

A. Des instructions disponibles directement

	Syntaxe	Exemple		Syntaxe	Exemple
addition	+	<pre>>>> 2 + 5 7</pre>	division	/	<pre>>>> 14 / 3 4.666666666666667</pre>
soustraction	-	<pre>>>> 2 - 5 -3</pre>	arrondir x à 10^{-n} près	round(x,n)	<pre>>>> round(2.56,1) 2.6</pre>
multiplication	*	<pre>>>> 2 * 5 10</pre>	maximum	max(a,b,...)	<pre>>>> max(2, 5, 3) 5</pre>
puissance	**	<pre>>>> 2 ** 5 32</pre>	minimum	min(a,b,...)	<pre>>>> min(2, 5, 3) 2</pre>
quotient et reste dans la division euclidienne	// (quotient) % (reste)	<pre>>>> 14 // 3 4 >>> 14 % 3 2</pre>	= et ≠	== et !=	<pre>>>> 21 / 2 == 10.5 True</pre>
			< , > ≤ , ≥	< , > ≤ , ≥	<pre>>>> 22 / 7 < 3 False</pre>










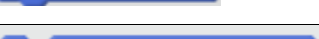




B. Avec les bibliothèques math ou random

	<pre>>>> from math import*</pre>
Nombre π .	<pre>>>> pi 3.141592653589793</pre>
Racine carrée d'un nombre.	<pre>>>> sqrt(2) 1.4142135623730951</pre>
Partie entière d'un nombre.	<pre>>>> floor(5.72) 5</pre>
Cos, sin, tan d'un angle mesuré en degrés.	<pre>>>> cos(radians(30)) 0.8660254037844387</pre>
Angle en degrés connaissant son cosinus, sinus ou sa tangente (analogues de \cos^{-1} , \sin^{-1} , \tan^{-1}).	<pre>>>> degrees(acos(0.8)) 36.86989764584401</pre>

	<pre>>>> import random</pre>
Nombre entier aléatoire de l'intervalle [a,b] (de [1 ;6] dans l'exemple). Analogue de ALEA.ENTRE.BORNES() sur le tableur.	<pre>>>> random.randint(1, 6) 4</pre>
Nombre réel au hasard dans l'intervalle [0 ; 1[. Analogue de ALEA() sur le tableur.	<pre>>>> random.random() 0.25236135296415485</pre>
Un élément tiré au hasard dans une liste L.	<pre>>>> L=[0, 5, 10, 15, 20] >>> random.choice(L) 10</pre>

C. Avec la bibliothèque turtle

On importe la bibliothèque par `import turtle` pour avoir accès aux commandes ci-après (et à beaucoup d’autres).

	Scratch	Actions	Python
Gestion du crayon et de la tortue		Cacher la tortue.	<code>turtle.hideturtle()</code>
		Montrer la tortue.	<code>turtle.showturtle()</code>
		Effacer tout.	<code>turtle.clear()</code>
		Effacer tout, recentrer la tortue, l’orienter vers la droite. Les variables sont aussi remises à 0.	<code>turtle.reset()</code>
		Abaisser le crayon pour pouvoir dessiner.	<code>turtle.down()</code>
		Relever le crayon.	<code>turtle.up()</code>
		Choisir la couleur du trait.*	<code>turtle.pencolor(couleur)</code>
		Choisir l’épaisseur du trait.	<code>turtle.width(épaisseur)</code>
		Remplir la forme qui va être tracée/ Arrêter le remplissage.	<code>turtle.begin_fill(couleur choisie) / turtle.end_fill()</code>
		Écrire un texte à la position de la tortue.	<code>turtle.write(texte)</code>
Mouvement		Orienter la tortue.	<code>turtle.setheading(0)</code> (0 ou 90 ou 180 ou 270)
		Aller à la position de coordonnées (x,y).	<code>turtle.goto(x,y)</code>
		Obtenir la position de la tortue.	<code>turtle.position()</code>
		Avancer d’un certain nombre de pas.	<code>turtle.forward(nombre de pas)</code>
		Reculer d’un certain nombre de pas.	<code>turtle.back(nombre de pas)</code>
		Tourner à droite d’un angle donné.	<code>turtle.right(mesure de l’angle)</code>
		Tourner à gauche d’un angle donné.	<code>turtle.left(mesure de l’angle)</code>
		Suivre un arc de cercle à partir de sa position de rayon r, d’un angle a (positif ou négatif selon la position du centre).	<code>turtle.circle(r,a)</code>
		Choisir la vitesse de la tortue : speed(1) lent, speed(10) rapide.	<code>turtle.speed(nombre entre 1 et 10)</code>

* Couleurs prédéfinies : ‘blue’, ‘red’, ‘green’, ‘yellow’, ‘orange’, ‘purple’, ‘brown’, ‘black’, ‘white’, ‘pink’, ‘grey’

À l’exécution, une fenêtre graphique s’ouvre (elle peut être réduite dans la barre des tâches). On finira le programme par l’instruction `turtle.exitonclick()` et il suffira de cliquer sur la fenêtre graphique pour la fermer.

Remarques

- un bug subsiste dans certaines versions de Python quand on relance le programme... il suffit de le relancer une fois de plus.
- on peut travailler avec plusieurs tortues auxquelles on peut donner des noms pour les identifier.

D. Avec matplotlib.pyplot

On choisit de créer un alias (plt) pour raccourcir l'écriture et faciliter la lisibilité des instructions.

		<code>>>> import matplotlib.pyplot as plt</code>
Repère	Choisir un repère orthonormé.	<code>plt.axis('equal')</code>
	Choisir [xmin, xmax, ymin, ymax].	<code>plt.axis([0, 6, 0, 20])</code>
	Afficher la grille.	<code>plt.grid()</code>
	Indiquer une légende sur l'axe des abscisses ou des ordonnées.	<code>plt.xlabel('temps en h')</code> <code>plt.ylabel('distance en km')</code>
Points	Placer le point A (xA ; yA).	<code>plt.plot([xA], [yA])</code>
	Marquer A(xA ; yA) par un petit point rouge.	<code>plt.plot([xA ; yA], 'r.')</code>
	Marquer A(xA ; yA) par un gros point bleu.	<code>plt.plot([xA], [yA], 'bo')</code>
	Marquer A(xA ; yA) par un gros carré bleu (bs = blue square).	<code>plt.plot([xA], [yA], 'bs')</code>
	Marquer B(xB ; yB) par un triangle vert (g^ = green triangle).	<code>plt.plot([xB], [yB], 'g^')</code>
	Marquer C(xC ; yC) par un trait en pointillés rouge.	<code>plt.plot([xC], [yC], 'r--')</code>
Ligne	Placer les points A(xA ; yA), B(xB ; yB), C(xC ; yC) et D(xD, yD).	<code>plt.plot([xA, xB, xC, xD], [yA, yB, yC, yD])</code>
	Tracer le segment [AB].	<code>plt.plot([xA, xB], [yA, yB], '-')</code>
	Tracer le segment [AB] en rouge.	<code>plt.plot([xA, xB], [yA, yB], 'r-')</code>
Figure	Créer une liste vx de valeurs de t. Placer les points (t, t²) pour créer une courbe.	<code>vx = [t/10 for t in range(-5, 55)]</code> <code>vy = [t**2 for t in vx]</code> <code>plt.plot(vx,vy)</code>
	Afficher la figure.	<code>plt.show()</code>
Figure	Créer différentes figures.	<code>plt.figure(1)</code> <i>instructions</i> <code>plt.figure(2)</code> <i>instructions</i>

Exemple

```

1 import matplotlib.pyplot as plt
2
3 plt.grid()
4 # liste des nombres de 0 à 5 avec un pas de 0,1
5 vx = [t/10 for t in range(0, 51)]
6 # liste des carrés des nombres de la liste vx
7 vy = [t**2 for t in vx]
8 plt.plot(vx,vy, 'r-')
9 plt.plot([0, 5], [0, 25], 'b--')
10 plt.show()

```

