

Initiation à Python

Séance 4

Valentin Bahier

2020-2021



Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue

Tracer des courbes

Un module approprié pour tracer des courbes est `matplotlib`, plus précisément le sous-module `matplotlib.pyplot`, qu'il est d'usage d'importer avec l'alias `plt`. Préparer les données à tracer se fera généralement à l'aide du module `NumPy`, c'est pourquoi dans cette séance nous commencerons par écrire les deux lignes

```
1 import numpy as np
2 import matplotlib.pyplot as plt
```

Exercice 28

- Exécuter le code suivant et observer le résultat

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.array([1, 2, 3, 6])
5 y = np.array([2, 1, 3, 3])
6 plt.plot(x, y)
7 plt.show()
```

- Modifier le code ci-dessus de sorte à tracer le carré $ABCD$ de sommets de coordonnées $A(0;0)$, $B(0;1)$, $C(1;1)$ et $D(1;0)$.

Exercice 28

- Exécuter le code suivant et observer le résultat

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.array([1, 2, 3, 6])
5 y = np.array([2, 1, 3, 3])
6 plt.plot(x, y)
7 plt.show()
```

- Modifier le code ci-dessus de sorte à tracer le carré $ABCD$ de sommets de coordonnées $A(0;0)$, $B(0;1)$, $C(1;1)$ et $D(1;0)$.

Réponse : x représente les abscisses des points, et y les ordonnées des points. Pour la deuxième question il suffit donc de remplacer les lignes 4 et 5 par

```
1 x = np.array([0, 1, 1, 0, 0])
2 y = np.array([0, 0, 1, 1, 0])
```

⇒ La méthode `plot` indique ce qu'il y a à tracer, puis la méthode `show` affiche la figure à l'écran.

Plan

Matplotlib

Tracer des courbes

Quelques options d'affichage

Turtle

Nommer sa tortue

Promener sa tortue

Customiser sa tortue

Adapter le terrain

Colorier les figures réalisées

Cloner sa tortue

Formats de courbes

Les options suivantes doivent être mises entre guillemets et concaténées en paramètre du `plot` (on donne ici seulement quelques possibilités) :

Couleurs

b	bleu
g	vert
r	rouge
c	cyan
m	magenta
y	jaune
k	noir
w	blanc

Styles de lignes

-	ligne continue
--	tirets
:	pointillés
-.	tirets points

Marqueurs

.	point
,	pixel
o	cercle
s	carré
+	+
x	x
*	étoile

Par exemple, le code

```
1 x = np.array([0, 1, 4, 5, 7])
2 y = np.array([0, 0.5, 1, 1, 1])
3 plt.plot(x,y,"r-.s")
```

prépare une courbe rouge avec des tirets et des marqueurs carrés.

Titres, légendes, étiquettes des axes...

Voici quelques méthodes du module `plt` pour améliorer la présentation d'un graphique :

<code>title</code> (chaîne de caractères)	titre du graphique
<code>xlim</code> (<i>a</i> , <i>b</i>)	abscisses entre les valeurs <i>a</i> et <i>b</i>
<code>ylim</code> (<i>a</i> , <i>b</i>)	ordonnées entre les valeurs <i>a</i> et <i>b</i>
<code>xlabel</code> (chaîne de caractères)	nom de l'axe des abscisse
<code>ylabel</code> (chaîne de caractères)	nom de l'axe des ordonnées
<code>xticks</code> (liste des endroits, liste des noms)	étiquettes des abscisses
<code>yticks</code> (liste des endroits, liste des noms)	étiquettes des ordonnées

Affichage de plusieurs courbes

Il est naturellement possible d'afficher plusieurs courbes sur un même graphique. Il suffit pour cela de préparer plusieurs tracés (avec la méthode `plot`) avant d'écrire l'instruction `plt.show()`. La couleur est automatiquement changée si on ne précise pas.

Afin de savoir quelle courbe représente quoi, il est pratique de mettre une légende, ce qui s'effectue de la manière suivante

```
1 x = np.array([0, 1, 4, 5, 7])
2 y1 = np.array([0, 0.5, 1, 1, 1])
3 y2 = np.array([2, 2, 1.5, 1, 0])
4 plt.plot(x,y1,label="ma première belle courbe")
5 plt.plot(x,y2,label="ma deuxième belle courbe")
6 plt.legend() # l'instruction sans laquelle les labels
               ne s'affichent pas
7 plt.show()
```

Exercice 29

Créer un graphique avec le titre "Fonction identité VS fonction carrée" sur lequel sont représentées les courbes d'équations $y = x$ et $y = x^2$, pour x entre -1 et 2 .

Exercice 29

Créer un graphique avec le titre "Fonction identité VS fonction carrée" sur lequel sont représentées les courbes d'équations $y = x$ et $y = x^2$, pour x entre -1 et 2 .

Réponse :

```
1 x = np.linspace(-1,2,100) # on prend un grand nombre
  de points pour que la courbe ait l'air lisse
2 y1 = x
3 y2 = x**2
4 plt.title("Fonction identité VS fonction carrée")
5 plt.plot(x,y1,label="fonction identité")
6 plt.plot(x,y2,label="fonction carrée")
7 plt.legend()
8 plt.xlabel("abscisses") # facultatif
9 plt.ylabel("ordonnées") # facultatif
10 plt.grid() # facultatif (affiche une grille)
11 plt.show()
```

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue**

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue

J'peux pas, j'ai tortue

Turtle est un module ludique de Python qui permet de faire des dessins sur un écran en donnant des ordres de déplacements à une petite tortue (souvent symbolisée par une flèche).

On importe `turtle` comme ceci

```
1 import turtle
```

Maintenant que la tortue est là, il faut lui donner un nom. Nous avons très envie de l'appeler Franklin ou Michelangelo, mais pour faire simple et court nous allons juste l'appeler `t`, comme tortue.

```
1 t = turtle.Turtle()
```


Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue**

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue

Déplacements

Les quatre types de déplacements fondamentaux sont :

<code>forward(p)</code>	avancer de p pixels
<code>backward(p)</code>	reculer de p pixels
<code>right(d)</code>	tourner à droite de d degrés
<code>left(d)</code>	tourner à gauche de d degrés

Exercice 30

Exécuter le script suivant et observer le résultat


```
1 import turtle
2
3 t = turtle.Turtle()
4
5 t.forward(100)
6 t.right(60)
7 t.forward(100)
8
9 turtle.done()
```

Exercice 30

Exécuter le script suivant et observer le résultat

```
1 import turtle
2
3 t = turtle.Turtle()
4
5 t.forward(100)
6 t.right(60)
7 t.forward(100)
8
9 turtle.done()
```

Réponse : Une fenêtre contenant ceci s'affiche



⇒ La méthode `done` signale à Python qu'on a terminé. On peut éventuellement remplacer la ligne 9 par `turtle.exitonclick()`.

Exercice 31

- Créer un programme qui fait tracer un carré à la tortue.
- Adapter ce programme pour faire un triangle, puis un hexagone.

Exercice 31

- Créer un programme qui fait tracer un carré à la tortue.
- Adapter ce programme pour faire un triangle, puis un hexagone.

Réponse :

```
1 import turtle
2
3 t = turtle.Turtle()
4
5 for i in range(4): # range(3) pour un triangle, range
6     t.forward(100)
7     t.left(90) # t.left(120) pour un triangle, t.left
8     (60) pour un hexagone
9 turtle.done()
```

⇒ Nous pouvons mettre à profit tout ce que nous avons appris en Python jusqu'ici (boucles, conditions, fonctions, etc...).

Autres déplacements

D'autres déplacements existent, comme par exemple avec les méthodes

`goto`(coordonnées)

va à un endroit précis

`circle`(rayon du cercle)

parcourt un cercle

`dot`(diamètre du disque)

crée un disque centré

`home`()

retourne au point de départ

Il est possible de déplacer la tortue sans la faire dessiner, grâce à la méthode `up` , puis la méthode `down` pour la faire dessiner à nouveau.

La position (coordonnées) de la tortue est accessible avec la méthode `pos` .

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue**

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue

Une tortue bien stylée

La rapidité de la tortue se règle grâce à la méthode `speed`(entier entre 1 et 10) 1 étant le plus lent et 10 le plus rapide

La forme de la tortue se modifie avec la méthode `shape`(chaîne de caractères) à choisir parmi "arrow", "turtle", "circle", "square", "triangle", "classic"

L'épaisseur du tracé effectué par la tortue se règle avec la méthode `pensize`(un nombre positif) vaut 1 si non précisé

La couleur du tracé se modifie avec la méthode `pencolor`(couleur) par exemple "blue"

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain**

- Colorier les figures réalisées

- Cloner sa tortue

Modifier le décor

Il n'y pas que la tortue `t` que l'on peut changer, mais aussi l'arrière plan. Par exemple, la couleur du fond se modifie par la méthode `bgcolor(couleur)`

Le titre de la fenêtre se change par la méthode `title(chaine de caractères)`

 Ces deux méthodes s'appliquent à `turtle` et non `t`.

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées**

- Cloner sa tortue

Remplissage par une couleur

Pour remplir une forme d'une couleur, on encadre les instructions qui produisent la forme, comme ceci :

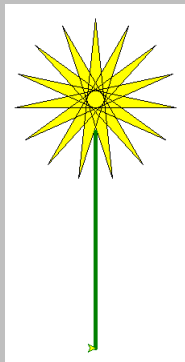
```
t.begin_fill()  
déplacements de la tortue  
t.end_fill()
```

Remarque : La position initiale n'a pas besoin d'être la même que la position finale pour le remplissage. En fait, le remplissage s'effectue automatiquement à la fin comme si la tortue revenait directement au point de départ.

La couleur du remplissage se modifie en appliquant à `t` la méthode `fillcolor(couleur)`

Exercice 32

Faire un programme qui dessine un pissenlit comme celui-ci



Exercice 32

Faire un programme qui dessine un pissenlit

Réponse :

```
1 import turtle
2
3 t = turtle.Turtle()
4
5 t.fillcolor("yellow") # couleur de remplissage
6 t.begin_fill() # début de la fleur à colorier
7 for i in range(15):
8     t.forward(200)
9     t.left(168) #  $180-168=12$ , et  $15*12=180$ 
10 t.end_fill() # fin du coloriage
11 t.up() # on soulève la tortue
12 t.goto(100.00,-30) # déplacement au début de la tige
13 t.down() # on pose la tortue
14 t.pencolor("green") # couleur du trait
15 t.pensize(5) # épaisseur du trait
16 t.goto(100.00,-300.00) # jusqu'en bas de la tige
17
18 turtle.done()
```

Plan

Matplotlib

- Tracer des courbes

- Quelques options d'affichage

Turtle

- Nommer sa tortue

- Promener sa tortue

- Customiser sa tortue

- Adapter le terrain

- Colorier les figures réalisées

- Cloner sa tortue**

Faire un clone

Pour avoir une autre tortue, il suffit de cloner `t`. Cela peut-être utile par exemple pour éviter de devoir alterner entre plusieurs styles pendant un tracé élaboré, ou bien pour des jeux comme dans le projet final proposé sur cette page <https://realpython.com/beginners-guide-python-turtle/>.

```
1 u = t.clone()
```

Beaucoup d'autres fonctionnalités (comme par exemple les *événements*, ou l'écriture de texte) sont décrites dans la documentation de `turtle`.

<https://docs.python.org/3/library/turtle.html>

Conclusion

Dans cette séance nous avons parcouru diverses possibilités de représentations graphiques avec les modules

- ▶ `matplotlib`
- ▶ `turtle`

Dans la prochaine séance, nous verrons comment importer des données, et visualiser celles-ci à l'aide d'histogrammes et diagrammes circulaires. Puis, nous nous confronterons à des exercices sous forme de problèmes, que nous tenterons de résoudre à l'aide des techniques acquises des quatre premières séances.

BONUS

Un dernier petit script à exécuter avant de se quitter :

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 t = np.linspace(0,2*np.pi,1000)
5 x = np.cos(t)
6 y = np.sin(t) + np.abs(x)**0.5
7 plt.axis("equal")
8 plt.plot(x,y,'r')
9 plt.show()
```