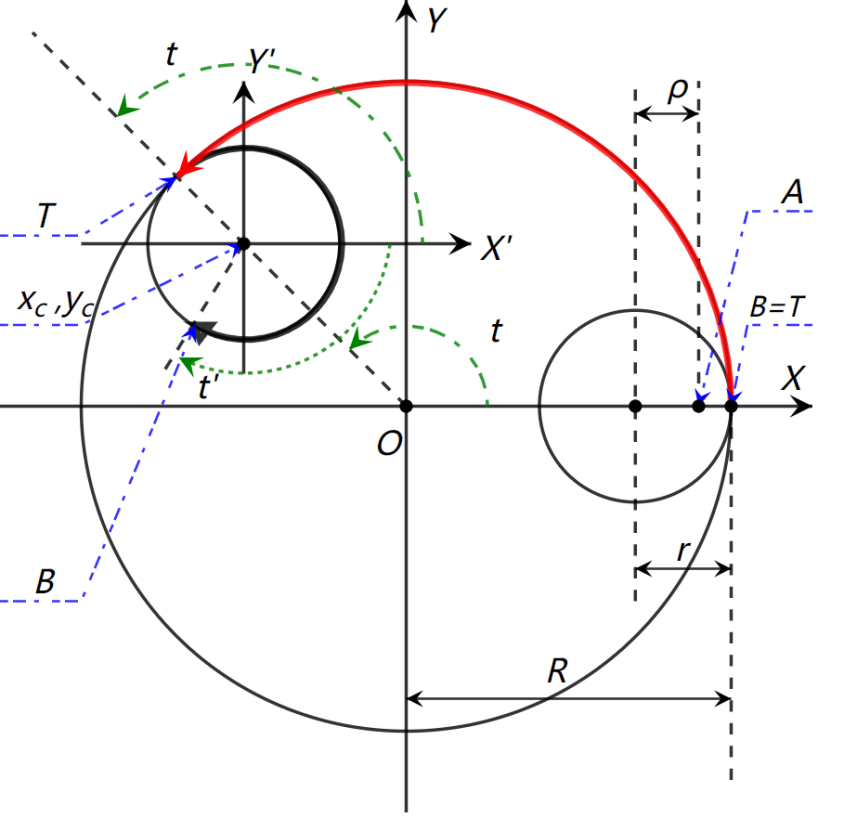
Python极客项目编程笔记

# 一、万花尺

## （一）理论基础

设外圆圆心为，内圆圆心为**，笔尖为点，是外圆上随滚动而变化的点，是内圆上随滚动而变化的点。

记内圆半径为，外圆半径为，，，在初始状态，点，，，都在轴上，并且点与点重合。

笔尖带动内圆贴着外圆滚动时，的位置将会发生变化，记与轴的夹角为，现规定：逆时针旋转时，增加，顺时针旋转时，减少。

现在需要求出当笔尖移动时，点的轨迹，实际上就是求出关于的的坐标方程。

为了求解，建立两个直角坐标系：坐标系以外圆圆心为原点；坐标系以内圆圆心为原点。在坐标系下的点的坐标表示为，在坐标系下的点的坐标表示为，点的坐标分量均采用极坐标系记法。

目标转换为：当内圆滚动了角度后，求。

求解思路：在坐标系下根据求，再根据求，最后使用坐标系转换公式求得

当内圆滚动了角度后，容易求得并且在外圆经过的弧长为，也在内圆上**顺时针**经过了相同长度的弧长，记在内圆上的角度变化了，那么有：

那么当前的角度为，于是，那么。

坐标系变换到坐标系的公式为：

由此可以计算出，简化为：

方程中的关键参数：

：影响图形整体大小

：影响内圈大小

：影响图形形状

**至少转多少圈才能完整地绘制出图形？**

记和的最大公约数是，那么至少需要转圈后才能完整地绘制出图形，也就是说，的变化范围是。

## （二）使用turtle实现万花尺

编写Spiro类，该类的一个对象就代表一个万花尺图形，写成类是为了方便后续在创建多个万花尺图形时更方便。

类实例化时接收参数：

：外圆的半径

：内圆的半径

：笔尖离内圆圆心的距离与内圆半径的比值

在构建函数\_\_init\_\_中，会根据这三个参数计算好、等，同时还会将初始化绘图状态，包括调用实例方法restart()

1.实例方法get\_x\_y(self, t)

该方法通过公式来计算角度时候的笔尖所在点的横坐标、纵坐标

2.实例方法setheading(self, x, y)

该方法用于调整海龟的朝向，核心是使用turtle.setheading函数，一般来说，不调整海龟朝向也可以图形，海龟将默认朝轴正方向，但动画不够形象。

该方法需要根据当前位置turtle.pos()，和即将到达的位置来计算方向向量与轴的**方向角**，然后使用turtle.setheading来调整海龟的朝向。

该方法实现时有一些事需要注意：

1.计算方向向量时，可能会遇到零向量的情况，此时计算方向角将会导致除以0的异常，需要做特殊处理，例如直接跳出函数

2.获取方向角时，可能会出现获取到的方向角仅有范围，此时可以判断向量是否位于第三第四象限（方向向量纵坐标小于0）来纠正方向角

3.实例方法restart(self)

该方法调整整个绘图的状态为初始状态，包括：

·显示画笔

·提起画笔，并调整海龟朝向、移动到初始位置

4.实例方法draw(self)

此方法负责画出完整的万花尺图形，主要做法是取在内的离散点，就是角度，对于每一个角度，使用实例方法get\_x\_y获取即将到达的位置，使用实例方法setheading调整海龟朝向，然后使用turtle.setpos来让海龟到达新位置。结束后隐藏海龟画笔。

其中需要注意的是取离散点的步长设置，步长设置太小的话绘制速度慢，太大则曲线不够精细，可以设置为5。

5.类的使用方法

spiro = Spiro(R, r, l)

spiro.draw()

## （三）使用matplotlib实现万花尺

turtle库画万花尺图形很慢，如果想让画图的速度更快，可以使用matplotlib库。

编写Spirograph类来实现matplotlib画万花尺图形，类在实例化时需要接收三个参数：、、，对应公式中的参数，在构造方法\_\_init\_\_中，计算公式相关的变量。

因为要使用matplotlib.plot.plot方法画多个折线来实现，所以类实例变量还需要有self.before\_x和self.before\_y，记录上一次画折线后停留的位置。

类的实例方法self.get\_x\_y与Spiro类的一样，都是给定角度，返回笔尖的横坐标和纵坐标。

1.实例方法coord\_generator(self)

该方法是一个生成器，用于根据一个完整的角度范围，给定笔尖的横坐标和纵坐标，基本做法是按一个步长在中生成离散的，然后使用self.get\_x\_y计算坐标，并返回，步长默认设置为3。

2.实例方法draw(self)

该方法用于绘图，步骤如下：

1.初始化：生成画布，让self.before\_x和self.before\_y置为初始位置

2.使用self.coord\_generator生成坐标，并使用plt.plot画折线，更新self.before\_x，self.before\_y

3.设置xlim和ylim

4.让坐标系适应图像大小，使用plt.axis('image')，并隐藏坐标系

5.保存画布、显示画布

# 二、Conway生命游戏

## （一）游戏规则

游戏在一个N×N的二维网格网络中进行，每一个网格代表一个细胞，细胞只有两种状态：ON/OFF，细胞周围的8个邻居影响细胞的状态。

1.细胞状态变化

记一个细胞的状态为，使用代表OFF状态，代表ON状态

记邻居细胞中为ON的数量为，显然

对于一个细胞，它的状态根据以下规则来变化：

1.当时，的下一个状态

2.当时，的下一个状态

这些规则是为了表示，当生物种群太少或太多时都会杀死细胞，只有当种群比较平衡时细胞才会继续繁衍。

2.处理边界

对于处在边界中的细胞，如果邻居超出了上边边界，则在下方边界出现，如果超出了左边边界，则在右方边界出现，以此类推，这样就能保证每个细胞都有8个邻居。

## （二）编写逻辑

首先要把游戏的各个要素用编程来实现，编写Cell类来代表细胞，CellNetwork来代表一个细胞网格网络。

1.用Cell类代表细胞

一个细胞应该包含的基本属性：所在行、所在列、当前状态。所以Cell类的实例变量应该有这三个，在编写逻辑阶段，Cell类只需包含这么多要素即可，后续在编写界面阶段将会实界面随着Cell类实例的状态改变而发生变化。

2.用CellNetwork类表示细胞网络

一个细胞网络应当包含网络的大小，并且在实例化时提供随机生成细胞功能。

细胞网络还应该包含更新一个细胞的状态的方法，更新步骤：

1.计算邻居的ON数量，对于超出边界的邻居，可以使用取余运算来获取下标

2.根据状态更新规则来更新一个细胞的状态

为了简化编码，在更新所有细胞时，按照先行后列的方式遍历细胞网络，逐个更新细胞

## （三）编写界面