Pour la couleur bleu, les élèves doivent effectuer des recherches sur internet.

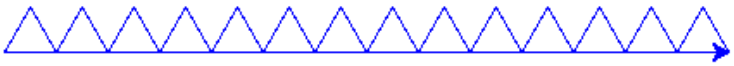


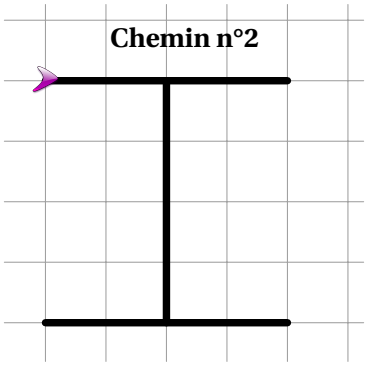
Figure n°12

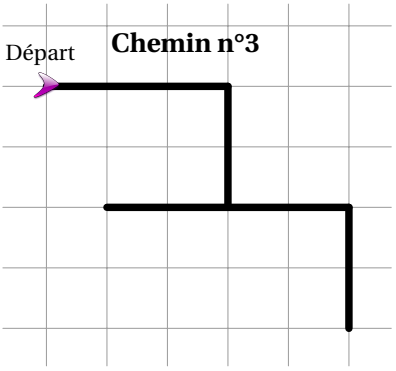
Un seul élève a réussi le dernier défi :

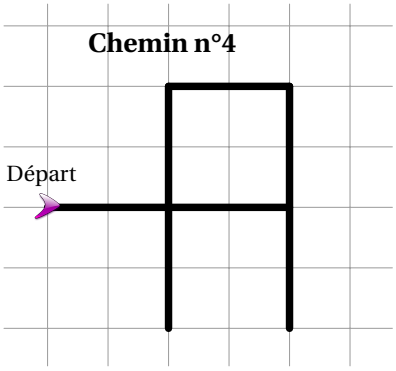
Écrire un programme qui demande un nombre puis qui dessine un polygone régulier avec ce nombre de côtés.

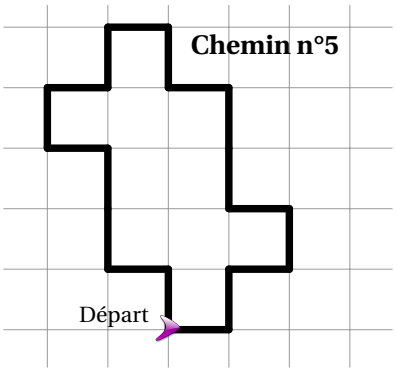
Voici une idée pour commencer :

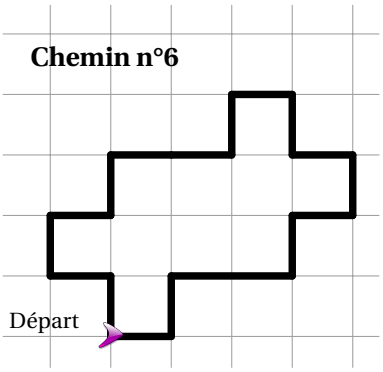
```
import turtle as tortue
import random
tortue.reset()

n=int(input("Donner le nombre de cotes : "))
```

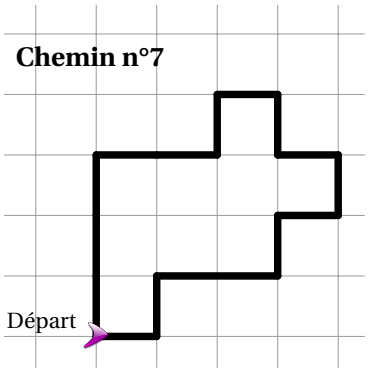

Chemin n°2

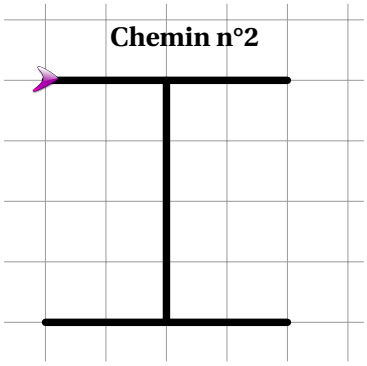

Départ
Chemin n°3

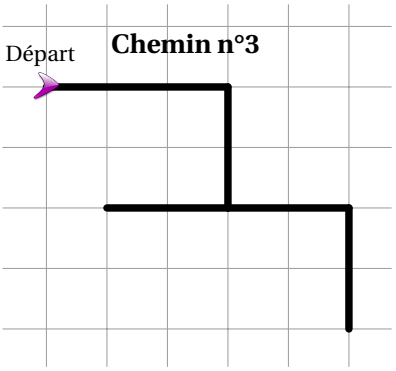

Chemin n°4
Départ

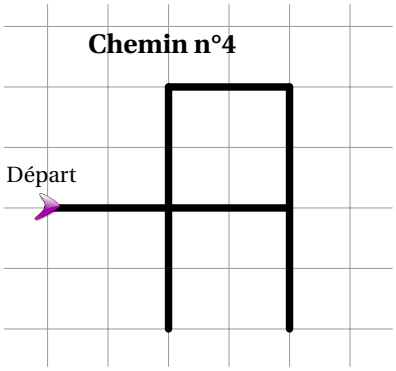

Chemin n°5
Départ

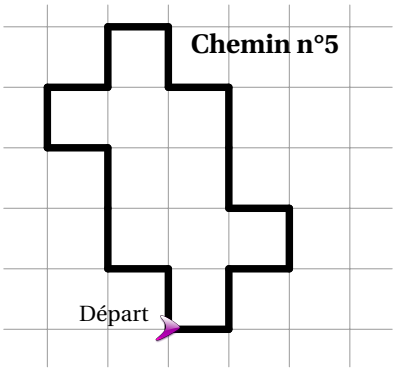

Chemin n°6
Départ

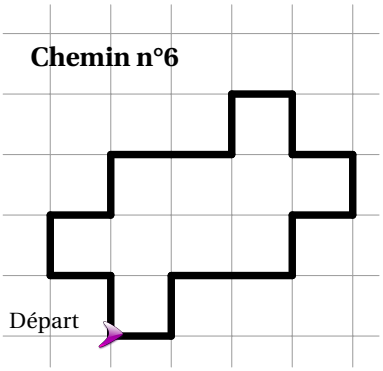

Chemin n°7
Départ

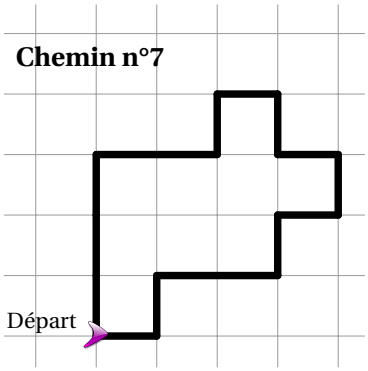

Chemin n°2

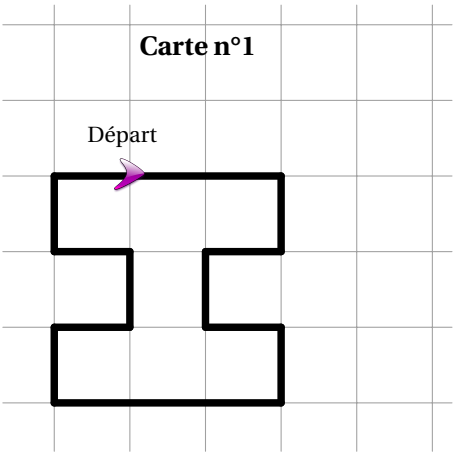

Départ
Chemin n°3

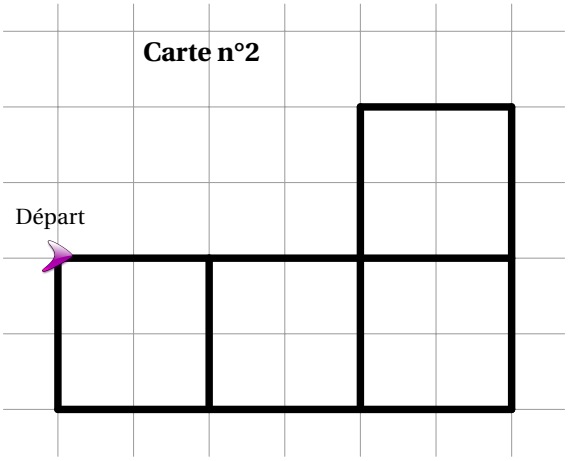

Chemin n°4
Départ

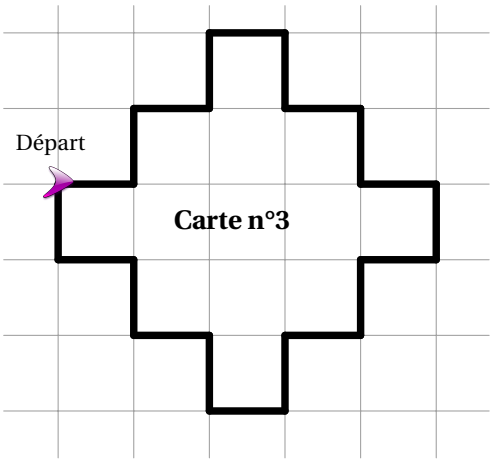

Chemin n°5
Départ

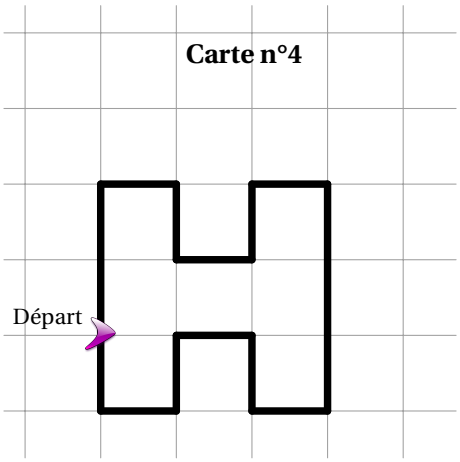

Chemin n°6
Départ

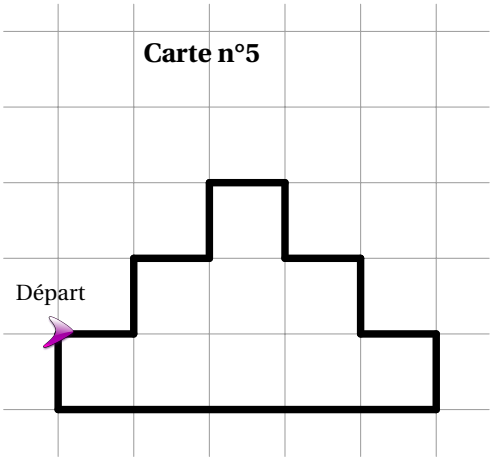

Chemin n°7
Départ

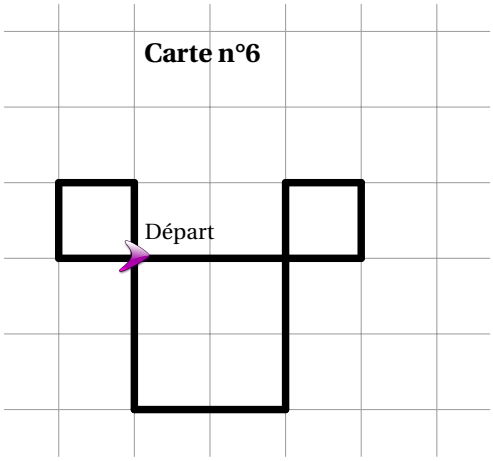

Carte n°1
Départ

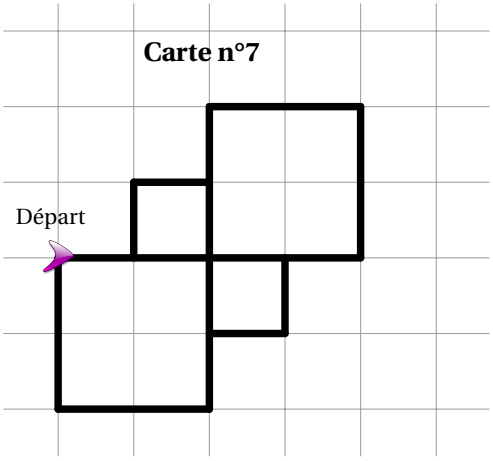

Carte n°2
Départ

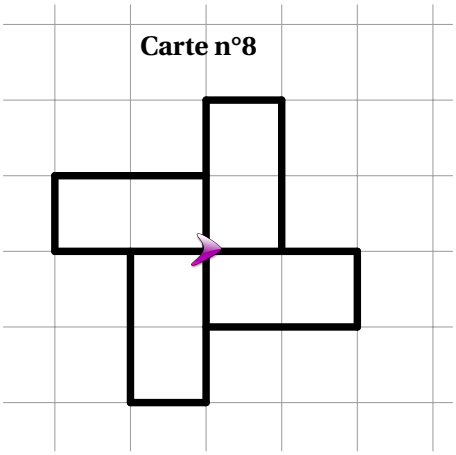

Départ
Carte n°3

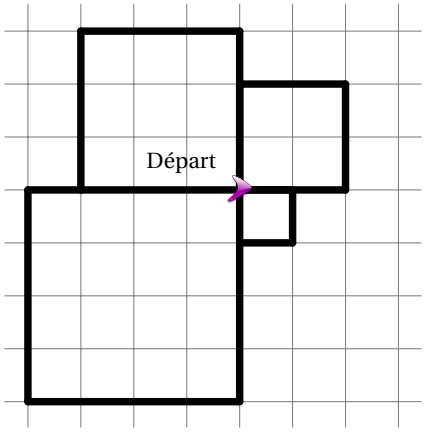

Carte n°4
Départ

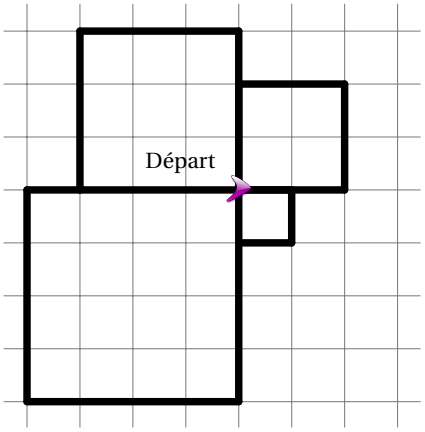

Carte n°5
Départ

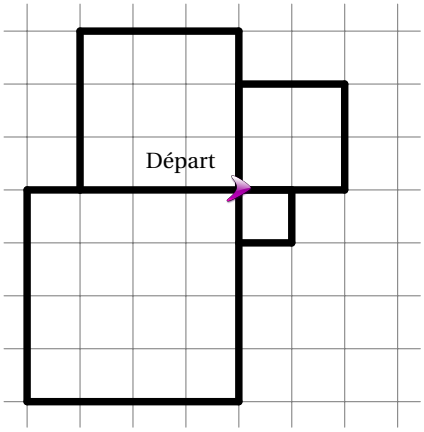

Carte n°6
Départ

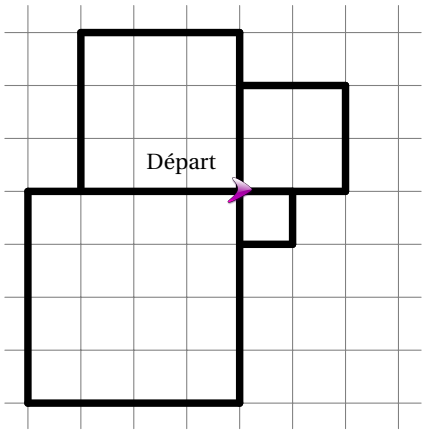

Carte n°7
Départ

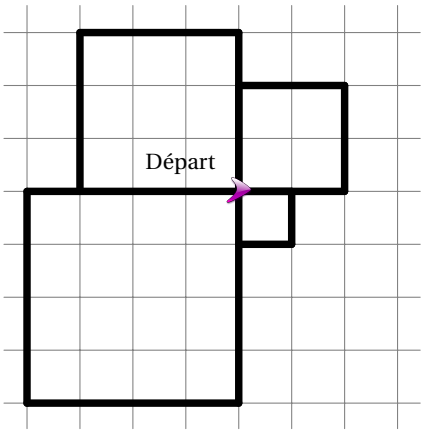

Carte n°8

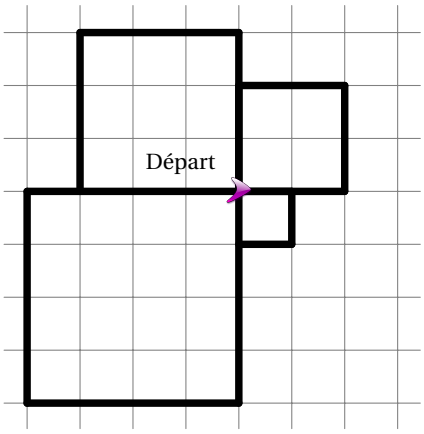

Départ

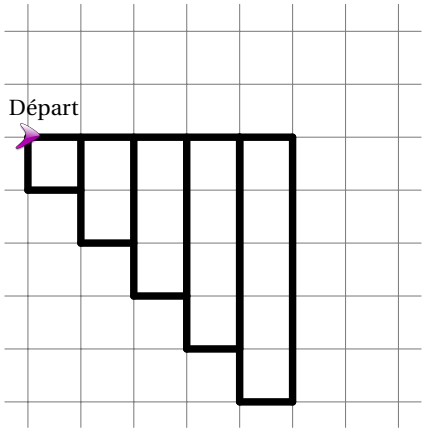

Départ

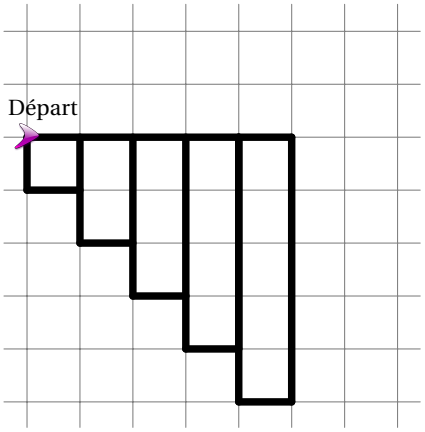

Départ

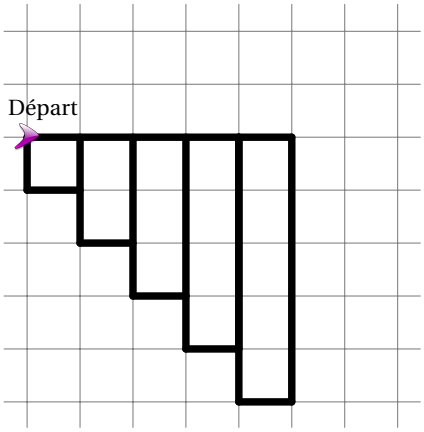

Départ

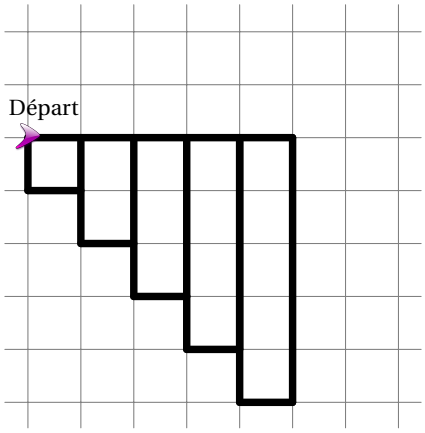

Départ

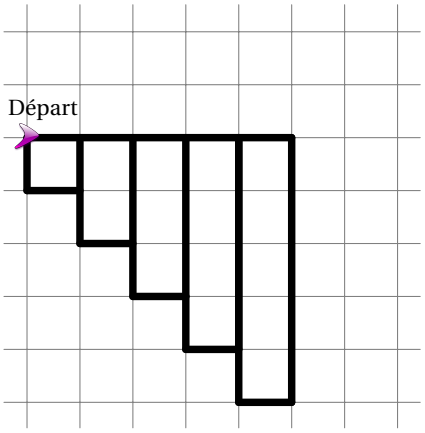

Départ

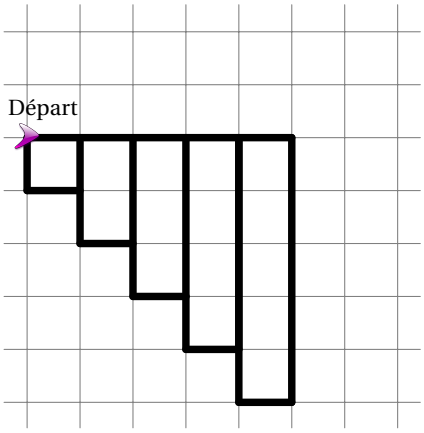

Départ


Départ


Départ


Départ


Départ


Départ

Chemin ou Carte n°.....

Chemin ou Carte n°.....

- - - - - - - - - - - - - - - ✂ -

Chemin ou Carte n°.....

Chemin ou Carte n°.....

- - - - - - - - - - - - - - - ✂ -

Chemin ou Carte n°.....

Chemin ou Carte n°.....