Tools文件分为两大部分，一部分是对视频进行处理，一部分是通过制定的规则对视频进行判断从而返回满足的结果。

**第一部分：**

首先读取args中的视频地址，然后利用args中的参数进行判断是capture还是match操作。

如果是capture，那么调用save\_match\_capture(path:str, video\_file:str)函数，代码内容为：

1. 打开地址中的视频，并读取视频的每一帧
2. 利用cv2.selectROI函数让用户在图片上画框
3. 提取框的坐标信息后进行裁剪并展示裁剪后的图片
4. 保存结果图

如果是match操作，那么根据args选择是否要匹配所有模板，然后调用match\_template(path, video, match\_all)函数。这个函数代码内容为：

1. 定义目标像素，然后取出一个模板图像或者当前模板图像路径下的所有模板图像。
2. 打开视频文件，并且一帧一帧读视频，将每一张图片调整像素和转换为灰度图像。
3. 调用cv2.matchTemplate函数将视频帧和模板进行匹配，并且读出匹配结果最好的位置坐标。
4. 使用cv2.rectangle函数根据模板大小在视频帧上面画出匹配的矩形框并调用put\_text函数在中心标出坐标数值。
5. 调用cv2.imshow函数显示当前帧，并通过key变量储存键盘输入的变量，如果为空格，则暂停视频并显示视频帧，当再次敲空格，则继续播放。

**第二部分：**

这一部分代码在主函数中并没有调用，可能在别的文件中使用。

首先compute函数用于计算关键点和关键点直接的距离和角度，用于判断是否满足制定的6个规则。这其中，compute\_angle函数输入是A、B、C、D四个点的坐标，输出是计算AB和CD之间的夹角度数；compute\_dist函数输入是P、A、B三个点的坐标，如果B为空则是计算P和A两点的距离，如果B不为空，则是计算P点到AB向量的距离。

接着使用decide函数对compute函数计算的结果进行判断。这其中，输入是degdists列表，这其中有三帧的计算结果，用于判断前后几帧运动过程中是否满足指定的规则。最后返回的值是一个含有6个值的列表，用以表达是否满足某一个规则。

最后put\_decision函数输出结果，这其中使用了自定义的put\_text\_shading函数来画底纹框和绘制文本。最后在图像上写入的文本就是各项参数和是否满足6个规则这些信息。

（cv2.putText函数仅在给定的图像（通常是一个numpy数组）上绘制文本，但它并不会显示这个图像。要在屏幕上显示图像，需要使用cv2.imshow函数）