گزارش‌کار تمرین سوم سیستم‌های نهفته و بیدرنگ

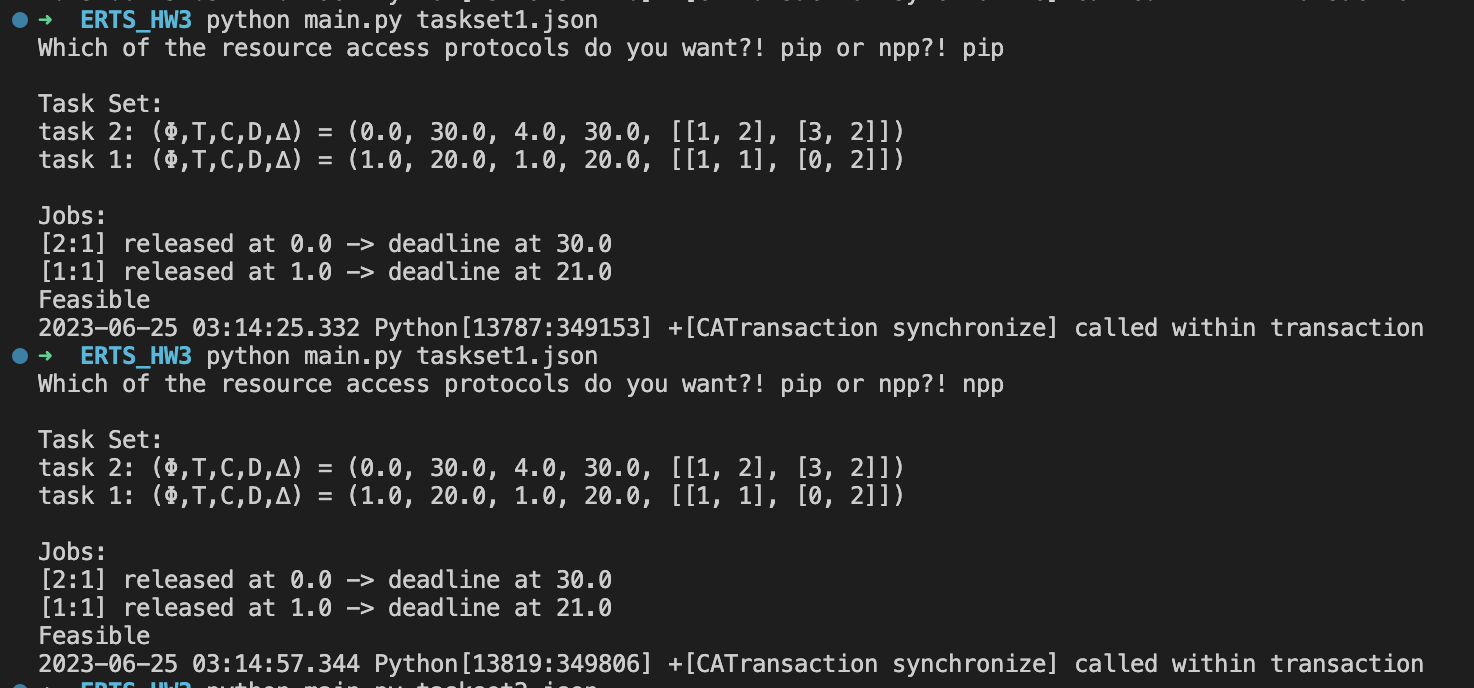
امیررضا طربخواه ۹۸۳۱۰۴۱

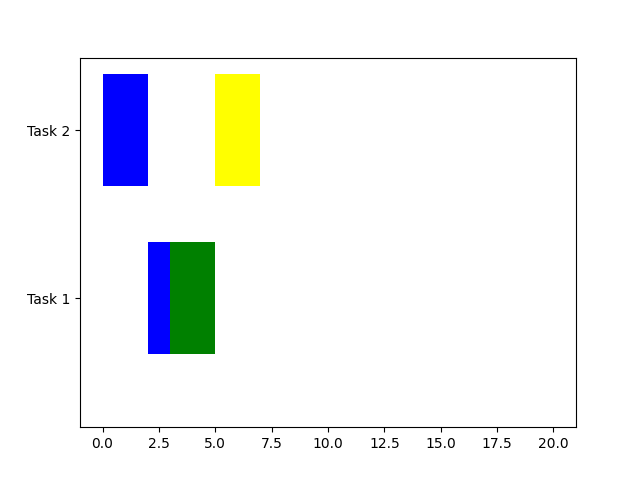
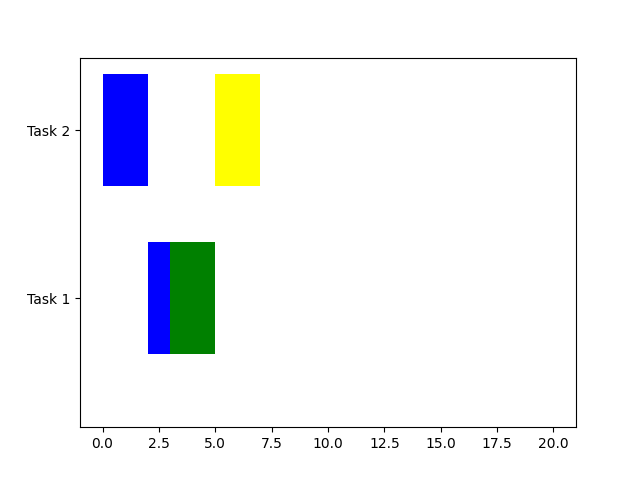
گزارش کد:

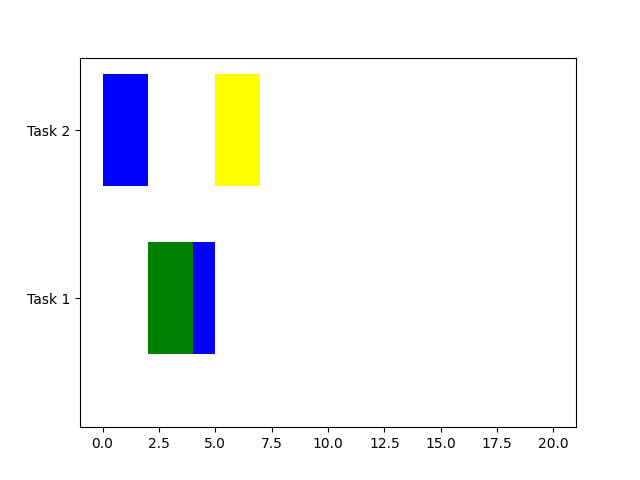
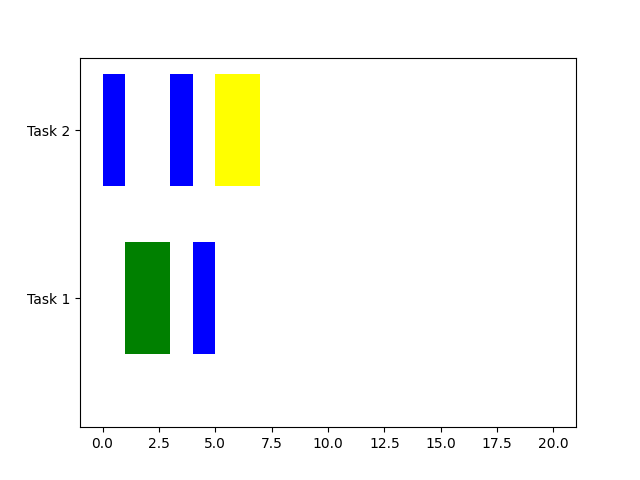
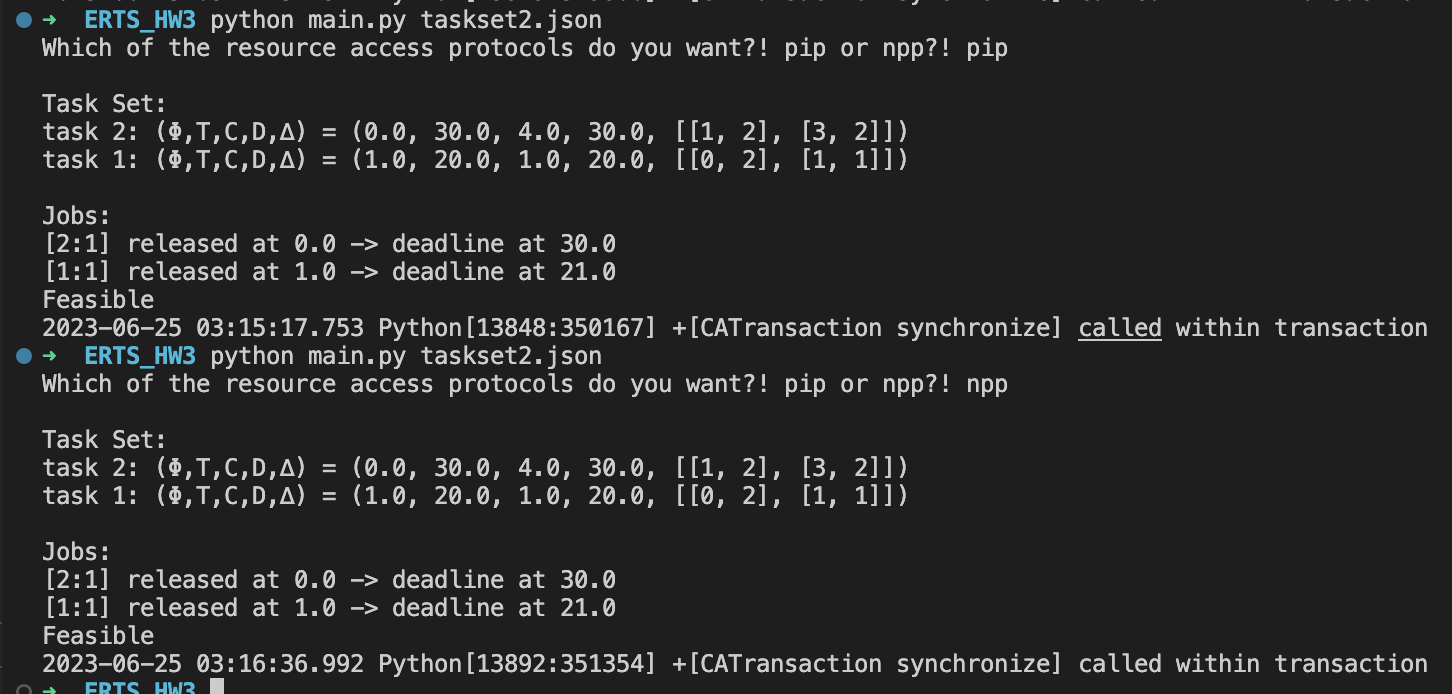
برای پیاده‌سازی تمرین از دو کلاس queue و scheduler استفاده کردم. در queue هر بخش از job که متعلق به یک سمافور است را به عنوان یک sub job در نظر گرفته و عملیات scheduling را روی آنها پیاده‌سازی میکنیم. هر sub job متشکل از یک لیست است که در کامت کد جزئیات آن وارد شده است.

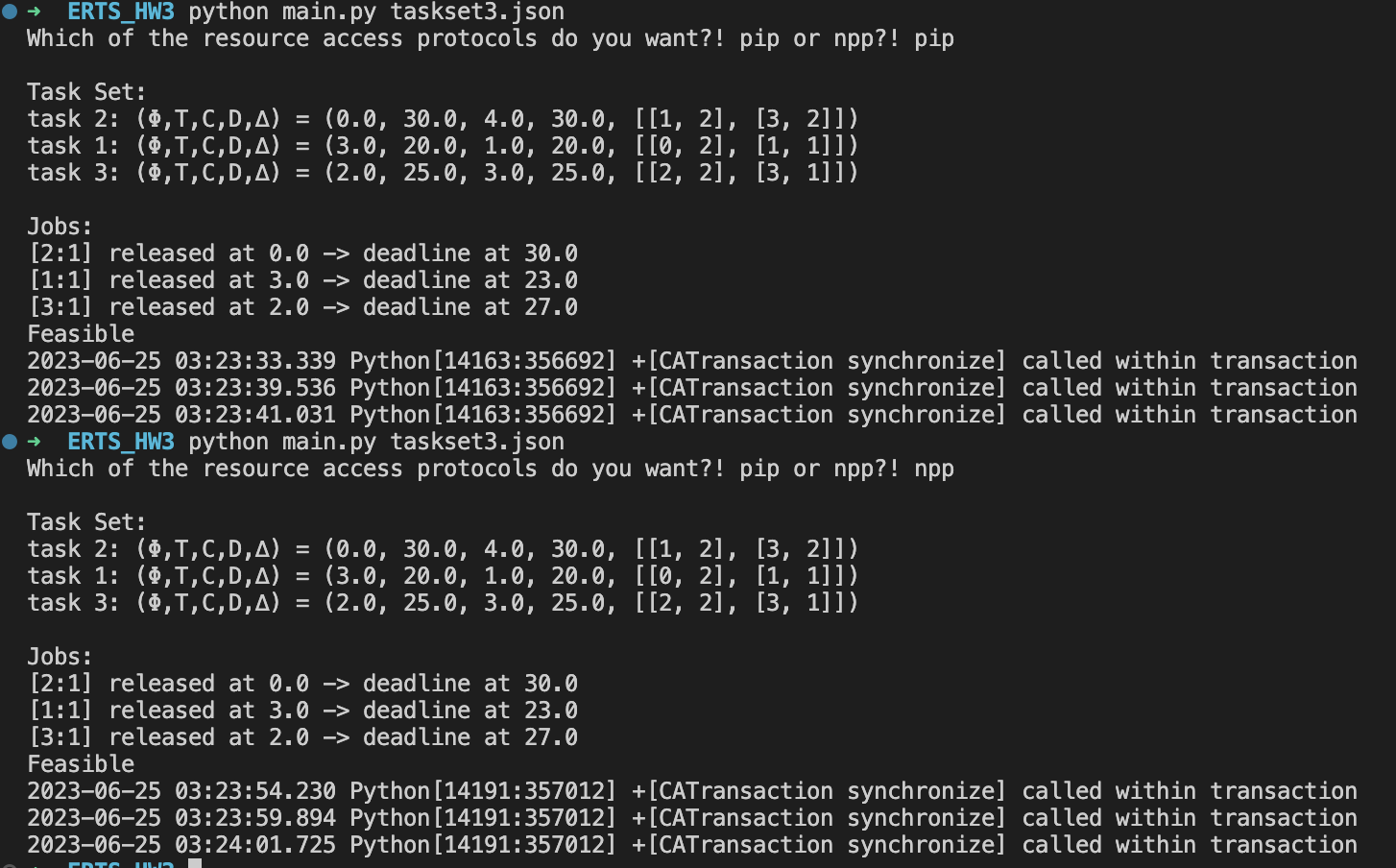
در ابتدا با استفاده از تابع setSubJobs تمامی sub jobهایی که داریم را بدست می‌آوریم. برای این‌که بتوانیم sub job را انتخاب کنیم، از تابع getSubJob کمک میگیریم. به این صورت که با دادن زمان، تابع از بین sub jobهایی که دارد، sub jobای که بیشترین اولویت برای پیاده‌سازی را دارد، برمیگرداند. سپس این مورد را بررسی میکنیم که jobای مه داریم باید پیاده‌سازی شود یا خیر. سپس job را پیاده‌سازی میکنیم.

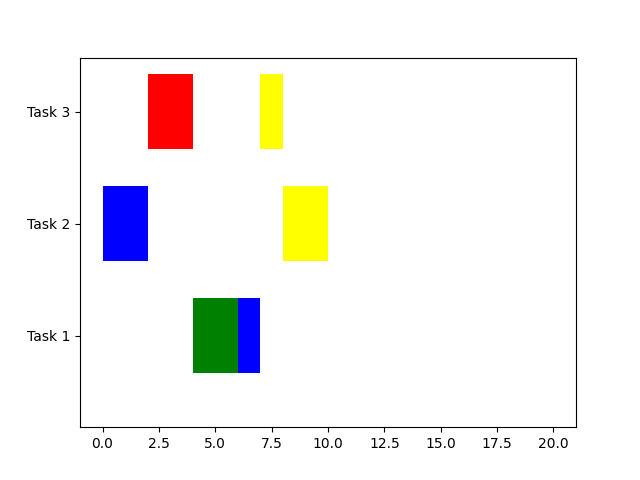
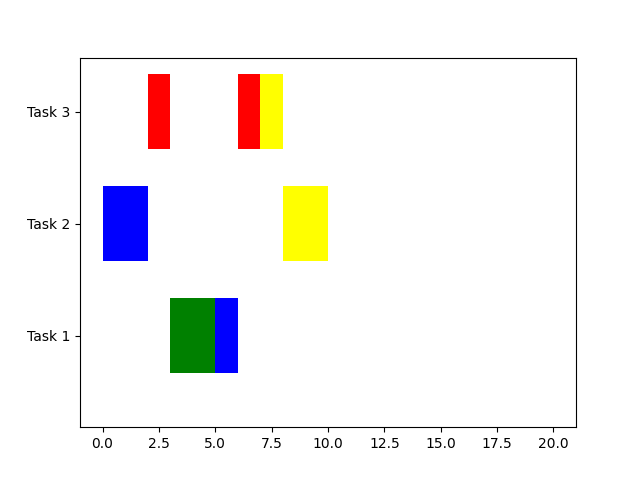
برای پیاده‌سازی از ۴ تسک‌ست استفاده شده است. داریم (اشکال سمت راست مربوط به pip هستند):

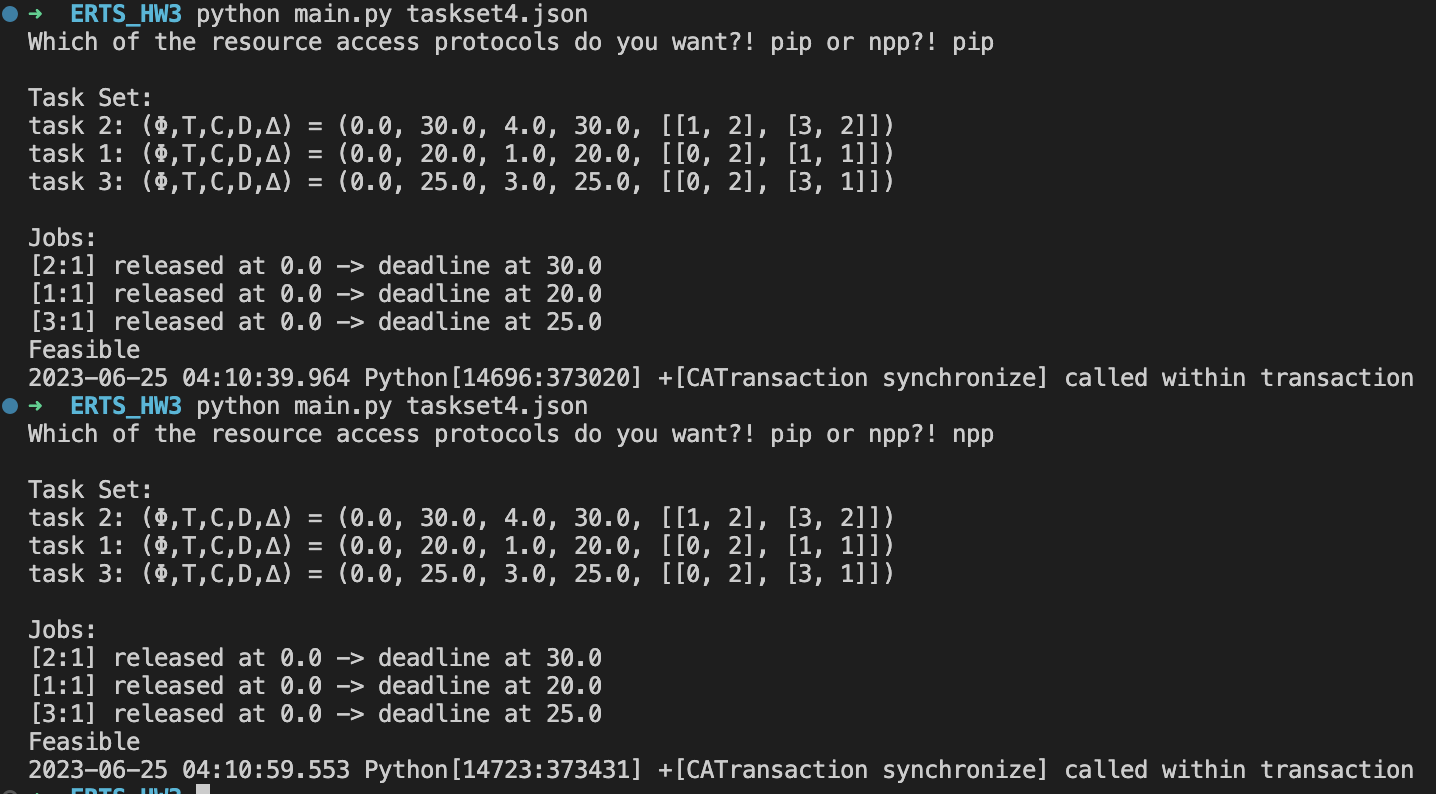


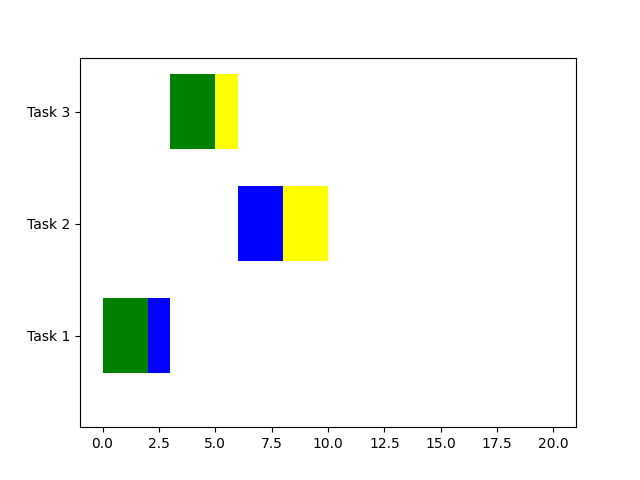
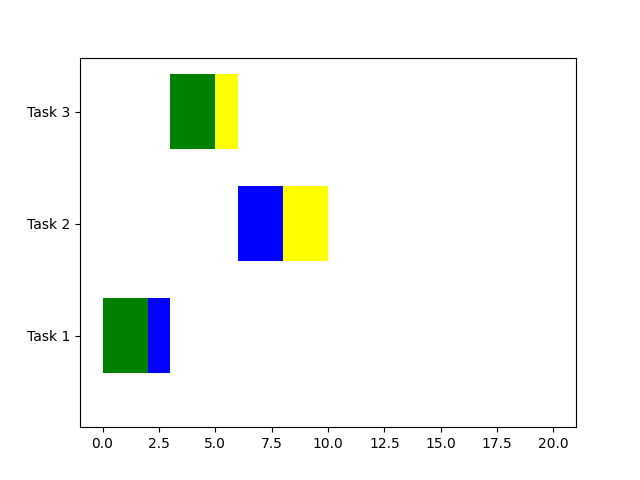


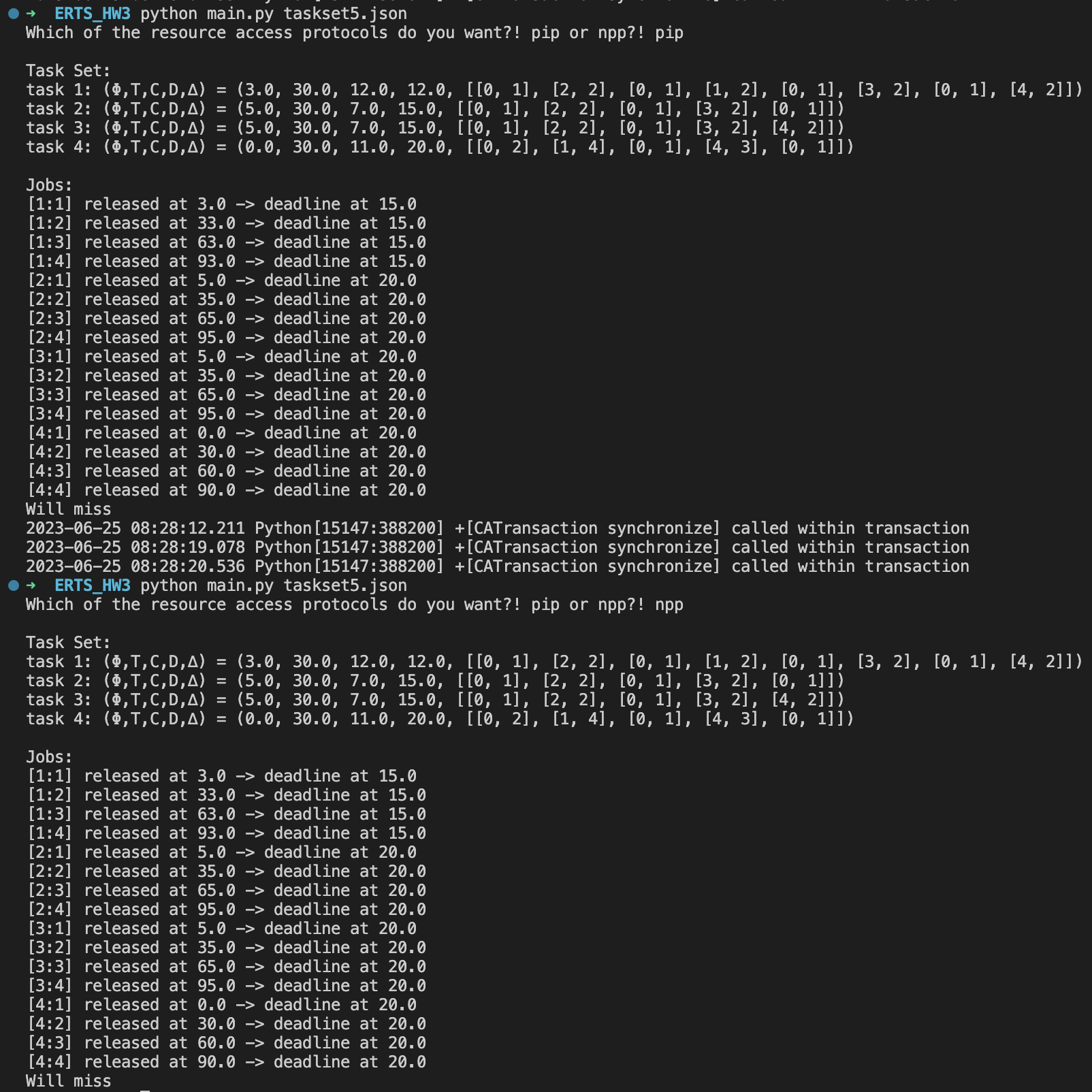


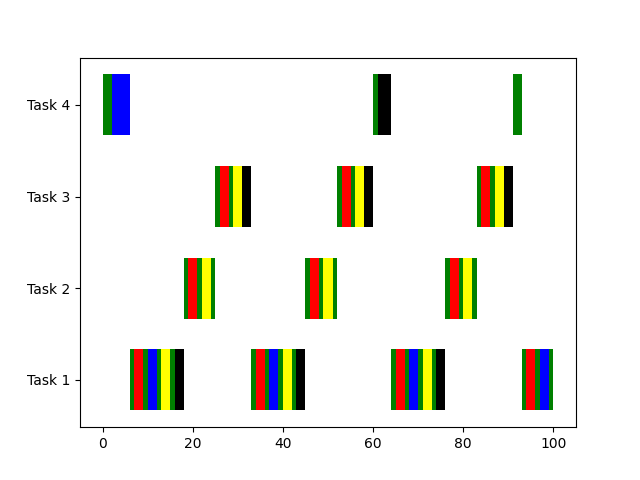
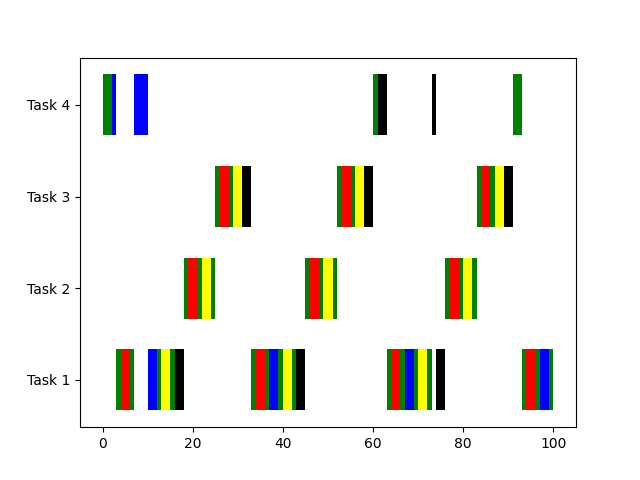












برای تعیین offset میتوان گفت که اگر نداشته باشیم، عملا نیازی که پروتکل همگام‌سازی نداریم (مثال چهارم) و خود DM درحالت عادی پیاده‌سازی میشود.

درمورد تعداد تسک‌ها هم باید گفت که هرچه بالاتر برود، نیاز به پروتکل بیشتر حس میشود زیرا مدیریت آن سخت‌تر میشود. این مورد به این دلیل است که تقاضا برای استفاده از نواحی بحرانی بالاتر میرود.

درمقایسه دو پروتکل میتوان گفت که در NPP قبضه تنها در صورتی رخ میدهد که تسکی با اولویت پایین‌تر از یک ناحیه بحرانی خارج میشود. در این صورت اگر تسکی با اولویت بالاتر وجود داشته باشد، آن را قبضه میکد. همچنین اگر در ناحیه غیربحرانی باشد، منتظر اتمام آن نمانده و درجا قبضه میکند.

اما در PIP وقتی تسکی با اولویت بالاتر وارد میشود، اگر سمافور مورد نیاز آن فرق داشته باشد، درجا قبضه میکند و CPU را در اختیار تسک با اولویت پایین‌تر قرار نمیدهد مگر این که یا تمام شود و یا این که به سمافوری نیاز داشته باشد که تسک با اولویت پایین‌تر در حین استفاده از آن بود که قبضه شد.