

I - TP 1


Découverte de la programmation

1 Le but de cet exercice est de découvrir l'interface, ainsi que les bases langage Python et du module turtle. Il n'y a rien de particulier à faire à part exécuter du code et chercher à le comprendre.

Pour réaliser cet exercice rejoindre [la page de code](#) correspondant à l'exercice.

L'interface est découpée en deux zones principale:

- **l'éditeur de code** qui se trouve à gauche;
- **la console** qui se trouve à droite.

Le bouton  permet d'afficher la zone d'affichage graphique.

Le bouton  permet d'afficher la console.

1. Le code python ci-dessous est déjà entré dans l'éditeur de code.

Code python

```

1  from turtle import *
2
3  forward(180)
4  left(90)
5  forward(100)
6  right(240)
7  forward(50)
8
9  # Commencer une ligne par un # permet de laisser un commentaire
10 # forward(70)
11 # le code en commentaire n'est pas exécuté
12 done() # Ne pas effacer
13 # Ne rien écrire ici

```

Cliquer sur le bouton "Exécuter" juste en dessous pour l'exécuter. Aller dans l'affichage graphique pour voir le résultat.

2. Modifier au minimum trois lignes de code comme vous le souhaitez (par exemple en changeant `left(90)` en `left(45)`, en commentant ou en décommentant une ligne de code), puis exécuter à nouveau.

Que font les fonctions `forward`, `left`, et `right` ?

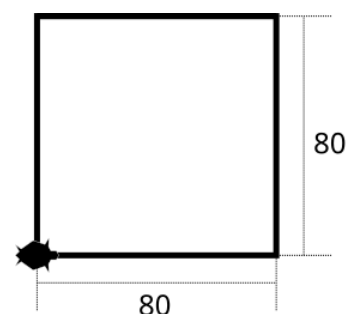
3. Remplacer le contenu de l'éditeur de code par : À l'aide des fonctions `forward` et `left` du module `turtle`, réaliser l'image suivante :

Code python

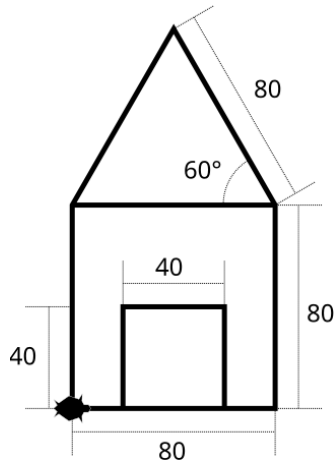
```

1  from turtle import *
2
3  # À compléter (plusieurs lignes
   → de codes sont attendues)
4
5  done() # Ne pas effacer
6  # Ne rien écrire ici

```



2 Écrire un code python permettant de réaliser la figure ci-dessous.



3 Pour répéter une série d'instructions plusieurs fois avec le langage Python, on utilise:

Code python

```
1 for i in range(nombre):  
2     # instruction répétée  
3     # ...  
4     # instruction répétée  
5 # instruction non répétée
```

Où nombre est le nombre de répétition voulue. Pour identifier le bloc d'instructions qui sera répété, *il est décalé vers la droite de 4 espaces*. On parle d'**indentation**.

L'indentation permet de créer l'équivalent d'un bloc en Scratch. En python, l'indentation est **significative** et peut être une source d'erreurs.

1. Rejoindre la [page de la question](#). Le code python ci-dessous y a été tapé.

Code python

```
1 from turtle import *  
2  
3 for i in range(3):  
4     forward(80)  
5     left(120)  
6  
7 done() # Ne pas effacer, ne pas écrire en dessous.
```

Modifier l'indentation, et uniquement l'indentation, du code proposé pour représenter un triangle équilatéral de côté 80 pixels.

2. Rejoindre la [page de la question](#).

a. Exécuter le programme.

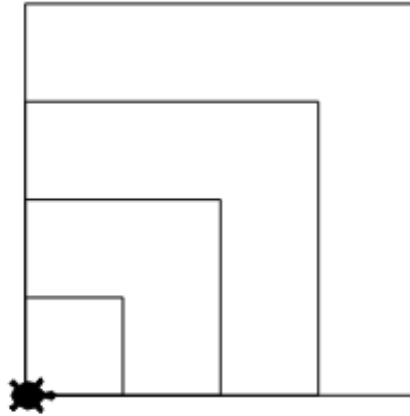
Quel type d'erreur est utilisé par Python pour signifier qu'il y a un problème avec les espaces en début de ligne ?

b. Corriger le code pour que le programme s'exécute sans erreur.

3. Écrire un code python permettant d'afficher un carré de côté 80.

Celui-ci ne doit comporter qu'une seule fois le mot forward et le mot left.

4 Écrire un code python permettant de réaliser la figure ci-dessous.



5 Nous allons par la suite souvent représenter des carrés, mais nous aimerions ne pas à avoir à nous répéter à chaque fois que l'on souhaite dessiner un carré.

De la même façon que la fonction forward nous permet d'avancer, nous aimerions créer une fonction carre qui permet de dessiner un carré.

La syntaxe pour définir sa propre fonction est:

Code python

```
1 def ma_fonction(argument1, argument2):
2     # instruction
3     # ...
4     # instruction
5     # fin de la définition
```

Pour utiliser une fonction, il suffit de **l'appeler** en écrivant son nom avec le bon nombre d'arguments:

Code python

```
1 ma_fonction(40, 70)
```

Rejoindre la [page de la question](#). Le code ci-dessous y a été tapé.

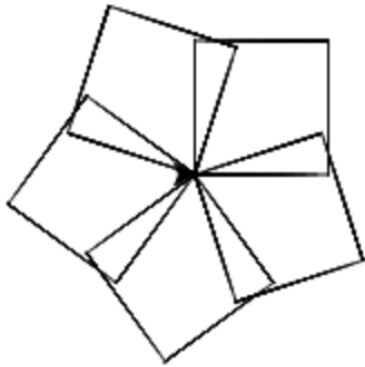
Code python

```
1 from turtle import *
2
3 def triangle(longueur):
4     for i in range(3):
5         forward(longueur)
6         left(120)
7
8 triangle(50)
9 # À compléter.
```

1. À quelle ligne se trouve la fin de la définition de la fonction triangle ?
2. Exécuter le code, puis le modifier pour qu'il affiche *en plus* un triangle de côté de longueur 100.
Attention, il faut appeler la fonction et **sans** modifier sa définition.
3. Définir une fonction carre qui permet d'afficher un carré dont la longueur des côtés sera donnée en argument de la fonction.
4. Tracer un carré de longueur 100 à l'aide de la fonction carre.
5. Dessiner la figure de la maison à l'aide des fonctions carre et triangle.

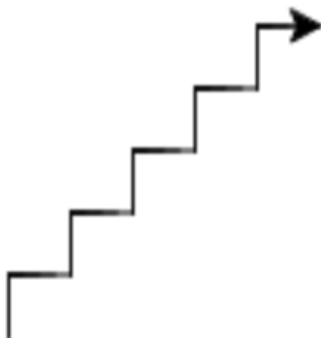
6

1. À l'aide de la fonction `carre` et d'une boucle `for` reproduire une figure similaire à la figure suivante, constituée de carrés de longueur 70 pixels.

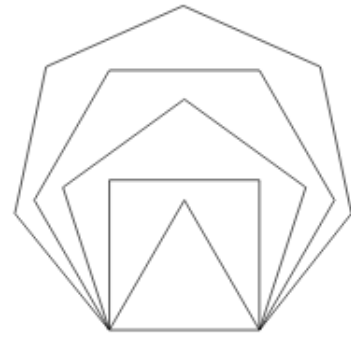


2. Écrire une fonction `escalier` possédant deux arguments: `hauteur_marche` et `nbr_marches`, dessinant un escalier possédant `nbr_marches` marches de hauteur `hauteur_marche`.

Par exemple, l'appel `escalier(20, 5)` doit afficher le dessin suivant :



3. a. Écrire une fonction `pentagone` d'argument `longueur` qui trace un pentagone de côtés de longueur: `longueur`.
- b. Les fonctions `carre`, `triangle` et `pentagone` se ressemblent beaucoup, or en informatique nous n'aimons pas nous répéter. Définir une fonction `polygone` de paramètres `nbr_cotes` et `longueur` qui construit le polygone régulier à `nbr_cotes` côtés de longueur: `longueur`.
- c. À l'aide de la fonction `polygone` et d'une boucle reproduire le dessin suivant :



4. La fonction `penup` permet de lever le stylo de la tortue, et donc de ne plus laisser de trace lorsqu'elle avance.

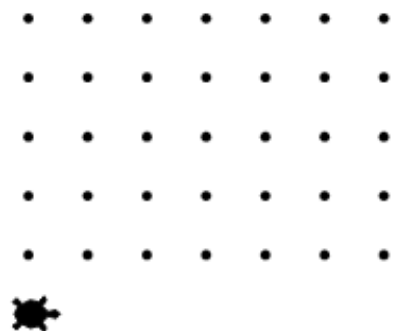
La fonction `dot` permet de tracer un point à l'emplacement de la tortue.

Code python

```
1 from turtle import *
2
3 penup()
4 dot()
5 forward(30)
6 dot()
7 forward(50)
8 dot()
9 done()
```

- a. Recopier et exécuter le code ci-dessus.
- b. Écrire un code python permettant de réaliser la figure ci-dessous.

L'écart entre chaque point est de 30 pixels, il y a 5 lignes et 7 colonnes de points.



Indication. On pourra commencer à écrire une fonction `ligne` de paramètre `nbr_points` qui représente une ligne de `nbr_points` points espacés de 30 pixels.