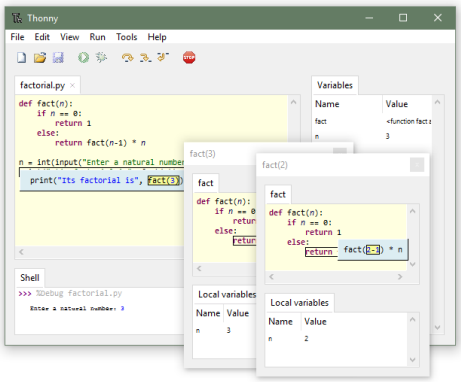
Activité 1 : Découverte

Les figures dynamiques : Dessiner avec Python et Turtle

1. L'IDE Thonny



Un IDE en anglais ou EDI en français (**E**nvironnement de **D**éveloppement **I**ntégré) est un logiciel qui réunit plusieurs outils nécessaires aux informaticiens pour créer des programmes.

Les fonctions de base d'un IDE sont donc :

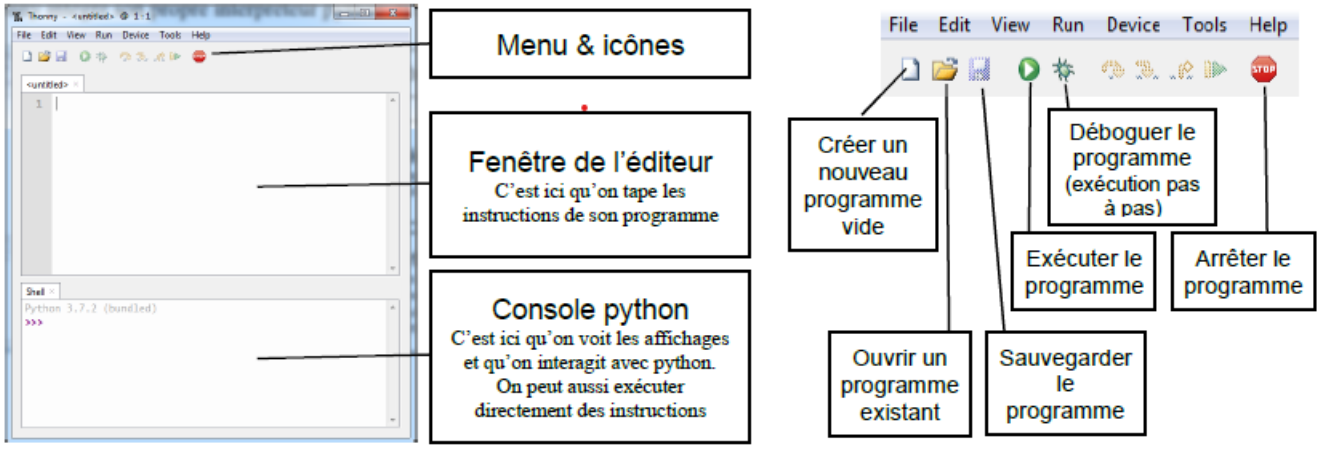
- L'éditeur de texte qui permet de saisir le code source
- Le module d'exécution qui comprend le compilateur et/ou l'interpréteur et permet de tester le programme
- Le débogeur (debugger en anglais) qui permet l'exécution pas à pas du programme et aide à trouver les erreurs

Certains EDI permettent d'aller plus loin en offrant des fonctions d'aide, la complétion automatique, un système de versionnage, de reformatage du code, de détection d'erreur, le support de plusieurs langages, ...

Nous utiliserons Thonny, un EDI pour python qui est assez simple et pédagogique.

Il est téléchargeable sur le site officiel : https://thonny.org/ (il existe une version portable particulièrement utile si on veut pouvoir développer sur n'importe quel ordinateur).
Thonny intègre son propre interpréteur python. Il n'est donc pas nécessaire d'installer python séparément.

Aperçu de l'interface de Thonny :



Cette interface permet d'utiliser python de deux façons différentes :

- En mode interactif

  On utilise alors la console python (partie du bas). Celle-ci affiche l'*invite de commande* (« `>>>` »). On peut alors taper une instruction et lorsqu'on appuie sur « Entrée », celle-ci est immédiatement exécutée et si elle produit un résultat, celui-ci est automatiquement affiché.
  Ce mode est adapté pour faire des essais et comprendre le comportement du langage.

- En mode script (ou « programme)

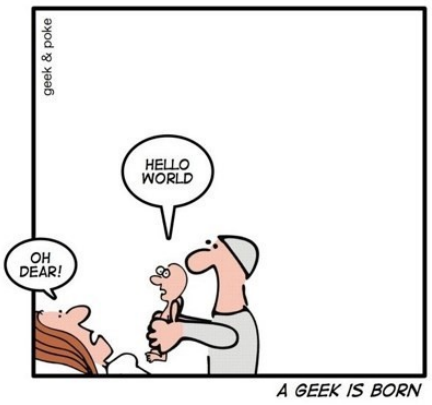
  On utilise c*ette fois la fenêtre de l'éditeur (en haut de l'interface) et on peut écrire une suite d'instructions qui seront enregistrées dans un fichier texte avec l'extension « `.py` ». Pour exécuter ces instructions, il faudra cliquer sur l'icône verte d'exécution, choisir l'option de menu « Run/Run current script » ou faire F5.*
  Ce mode est adapté pour écrire des programmes. C'est celui qu'on utilisera presque tout le temps.

2.Premier programme : Hello world

Les programmes python doivent suivre quelques règles de présentation :

- Python est sensible à la casse. C'est-à-dire qu'il fait la différence majuscule/minuscule. Ainsi pour lui print, Print et PRINT sont trois expressions différentes (et seule la première correspond à une fonction de python).
- Une ligne de programme se termine au saut de ligne : chaque ligne de texte du fichier correspond à une ligne du programme.

Bonjour le monde !



Il est de tradition chez les informaticiens de commencer l'apprentissage d'un nouveau langage par un programme qui ne fait qu'afficher une phrase à l'écran : c'est le programme « Hello world ! ».
L'instruction qui affiche du texte dans la console est dans presque tous les langages la commande `print`.
Comme toutes les fonctions, il faut lui fournir entre parenthèses les *arguments*, c'est-à-dire les données avec lesquelles elle doit travailler. Ici on fournit en argument le texte à afficher.
Là aussi c'est une quasi-constante dans tous les langages, les textes (ou *chaînes de caractères*) doivent être mis entre guillemets simples `'` ou doubles `"`

- ✗ **Écrire** un programme « Hello_world.py » qui affiche « Hello world ! »
- ✗ **l'exécuter et vérifier** qu'on obtient bien le résultat attendu.

3.Principes de base

Maintenant que l'on sait écrire un programme et l'exécuter, on va s'en servir pour mettre en évidence quelques concepts de programmation.

Le principe de fonctionnement global de tous les langages est le même : à chaque ligne du programme la machine va chercher à ***évaluer*** l'expression qu'il rencontre. Pour ce faire il décompose la ligne en sous-expressions puis applique les opérateurs ou fonctions qu'il rencontre. Chaque fonction ou opérateur renvoi un résultat jusqu'à ce que l'évaluation globale de la ligne produise également un résultat (qui peut être `None`, c'est-à-dire rien d'utile en python).

4.Dessiner avec Python et Turtle

Dans cette activité, nous allons réaliser de figures dynamiques en utilisant le langage de programmation Python et particulièrement sa bibliothèque turtle. Ce module permet de déplacer un point dans un espace 2D (ce point est souvent vu comme une tortue), et ainsi de réaliser des dessins. Les commandes de bases de la bibliothèque Turtle sont les suivantes :

up() : lève le crayon
down() : baisse le crayon
forward(n) : avance de n
left(d) : tourne vers la gauche de d degrés
right(d) : tourne vers la droite de d degrés
goto(x,y) : se déplace vers le point de coord. (x,y)
circle(r) : dessine un cercle de rayon r

width(e) : définit l'épaisseur du trait
speed("texte") : définit la vitesse d'exécution
write("texte") : écrit le texte
color("couleur") : définit la couleur du trait
bgcolor("couleur") : définit la couleur de fond
reset() : efface tout
done() : arrête le dessin

Remarques :
- Dans les instructions color() et bgcolor() précédentes, les couleurs suivantes peuvent être choisies : blue, red, black, green, . . .
- Pour la vitesse, on peut choisir (du plus rapide au plus lent) : slowest, slow, normal, fast et fastest.
- Le début de chaque programme doit faire appel à la bibliothèque de fonctions Turtle en tapant la directive from turtle import *

1. Les premières figures...

- ✗ **Créer** un fichier nommé first.py et contenant le code ci-contre.
- ✗ **Exécuter** ce programme et **analyser** le résultat afin d'identifier le rôle de chaque instruction
- ✗ **Modifier** ce programme afin de tester les fonctions suivantes : right, circle, width et color.
- ✗ **Dessiner** un carré rouge de côté 100 et un autre vert deux fois plus petit et centré sur le premier
- ✗ **Dessiner** un carré rouge de côté 100 et un cercle vert de rayon 80 centré sur le carré

```
from turtle import *
longueur = 120
angle = 90
reset()
forward(longueur)
left(angle)
color('red')
forward(longueur-40)
done()
```

5.Booster ses programmes

Les répétitions

Nous nous rendons compte que le programme précédent comporte plusieurs fois les mêmes instructions. Il est possible de simplifier le code en répétant plusieurs fois le groupe d'instruction à l'aide d'une boucle for. La syntaxe de de cette structure répétitive est la suivante :

```
for i in range(2) :
    Instruction 1
    Instruction 2
```

Instructions à répéter. Elles doivent décalées d'une tabulation

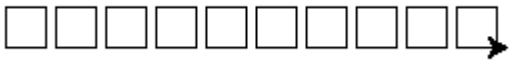
Nombre de répétitions

En utilisant cette boucle répétitive, le code pour dessiner un carré de 50 de côté devient :

```
from turtle import *
reset()
for i in range(4) :
    forward(50)
    Left(90)
done()
```

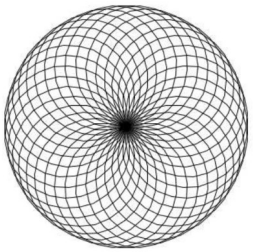
✗ **Dessiner** un triangle équilatéral vert de 70 de côté en utilisant une boucle for pour répéter 3 fois le dessin d'un côté du triangle

✗ **Réaliser** le dessin de la figure suivante constituée de dix carrés de côté 20 avec un espace de 5 entre chacun d'eux.



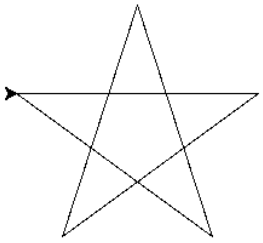
✗ **Écrire** un programme réalisant le dessin ci-contre.

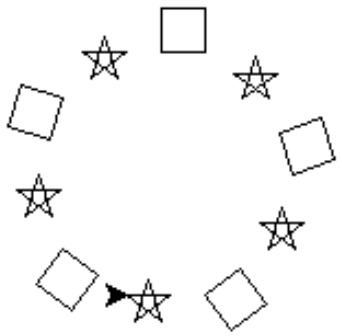
Pour obtenir ce dessin, on peut observer qu'il est constitué de cercles de même rayon (80 dans notre cas), avec un décalage de 10 degrés entre deux cercles successifs (soit 36 cercles au total).



6. Pour aller plus loin

✗ **Écrire** un programme, qui trace les figures suivantes :





7. Jouons avec les couleurs

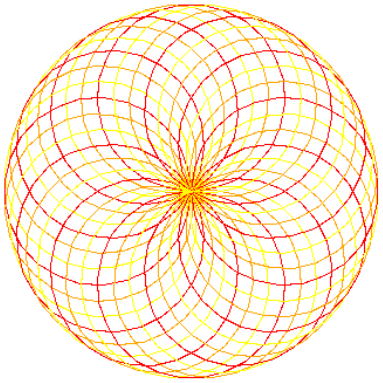
La fonction color() modifie la couleur du tracé en cours. En conditionnant l'appel de cette fonction à certaines conditions il est alors possible d'introduire des couleurs dans les figures.
Des conditions peuvent être implémentées dans le programme avec l'instruction if. Cette instruction exécute une suite d'instructions si l'expression associée est vraie :

Instructions à effectuer. Elles doivent être décalées d'une tabulation

```
if expression :
    Instruction 1
    Instruction 2
```

Expression associée de test

✗ **Modifier** les deux programmes précédents afin d'obtenir les figures suivantes :



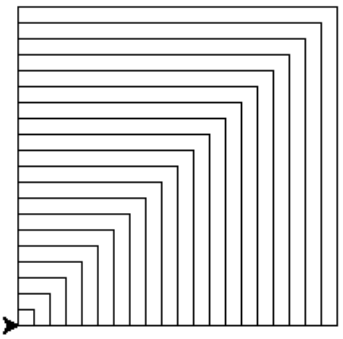
8. Les variables

Les variables sont l'un des concepts qui se retrouvent dans la totalité des langages de programmation. Le principal intérêt d'une variable réside dans le fait qu'il est possible de la faire évoluer durant l'exécution du programme. Par exemple le code ci-contre permet de dessiner deux carrés l'un dans l'autre.

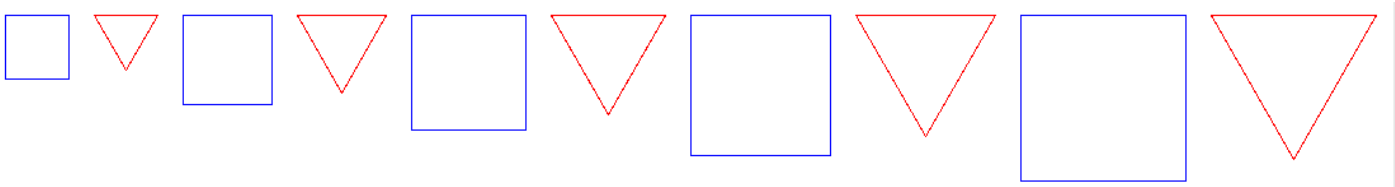
✗ **Analyser** ce code et **déterminer** la longueur des côtés des deux carrés

```
from turtle import *
reset()
speed('fastest')
down()
lg = 80
for i in range(2) :
      for j in range(4):
            forward(lg)
            left(90)
      lg = lg + 200
done()
```

✗ **Modifier** le programme précédent afin d'obtenir la figure suivante. On peut noter qu'elle est constituée de 20 carrés dont le plus petit à des côtés de 10 et le dernier de 200



✗ **Écrire** un programme, qui trace un carré puis un triangle, qui grossissent au fur et à mesure.



9. Pour aller plus loin

✗ Voici des figures dont vous devez trouver le code :