Le module Kandinsky

Le module graphique *Kandinsky* est présent dans la Numworks, de même que les modules math (fonctions mathématiques usuelles), cmath (nombres complexes) et random (nombres aléatoires). La particularité de ce module est d'être une invention de Numworks : ce module n'est pas proposé par les distributions habituelles de Python qui propose à sa place le module Tkinter (beaucoup plus complet). Dans ce module, il y a quatre fonctions.

3.4. Le module kandinsky

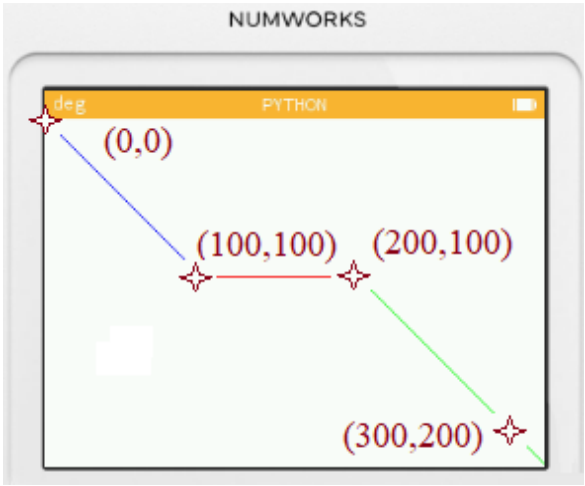
Voici la description exhaustive du module `kandinsky`. Vous pouvez avoir cette liste sur votre calculatrice en appuyant sur (Toolbox) et en allant dans Modules puis dans kandinsky.

Fonction	Explication
`color(r,g,b)`	Génère la valeur de la couleur `r`, `g`, `b`
`get_pixel(x,y)`	Renvoie la couleur du pixel aux coordonnées `x`, `y`
`set_pixel(x,y,color)`	Allume le pixel `x`, `y` de la couleur `color`
`draw_string(text,x,y)`	Affiche le texte `text` aux coordonnées `x`, `y`

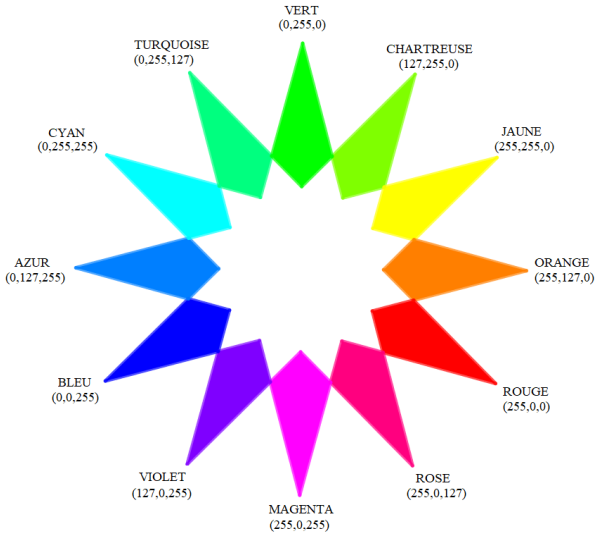
Changer la couleur d'un pixel :

L'écran de la calculatrice est composé de 320 pixels sur sa longueur et de 222 pixels sur sa hauteur, ce qui fait 71040 pixels en tout. On peut fixer la couleur de chacun de ces pixels en exécutant l'instruction set_pixel(x,y,color). Pour cela, il suffit d'indiquer

- les coordonnées (x,y) du pixel
- la couleur color(r,g,b)



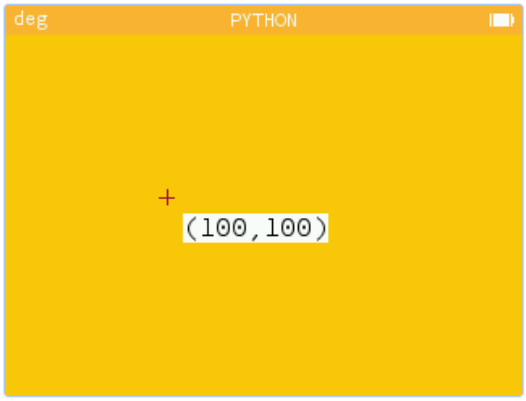
La couleur est définie selon le système rouge, vert, bleu (rgb = *red*, *green*, *blue*) qui nécessite d'indiquer une valeur entière comprise entre 0 et 255 pour chacun des trois paramètres. Si on indique color(255,0,0) on obtient un rouge pur, pour un gris clair, ce sera par exemple color(150,150,150) et un pourpre color(158,14,64). Le noir est color(0,0,0) et le blanc, à l'opposé, est color(255,255,255). Le cercle chromatique ci-contre donne davantage d'exemples mais, pour une nomenclature plus complète, consulter cette page ou bien une des innombrables applications donnant les codes couleur. Avec ces deux premières instructions, il est donc possible de dessiner à peu près ce qu'on veut, de la couleur qu'on veut. Il suffit de dire quels pixels colorer et comment. Pour tracer la ligne brisée de l'illustration, j'ai écrit le programme suivant :




```
1  from kandinsky import *
2  for i in range(100):
3    set_pixel(i,i,color(0,0,255))
4    set_pixel(i+100,100,color(255,0,0))
5    set_pixel(i+200,i+100,color(0,255,0))
6
```

En faisant varier les cinq nombres entiers x, y, r, g et b dans leur domaine respectif, on obtient une des $320\times222\times256^3$ (environ 1200 milliards) images fixes que ce système graphique permet de créer.

Écrire un texte :



L'instruction draw_string(texte,x,y) permet d'écrire un texte dans l'image, toujours en noir sur fond blanc, la taille et la police des caractères étant immuables. C'est minimaliste, ce qui n'a pas que des inconvénients.
Le rectangle dans lequel est inscrit le texte est repéré par les coordonnées de son sommet supérieur gauche. Dans l'image ci-contre, j'ai tracé un + pour indiquer le point de coordonnées (100,100) mais j'ai dû écrire le texte des coordonnées en le décalant pour ne pas qu'il chevauche le dessin du + : son sommet supérieur gauche a, ici, les coordonnées (110,110).

Capturer la couleur d'un pixel :

L'instruction get_pixel(x,y) renvoie la couleur du pixel situé aux coordonnées (x,y). Je me demande bien à quoi peut servir cette instruction mais, comme elle représente à elle seule 25% des instructions du module, je suppose que cela en vaut la peine. Après quelques expérimentations, j'essaie c=get_pixel(0,0) et écrit sur l'écran la conversion de c en chaîne de caractères (on ne dispose plus de la console dès lors qu'un pixel a été allumé). J'obtiens un nombre ! Si la couleur est (0,0,255) j'obtiens 31, avec (0,255,0) j'obtiens 2016 et avec (255,0,0) j'obtiens 63488. Le nombre varie entre 0 (noir) et 65535 (blanc). Je m'aperçois que get_pixel(x,y) renvoie un nombre. Je peux lui ajouter un autre nombre et employer le résultat comme couleur, je peux même déclarer une couleur avec un tel nombre. Par exemple, si c=63488, l'instruction set_pixel(0,0,c) est correcte. Pour faire du blanc, je peux partir d'un jaune 65504 qui correspond à (255,255,0) et lui ajouter le bleu intense 31, ce qui donne 65535. Retrouver les composantes rgb à partir du nombre n obtenu par get_pixel est assez technique, mais il faut noter que les 256 valeurs de l'intensité d'une couleur ne sont pas différentes : de 0 à 7 l'intensité vaut 0, de 8 à 15 elle vaut 1, ainsi de suite jusqu'aux valeurs de 248 à 255 où l'intensité vaut 31. Il n'y a donc que 32 intensités différentes pour chacune des trois composantes.

Exercices :

1) Écrire un programme qui réalise un dégradé horizontal entre deux couleurs quelconques.
2) Écrire un programme qui place des points aléatoirement dans l'écran, la couleur de ces points étant fonction de la distance au centre de l'écran, le point de coordonnées (160,111).
3) Écrire un programme qui trace un échiquier (un quadrillage de 8 lignes et 8 colonnes), les cases étant alternativement blanches et noires, sur fond vert.
4) Écrire un programme qui trace un rectangle de largeur L et de hauteur h, au centre de l'écran, sa bordure étant d'une couleur c1 et d'une épaisseur e et son intérieur étant d'une couleur c2.
5) Écrire un programme qui trace un cercle de centre O(x,y), d'épaisseur e et de rayon R.
6) Écrire un programme qui trace une horloge dont les deux (ou trois) aiguilles indiquent une heure donnée à la minute près (ou à la seconde près).
7) Écrire un programme qui trace une de ces rosaces qui ne sont constituées que de cercles (les deux premières sont classiques, la 3ème est appelée « fleur de vie » et la dernière, trouvée sur internet, y est appelée spirale de Georgia).