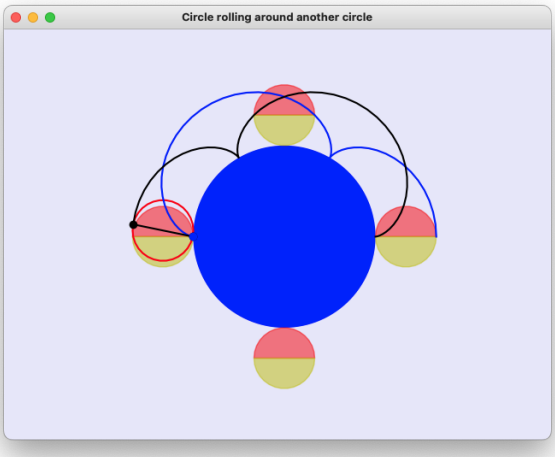
Python 23-24 for dummies : problème 1
Un cercle roulant sans glisser autour d'un autre cercle !



Nous allons calculer la trajectoire d'un cercle de rayon unitaire roulant sans glisser le long d'une cercle d'un rayon entier α quelconque. On observera ainsi que le petit cercle effectuera un nombre $\alpha+1$ de rotations dans ce mouvement.

Sur la figure, on observe le mouvement du petit cercle pour une valeur de $\alpha = 3$. Le petit cercle a donc la même orientation pour les quatre points cardinaux. On y a aussi tracé la trajectoire de deux points du petit cercle. Paradoxalement, le cercle tourne localement autour de son centre de masse 3 fois et il convient d'y ajouter la rotation globale effectuée. Une autre manière de comprendre ce paradoxe est de noter que le centre de masse du petit cercle se balade en fait sur un cercle de rayon $\alpha = 1$.

Dans ce premier devoir, il s'agira de vous familiariser avec les tableaux `numpy` et d'implémenter des opérations élémentaires sur ce dernier. Il s'agira aussi de découvrir comment il est possible d'obtenir une animation avec `matplotlib`. Ce n'est pas bien compliqué, mais cela peut paraître un peu mystérieux :-)

Plus précisément, on vous demande de :

1. Ecrire une fonction

   ```
   [theta,x,y] = circlesCreate(radius,n)
   ```

   qui construit une discrétisation d'un cercle (Θ_k, X_k, Y_k).

   Les arguments `radius` et `n` sont respectivement le rayon du cercle centré à l'origine et le nombre de valeurs à calculées. On divisera le cercle en un nombre $n-1$ d'arcs égaux et un nombre n d'angles et de points. Le premier et le dernier point coïncideront. La fonction renverra trois tableau `numpy` de dimension n contenant les angles et les deux composantes cartésiennes des points. Le premier élément correspondra à l'angle nul et le dernier élément à l'angle 2π.

2. Ensuite écrire une fonction

   ```
   thetas = circlesAngles(ratio)
   ```

   qui calcule les angles du point de contact du petit cercle sur le grand cercle où ce dernier a la même orientation qu'à la condition intiale avec un angle nul. On parcourt bien le grand cercle de manière anti-horlogique en partant de la position la plus à droite qui correspond à 15 heures :-)

   L'argument `ratio` est un entier positif qui est le rapport entre les deux rayons. Une valeur nulle est admise. La fonction renverra un tableau `numpy` contenant l'ensemble des angles requis en y incluant la valeur nulle initiale. Attention : même si on présente un exemple avec un ratio $\alpha = 3$ dans la figure, il faut écrire un programme pour un ratio quelconque ! :-)

3. Finalement, écrire une fonction

   ```
   [x,y] = circlesAnimate(theta,x_rolling,y_rolling,ratio)
   ```

   qui calcule les nouvelles positions des points `x_rolling`, `y_rolling` du petit cercle lorsque son point de contact est caractérisé par un angle `theta` sur le grand cercle. Le dernier argument `ratio` est toujours le rapport entre les deux rayons.

4. Pour tester votre programme, on vous a fourni un tout petit programme `circlesTest.py`.
   Ce petit programme devrait vous permettre d'écrire et de tester votre code avec votre ordinateur

   ```
   ratio = 4
   n_steps = 31
   n = (ratio+1)*n_steps+1
   i_mid = n_steps//2

   [theta_inner,x_inner,y_inner] = circlesCreate(ratio,n)
   [theta_rolling,x_rolling,y_rolling] = circlesCreate(1,n_steps)

   import matplotlib.pyplot as plt
   from matplotlib.animation import FuncAnimation

   fig = plt.figure("Circle rolling around another circle")
   x_left  = np.ones_like(x_inner) * np.nan
   y_left  = np.ones_like(y_inner) * np.nan
   x_right = np.ones_like(x_inner) * np.nan
   y_right = np.ones_like(y_inner) * np.nan
   line1, = plt.plot([], [], 'r-')
   line2, = plt.plot([], [], 'b-')
   line3, = plt.plot([], [], 'k-')
   plt.fill(x_inner,y_inner, 'b-')
   for theta in circlesAngles(ratio) :
       [x,y] = circlesAnimate(theta,x_rolling,y_rolling,ratio)
       plt.fill(x[0:i_mid+1],y[0:i_mid+1],color='r',alpha=0.5)
       plt.fill(x[i_mid:],y[i_mid:],color='y',alpha=0.5)

   def animate(i):
       [x,y] = circlesAnimate(theta_inner[i],x_rolling,y_rolling,ratio)
       x_left[i]  = x[0]
       y_left[i]  = y[0]
       x_right[i] = x[i_mid]
       y_right[i] = y[i_mid]
       line1.set_data(x,y)
       line2.set_data(x_left,y_left)
       line3.set_data(x_right,y_right)
       return line1,line2,line3,

   animation = FuncAnimation(fig,animate,frames=n,interval=200)
   plt.gca().set_aspect('equal')
   plt.axis('off')
   plt.show()
   ```

   Au passage, c'est une bonne idée de changer la valeur de `ratio` lors de vos tests !
   Observer bien aussi la manière dont on obtient une animation avec `matplotlib` !

5. Vos trois fonctions seront soumises via le site web du cours.

6. Attention : il ne faut pas recopier le programme de test `main` dans votre soumission : uniquement les deux fonctions que vous avez écrites. Vérifier bien que votre programme fonctionne correctement sur le serveur et pas uniquement sur votre ordinateur : aucun recours ne sera valable si le devoir n'est pas exécuté correctement sur le serveur. Pour rappel, toutes vos soumissions seront systématiquement analysées par un logiciel anti-plagiat. Faites vraiment votre programme seul... en vous inspirant uniquement des programmes fournis par l'enseignant :-)
   Ange et Nathan veillent au grain !