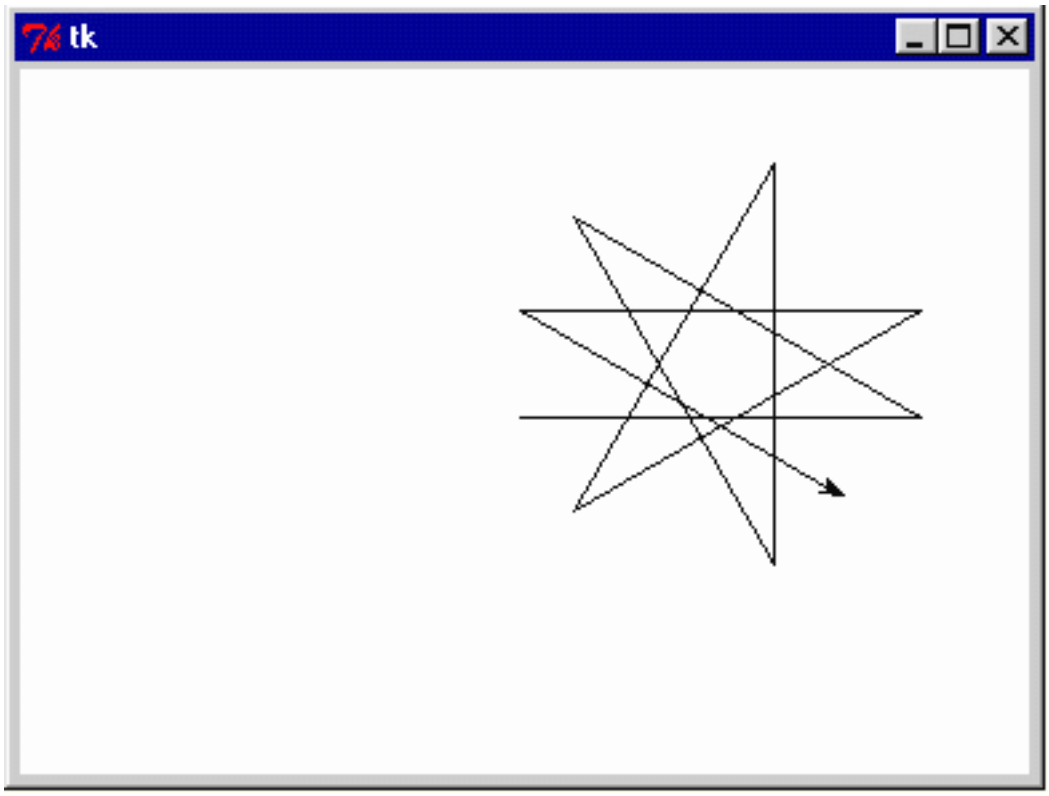
1. Le module "turtle", graphisme en Python

L'initiation au langage de programmation Python s'est fait entièrement en mode texte.
Le but du module "turtle" est de générer du graphisme, c'est-à-dire des lignes, des courbes, des figures géométriques, de diverses couleurs.



L'une des grandes qualités de Python est qu'il est extrêmement facile de lui ajouter de nombreuses fonctionnalités par importation de divers **modules**.
Pour illustrer cela, et nous amuser un peu avec d'autres objets que des nombres, nous allons explorer un module Python qui permet de réaliser des « graphiques tortue », c'est-à-dire des dessins géométriques correspondant à la piste laissée derrière elle par une petite « tortue » virtuelle, dont nous contrôlons les déplacements sur l'écran de l'ordinateur à l'aide d'instructions simples.

Exercice 1.1 :
Dans l'éditeur, créez un nouveau fichier et tapez :

```
# À vous d'écrire un commentaire pertinent ici.
from turtle import *
forward(120)
left(90)
color("red")
forward(80)
```

Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0110_turtle_debut.py**
Exécutez-le en pressant la touche **F5**.

Exercice 1.2 :
En utilisant le module "turtle" et les quelques instructions précédentes, dessinez un carré de 180 pixels de côtés.

Exercice 1.3 :
Modifiez le code précédent pour écrire une fonction qui dessine un carré de "long" pixels de côtés et de couleur "coul". "long" et "coul" sont deux paramètres de la fonction.
Si "coul == None", garder la couleur actuellement définie, ne pas la redéfinir.

Ajoutez une fonction qui dessine un triangle équilatéral de "long" pixels de côtés et de couleur "coul".

Ajoutez une fonction qui dessine un pentagone de "long" pixels de côtés et de couleur "coul".

Exercice 1.4 :
Écrivez une fonction qui dessine un polygone régulier à "nbcotes" côtés de longueurs "long" pixels et de couleur "nbcotes", "long" et "coul" sont trois paramètres de la fonction.
Si "coul == None", garder la couleur actuellement définie, ne pas la redéfinir.

Voici quelques fonctions mises à votre disposition dans le module "turtle" :

`setup(width=900, height=950, startx=10, starty=15)`	Définit la position et les dimensions.
`speed(0)`	Augmente la vitesse de la tortue.
`delay(0)`	Augmente encore plus la vitesse de la tortue.
`tracer(0, 0)`	N'affiche pas le tracé. C.f. "help(tracer)".
`update()`	Met à jour le tracé, donc affiche le tracé.
`reset()`	On efface tout et on recommence
`forward(distance)`	Avancer d'une distance donnée
`backward(distance)`	Reculer
`goto(x, y)`	Aller à l'endroit de coordonnées x, y
`left(angle)`	Tourner à gauche d'un angle donné (exprimé en degrés)
`right(angle)`	Tourner à droite
`up()`	Relever le crayon (pour pouvoir avancer sans dessiner)
`down()`	Abaisser le crayon (pour recommencer à dessiner)
`width(épaisseur)`	Choisir l'épaisseur du tracé
`colormode(255)`	Le triplet de couleur (r, g, b) se fait avec des nombres entre 0 et 255
`color(couleur)`	Couleur peut être une chaîne prédéfinie ('red', 'blue', etc.) ou un tripplet (r,g,b) c.f. : https://ecsdtech.com/8-pages/121-python-turtle-colors
`fillcolor(couleur)`	Couleur de remplissage, à mettre après l'instruction "color(...)"
`begin_fill()`	Début de la figure ayant un contour fermé, à remplir avec la couleur sélectionnée
`end_fill()`	Remplir un contour fermé à l'aide de la couleur sélectionnée
`write(texte)`	"texte" doit être une chaîne de caractères

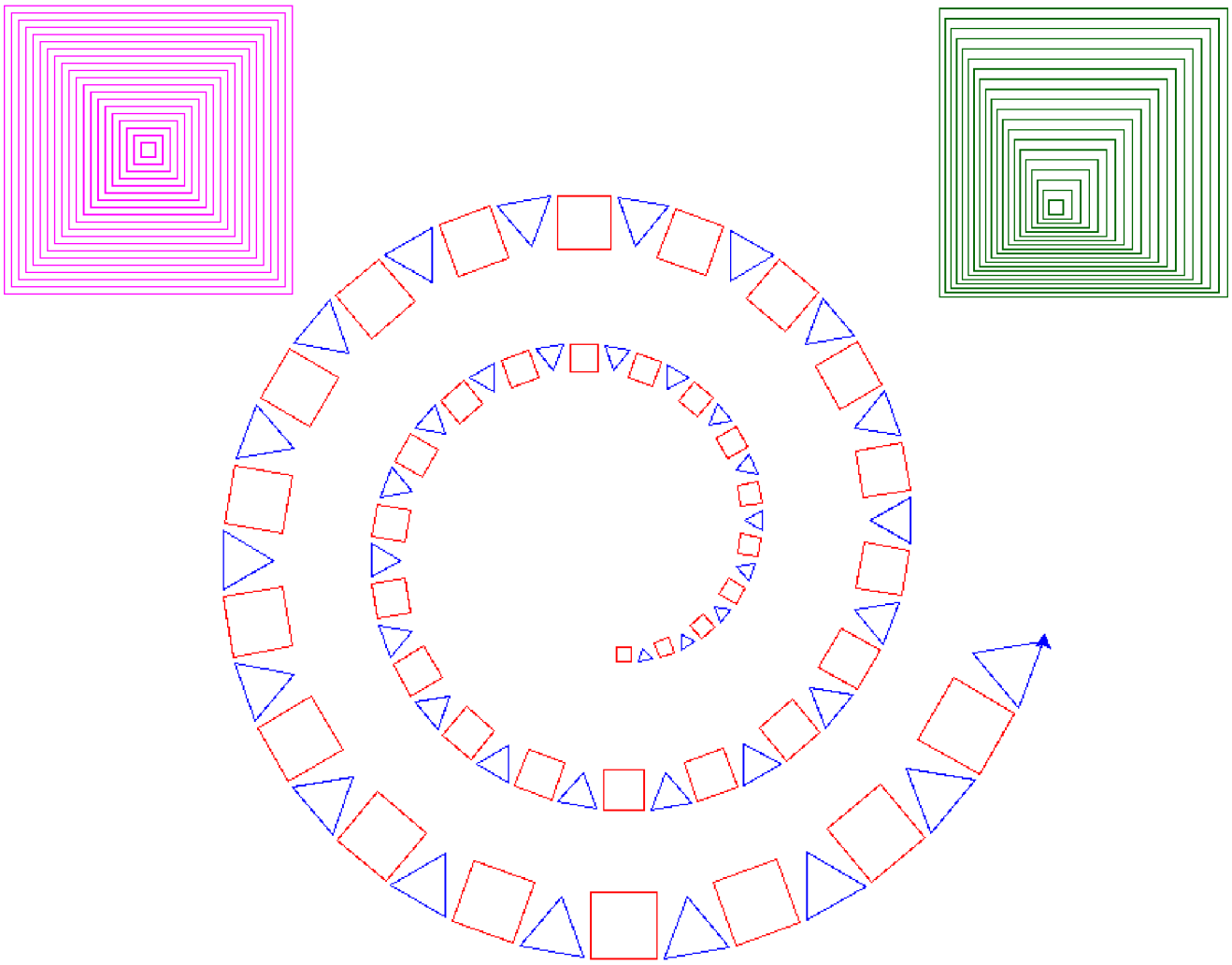
c.f. https://docs.python.org/release/3.7.2/library/turtle.html

Exercice 1.5 :
Dans l'éditeur, créez un nouveau fichier et dans le dossier **adoc** sous le nom **tu0150_carres_et_triangles.py**.

Écrivez une fonction **carre(taille, couleur)**, qui trace un carré de *taille* et de *couleur* définie dans les paramètres. Elle dessine le carré depuis la position donnée, avec l'orientation donnée.

Écrivez une fonction **triangle(taille, couleur)**, qui trace un triangle de *taille* et de *couleur* définie dans les paramètres. Elle dessine le triangle depuis la position donnée, avec l'orientation donnée.

Écrivez le code nécessaire pour dessiner la figure suivante.



Dans la spirale, chaque carré et chaque triangle est suivi d'une rotation de 10°.
La longueur initiale d'un côté est de 10. Elle augmente de 1 au passage à la figure suivante.

Il n'est pas nécessaire d'avoir exactement le même rendu.
Un rendu similaire est suffisant.

Méthodes Turtle et Screen disponibles

https://docs.python.org/release/3.7.2/library/turtle.html

Turtle motion

Move and draw

```
forward(distance) | fd(distance)
backward(distance) | back() | bk()
right(angle) | rt()
left(angle) | lt()
goto(x,y) | setpos() | setposition()
setx(pos_x)
sety(pos_y)
setheading(to_angle) | seth()
home()
circle(rayon{,angle,nb_cotes})
dot(diamètre, couleur)
stamp() #copie l'image de la souris → id
clearstamp(id)
clearstamps()
undo()
speed(V) #vitesse entre 1 et 10, 0=rapide.
```

Tell Turtle's state

```
position() | pos()
towards(x, y) → angle
xcor() → coord. x
ycor() → coord. y
heading() → angle
distance(x, y) → distance à (x,y)
```

Setting and measurement

```
degrees()
radians()
```

Pen control

Drawing state

```
pendown() | pd() | down() #dessiner
penup() | pu() | up() #pas dessiner
pensize() | width()
pen()
isdown()
```

Color control

```
color("#rrggbb") #00 <= r,g,b <=ff
pencolor()
fillcolor()
```

Filling

```
filling()
begin_fill() #début remplissage de la figure
end_fill() #effectue remplissage de la figure
```

More drawing control

```
reset()
clear()
write()
```

Turtle state

Visibility

```
showturtle() | st()
hideturtle() | ht()
isvisible()
```

Appearance

```
shape(s)  # s="arrow", "turtle", "circle", "square", "triangle", "classic".
resizemode()
shapesize() | turtlesize()
shearfactor()
settiltangle()
tiltangle()
tilt()
shapetransform()
get_shapepoly()
```

Using events

```
onclick()
onrelease()
ondrag()
```

Special Turtle methods

```
begin_poly()
end_poly()
get_poly()
clone()
getturtle() | getpen()
getscreen()
setundobuffer()
undobufferentries()
```

TurtleScreen

Window control

```
bgcolor(r,g,b)  # 0 <= r,g,b <= 1
bgpic()
clear() | clearscreen()
reset() | resetscreen()
screensize()
setworldcoordinates()
```

Animation control

```
delay(vitesse)  # vitesse max = 0
tracer(n,m)  # accélère l'affichage.
update()  # met à jour l'affichage
```

Using screen events

```
listen()
onkey() | onkeyrelease()
onkeypress()
onclick() | onscreenclick()
ontimer()
mainloop()
```

Settings and special methods

```
mode()
colormode(255)  # indique que les couleurs r, g et b sont des nombres entre 0 et 255.
getcanvas()
getshapes()
register_shape() | addshape()
turtles()
window_height()
window_width()
```

Input methods

```
textinput()
numinput()
```

Methods specific to Screen

```
bye()  #ferme la fenêtre
exitonclick()
setup()
title(str)  #str = "titre de la fenêtre
```

2. Plusieurs tortues

Exercice 2.1 :
Dans l'éditeur, créez un nouveau fichier et tapez :
 # À vous d'écrire un commentaire pertinent ici. Remarquez les commentaires ci-dessous.

```
from turtle import *

# défini la position et la taille de la fenêtre.
setup(width=900, height=600, startx=10, starty=15)

#delay(0) # Pour régler la vitessse d'animation

# Au lieu de la tortue de base, on crée une première tortue.
tortue1 = Turtle(shape="circle") # Autres: "arrow", "turtle", "circle", "square", "triangle", "classic".
tortue1.speed(5) # Vitesse de la tortue 1
tortue1.color("blue")
tortue1.goto(-300, 0) # Se positionne à gauche, au milieu de la hauteur.
tortue1.forward(400)

# Création d'une deuxième tortue
tortue2 = Turtle(shape="turtle")
tortue2.speed(5) # Vitesse de la tortue 2
tortue2.color("red")
tortue2.up()
tortue2.goto(300,50)
tortue2.down()
tortue2.left(180) # Se dirige vers la gauche.
tortue2.forward(400)
```

Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0210_deux_tortues.py**
Exécutez-le en pressant la touche **F5**.

Exercice 2.2 :
Dans l'éditeur, créez un nouveau fichier et tapez :
 # À vous d'écrire un commentaire pertinent ici. Remarquez les commentaires ci-dessous.

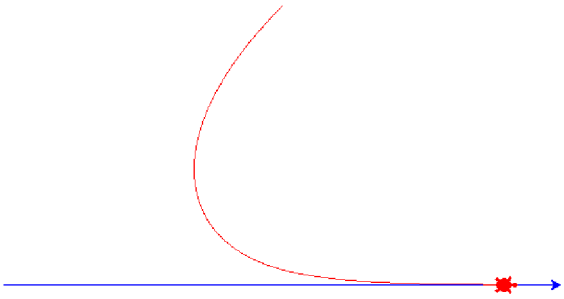
```
from turtle import *

# défini la position et la taille de la fenêtre.
setup(width=900, height=600, startx=10, starty=15)

delay(2) # Pour régler la vitessse d'animation

# Au lieu de la tortue de base, on crée une première tortue.
tortue1 = Turtle(shape="circle") # Autres: "arrow", "turtle", "circle", "square", "triangle", "classic".
tortue1.speed(0) # Vitesse lente de la tortue par défaut
tortue1.color("blue")
tortue1.up(); tortue1.goto(-300, -100); tortue1.down()

# Création d'une deuxième tortue
tortue2 = Turtle(shape="turtle")
tortue2.speed(0) # Vitesse lente de la tortue 2
tortue2.color("red")
tortue2.up(); tortue2.goto(0,290); tortue2.down()

# La tortue 1 avance, la deuxième se dirige vers la première
for nn in range(700):
    tortue1.forward(1) # La première tortue avance
    x1, y1 = tortue1.xcor(), tortue1.ycor() # Position de la première tortue
    tortue2.seth(tortue2.towards(x1, y1)) # La deuxième tortue s'oriente en direction de
la première
    tortue2.forward(1) # La deuxième tortue avance
```

Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0220_deux_tortues_dependantes.py**
Exécutez-le en pressant la touche **F5**.

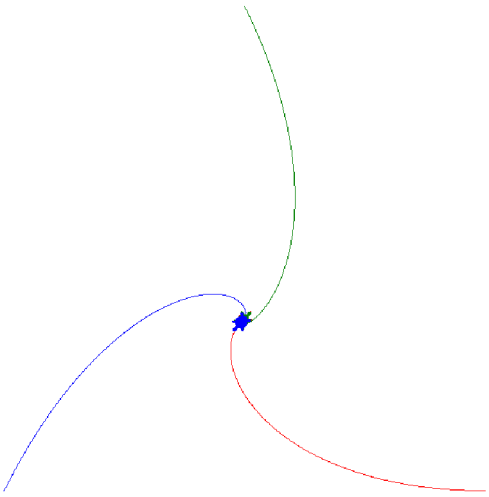
Exercice 2.3 :
Écrivez le code pour que trois tortues se placent en triangle et que la tortue 3 se dirige vers la tortue 2, qui se dirige vers la tortue 1, qui se dirige vers la tortue 3.
Elles s'arrêtent lorsqu'elles sont très proches les unes des autres.
Vous avez la liberté de placement des tortues, des couleurs de la vitesse, etc.
Remarque : tortue1 = turtles()[0] # Permet de donner un nom à la première tortue.

Exercice 2.4 :
Écrivez le code pour que quatre tortues se placent en rectangle et que la tortue 4 se dirige vers la tortue 3, qui se dirige vers la tortue 2, qui se dirige vers la tortue 1, qui se dirige vers la tortue 4.
Elles s'arrêtent lorsqu'elles sont très proches les unes des autres.
Vous avez la liberté de placement des tortues, des couleurs de la vitesse, etc.
Remarque : tortue1 = turtles()[0] # Permet de donner un nom à la première tortue.

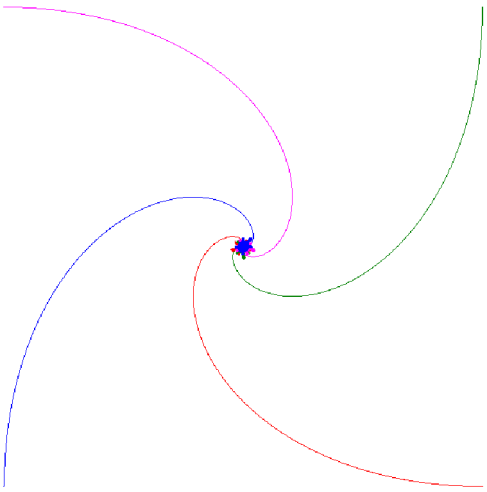
On pourrait demander d'écrire un code similaire avec 5 tortues ou 6 ou 7 ou ...
On se rend compte que le code est répétitif et qu'il nous manque un concept pour être plus efficace.



Résultat de l'exercice 2.2, la tortue rouge avance vers la tortue bleue.



Résultat de l'exercice 2.3, avec 3 tortues.



Résultat de l'exercice 2.4, avec 4 tortues.

3. Les listes

Au lieu d'avoir des variables "tortue1", "tortue2", "tortue3", etc., il est plus agréable d'avoir une liste de tortues. C'est ce que les **listes** de Python permet.

Dans l'éditeur, créez un nouveau fichier et tapez :

```
# tu0305_turtle_listes.py
'''
Utilisation de listes pour avoir beaucoup de tortues.
Dessine plusieurs tortues, en ligne, avances vers la droite
'''

from turtle import *
from time import sleep  # pas important, juste pour mettre des pauses d'exécutions.

# défini la position et la taille de la fenêtre.
setup(width=900, height=950, startx=10, starty=15)
delay(0) # Pour aller plus vite
colormode(255) # mode de définition des couleurs

nbTortues = 9 # Défini le nombre de tortues désirées

listeTortues = [] # Défini une liste vide

# Boucle de création de nouvelles tortues
for nn in range(nbTortues): # va de 0 à nbTortues-1
    tortueNouvelle = Turtle(shape="turtle") # Création d'une nouvelle tortue
    listeTortues.append(tortueNouvelle) # Ajoute dans la liste, une référence à la tortue.
    tortueNouvelle.speed(0) # Vitesse maximale de la tortue
    # défini la couleur, (r, g, b)
    tortueNouvelle.color((255 - 255*nn // nbTortues, 0, 255*nn // nbTortues))

    # Positionne la tortue
    tortueNouvelle.up()
    tortueNouvelle.goto(-350, -300 + 30*nn)
    tortueNouvelle.down()

# Pour changer des caractéristique de la première tortue.
listeTortues[0].color("green")
listeTortues[0].up(); listeTortues[0].goto(-350, -200); listeTortues[0].down()

print("Nombre de tortues =", len(listeTortues))

# Boucle pour faire avancer les tortues
for jj in range(0, 200):
    # Chaque tortue avance vers la droite
    for nn in range(nbTortues):
        listeTortues[nn].forward(1) # La tortue numéro  nn  avance

# Autre manière de faire pour accéder à chaque tortue
# Cela n'est pas utile, si on veut accéder à une tortue voisine.
for uneTortue in listeTortues:
        uneTortue.left(45)
        sleep(0.3)
        uneTortue.forward(100)
        sleep(0.3)
```

En **gras** se trouve les nouvelles instructions liées aux listes.

Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0305_turtle_listes.py**
Exécutez-le en pressant la touche **F5**.

Exercice 3.1 :
Sauvegardez le code précédent dans le dossier **adoc** sous le nom **tu0310_turtle_listes.py**
Modifiez le code pour que chaque tortue avant à une vitesse variable.
Le module random, avec la fonction randint(min, max) sera utile.

Exercice 3.2 :
En partant du code de la page précédente :
Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0320_turtle_listes.py**
- *Pour des essais, fixez* `nbTortues` *à un nombre entre 3 et 10. (une ligne)*
- *Les modules math sera utile. (une ligne)*
- *Disposer de* `nbTortues` *tortues sur un cercle de 350 pixels de rayon. (4 lignes)*
  Il faut connaître le cercle trigonométrique pour faire cela.
- *Ajoutez à la fin de la liste, une référence à la première tortue de la liste, cela sera utile pour simplifier la suite. (une ligne)*
- *Faites en sorte que chacune tortue s'oriente vers sa voisine de gauche. (3 lignes)*
- *Répétez 100 fois :*
  chaque tortue avance de 1 pixel,
  puis elle s'oriente vers sa voisine de gauche. (total = 5 lignes)

Si tout fonctionne bien, répétez quelques centaines de fois au lieu de 100 fois.
Si vous arrivez, faites en sorte que les tortues s'arrêtent lorsqu'elles sont proches les unes des autres.

Remarques supplémentaires sur les listes.

On peut aussi définir une liste de la manière suivante :
jourSemaine = ["dimanche", "lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"]

Testes possibles :
- "print(len(jourSemaine))", pour connaître la longueur de cette liste.
- ```
  for jour in  jourSemaine:
      print(jour)
  ```
- ```
  print(jourSemaine.index("mercredi"))
  ```
- ```
  maListeDeCaracteres = list("abcdef séparés")
  print(maListeDeCaracteres)
  maListeDeCaracteres[0] = "A"
  maChaine = ''.join( maListeDeCaracteres)
  print(maChaine)
  ```
- ```
  for nn, jour in enumerate(jourSemaine):
      print(nn, jour)
  ```

c.f. : https://www.w3schools.com/python/python_ref_list.asp

Exercice 3.3 :
En partant du code de la page précédente :
Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0330_turtle_random.py**
° *Pour des essais, fixez* `nbTortues` *à un nombre entre 3 et 10. (une ligne)*
° *Les modules math sera utile. (une ligne)*
° *Les modules random sera utile. (une ligne)*
° *Disposer toutes les tortues à l'origine. (4 lignes)*
° *Répétez 100 fois : (4 lignes)*
Pour chaque tortue :
elle tourne d'un angle aléatoire entre -20 et 20 degrés ;
elle avance de 1 pixel,

Si tout fonctionne bien, répétez quelques centaines de fois au lieu de 100 fois. Vous pouvez également augmenter le nombre de tortues.

Exercice 3.4 :
En partant du code précédent, du code sera ajouté.
Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0340_turtle_random.py**
° *La tortue numéro 0 est noire.*
° *Calculer la position moyenne des tortues de 1 à* `nbTortues`. *(3 lignes)*
On fait la moyenne des coordonnées X des tortues et la moyenne des coordonnées Y des tortues.
° *Positionner la tortue noire en cette position moyenne. (1 ligne)*
° *À la création des tortues, leur donner des directions aléatoires.*

Si vous arrivez, faites en sorte que les tortues s'arrêtent lorsqu'une tortue sort de la fenêtre.

Exercice 3.5 : *
En partant du code précédent, du code sera ajouté.
Sauvegardez votre fichier dans le dossier **adoc** sous le nom **tu0350_turtle_random.py**
° *La tortue numéro 1 est rouge.*
° *Calculer la position moyenne des tortues de 2 à* `nbTortues`. *(3 lignes)*
° *Une fois la moyenne calculée, calculer la variance, qui est aussi le carré de l'écart quadratique moyen. C'est la somme des carrés des distances entre les tortues et la position moyenne, divisée par le nombre de tortues.*
Cela estime la distance moyenne entre la position moyenne et les tortues.
° *Placez la tortue numéro 1 en la position (compteur de temps - 400, 0.01*variance - 400).*
Remarquez comment elle croit.

tracer(100) accélère la vitesse d'affichage. Plus l'argument est grand, plus l'affichage est rapide.

En augmentant le nombre de tortues, comment cela affecte-t-il la position moyenne et la variance ?

Définitions :
La variance est la somme des carrés des écarts à la moyenne, divisé par le nombre de tortues.
L'écart-type est la racine carrée de la variance. Elle estime la distance moyenne entre les tortues et leur position moyenne.
Elle a de meilleures propriétés mathématiques que la moyenne des distances entre les tortues et leur position moyenne.

Ces deux notions sont fondamentales en statistique et en probabilité.