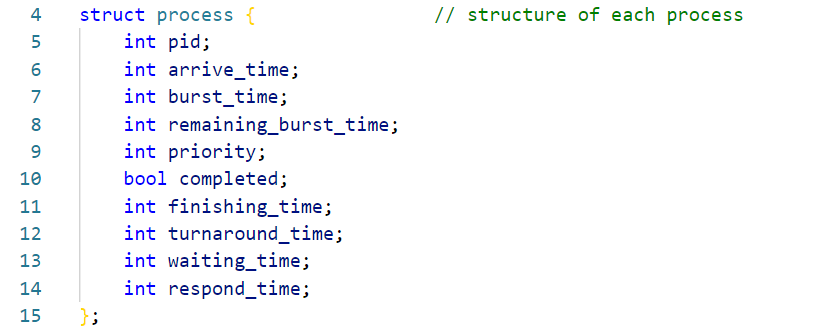
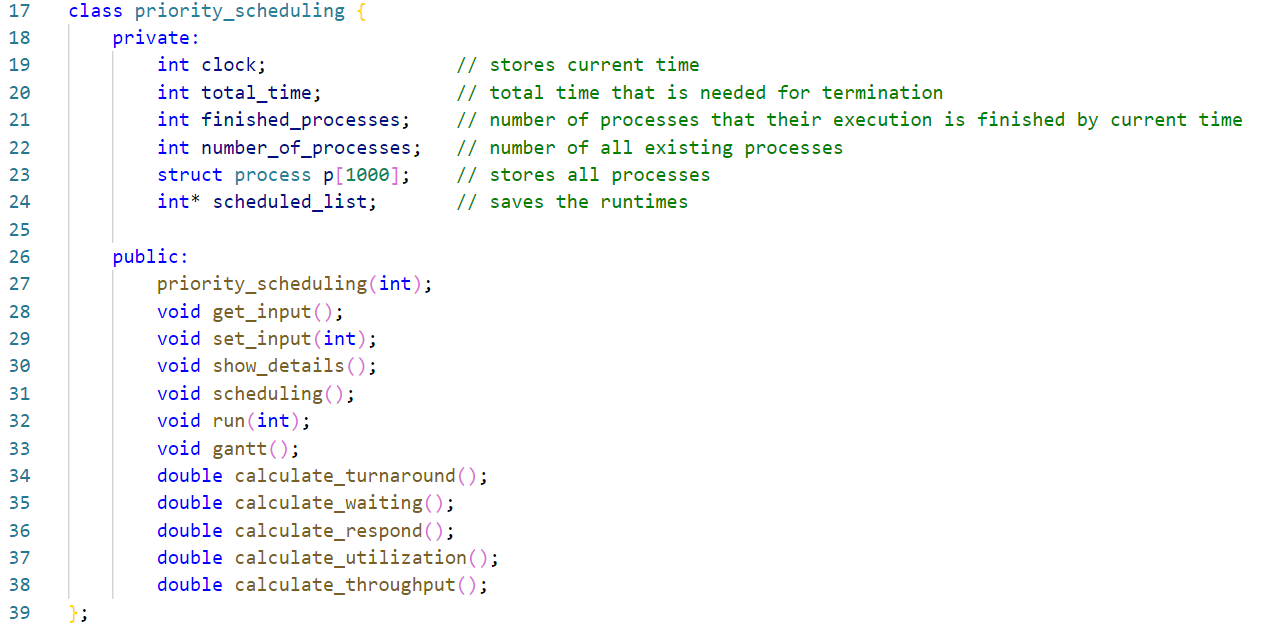
**Preemptive Priority Scheduling**

هنگامی که process های مختلف قرار است روی CPU اجرا شود باید برای اینکه کدام یک زودتر و کدام دیرتر اجرا شود برنامه ریزی انجام شود. برای این کار نیاز به الگوریتمی است که این process ها را ترتیب دهی کند. یکی از این الگوریتم ها الگوریتم Priority Scheduling است. این الگوریتم را میتوان به صورت preemptive یا nonpreemptive اجرا کرد که در این پروژه به صورت preemptive پیاده سازی شده است. Process ها وارد میشوند، یک زمان اجرا دارند و هرکدام اولویتی دارند که این مشخصات به الگوریتم داده میشود. هنگامی که process ای با اولویت بالاتر وارد صف میشود باید فرآیندی که درحال اجراست متوقف شود و فرایند با اولویت بالاتر اجرا شود و پس از اتمام آن فرایند های با اولویت پایین تر به کار خود ادامه دهند.

در اینجا به توضیح کد میپردازیم و همچنین برای یک مثال آن را اجرا میکنیم:



در ابتدا ساختار هر process را مشخص کرده ایم. هر فرایند یک pid دارد که شناسه ای است که با آن این فرآیند توسط CPU و ... شناخته میشود. Arrive\_time زمان ورود فرایند به صف را نشان میدهد. Burst\_time مدت زمانی که اجرای یک cycle فرایند طول میکشد را نمایش میدهد. Remaining\_burst\_time نشاندهنده ی مدت زمان باقیمانده از اجرای آن فرایند است. مثلا اگر فرایند 20 ثانیه برای اجرا نیاز داشته باشد و پس از 5 ثانیه توسط فرایندی دیگر متوقف شود remaining\_burst\_time آن برابر 15 ثانیه است. Priority اولویتی است که به یک فرایند اختصاص دارد. Completed یک متغیر بولین است که صرفا نشان میدهد که آیا اجرای یک فرایند تمام شده است یا هنوز به CPU نیاز دارد. Finishing\_time نشان دهنده ی زمانی است که اجرای یک فرایند بطور کامل خاتمه می یابد و remaining\_burst\_time آن برابر 0 میشود. Turn\_around\_time برابر فاصله ی بین زمان رسیدن و زمان خاتمه ی یک فرایند است. Waiting\_time مجموع کل مدت زمانی است که یک فرایند در انتظار اختصاص CPU است. Respond\_time زمانی است که طول میکشد تا یک فرایند برای اولین بار شروع به اجراشدن کند.



در این قسمت یک کلاس برای انجام عملیات روی فرایند ها تعریف کرده ایم. خصوصیات این کلاس بدین صورت است:

Clock : زمان کنونی سیستم

Total\_time : مجموع کل زمانی که برای خاتمه ی کل فرایند ها نیاز است

Finished\_processes : تعداد فرایند های خاتمه یافته

Number\_of\_processes : تعداد کل فرایند های موجود

P : آرایه ای از همه ی فرایند های موجود

Scheduled\_list : لیست نشاندهنده ی ترتیب اجرا طبق الگوریتم

Priority\_scheduling : تابع initializer که مقادیر اولیه را ست میکند

Get\_input : تابعی که فرایند ها و ویژگی های انها را از ورودی میگیرد

Set\_input : تابعی که ویژگی های فرایند ها را ذخیره میکند و به انها اختصاص میدهد

Show\_details : تابعی که فرایند های موجود و ویژگی های انها از جمله مقدار زمان اجرای باقیمانده شان را نشان میدهد.

Scheduling : تابع اصلی برای انجام عملیات زمانبندی فرایند ها

Run : تابعی که فرایندی که باید اجرا شود را به CPU اعلام میکند و مقادیر و ویژگی های فرایند اجرا شده را اپدیت میکند.

Gantt : تابعی که نمودار gantt فرایند ها را پس از زمانبندی نمایش میدهد.

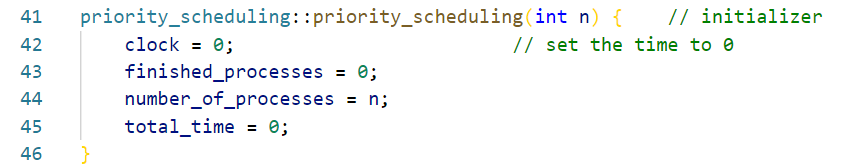
Calculate\_turnarond : متوسط زمان turnaround فرایند ها را محاسبه میکند.

Calculate\_waiting : متوسط زمان waiting فرایند ها را محاسبه میکند.

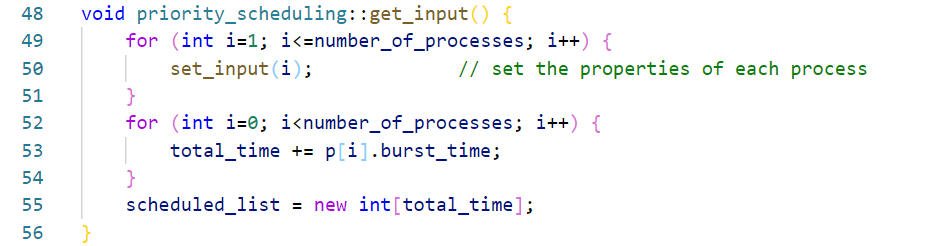
Calculate\_respond : متوسط زمان respond فرایند ها را محاسبه میکند.

Calculate\_utilization : درصد بهره وری CPU را محاسبه میکند.

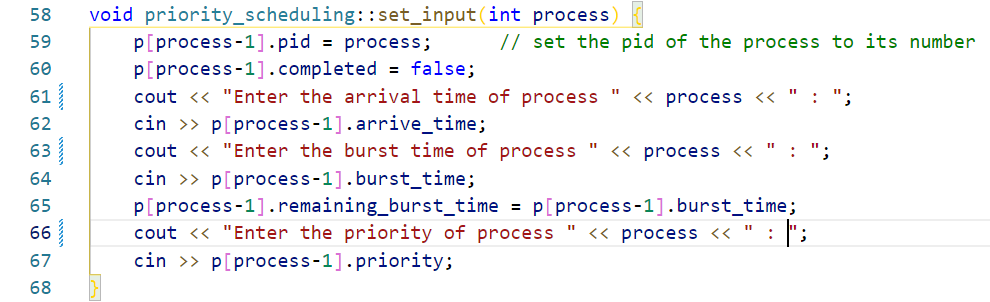
Calculate\_troughput : تعداد فرایندی که در یک واحد زمانی اجرا میشود را محاسبه میکند.



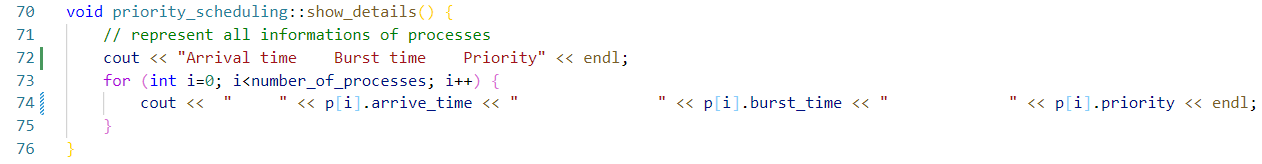
این تابع initializer کلاس است. در این تابع مقادیر اولیه ست میشود. زمان سیستم به 0 ست میشود. تعداد فرایند های خاتمه سافته و زمان کل اجرا هم مقادیر اولیه ی 0 میگیرند و تعداد کل فرایند ها هم از ورودی گرفته میشوند.



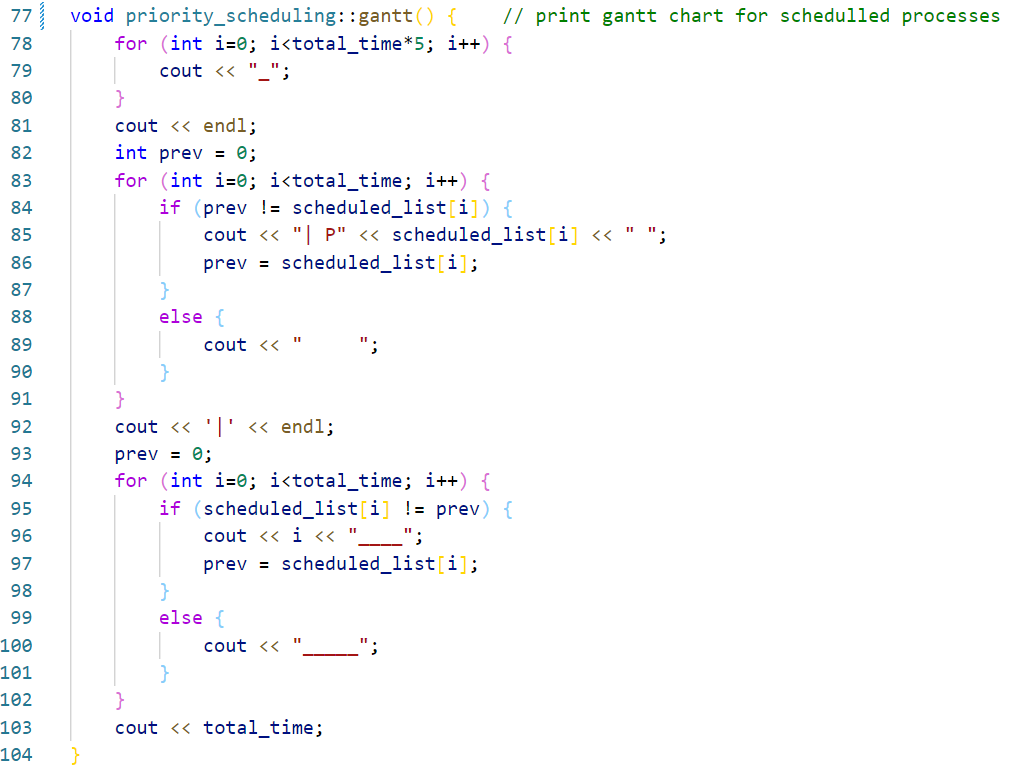
در این تابع برای تعداد کل فرایند های موجود تابع SET فراخوانی میشود. پس از مقدار دهی کردن ویژگی های فرایند ها مقدار کل زمان اجرا محاسبه میشود و ارایه ی نتیجه برای این مقدار زمان ساخته میشود.



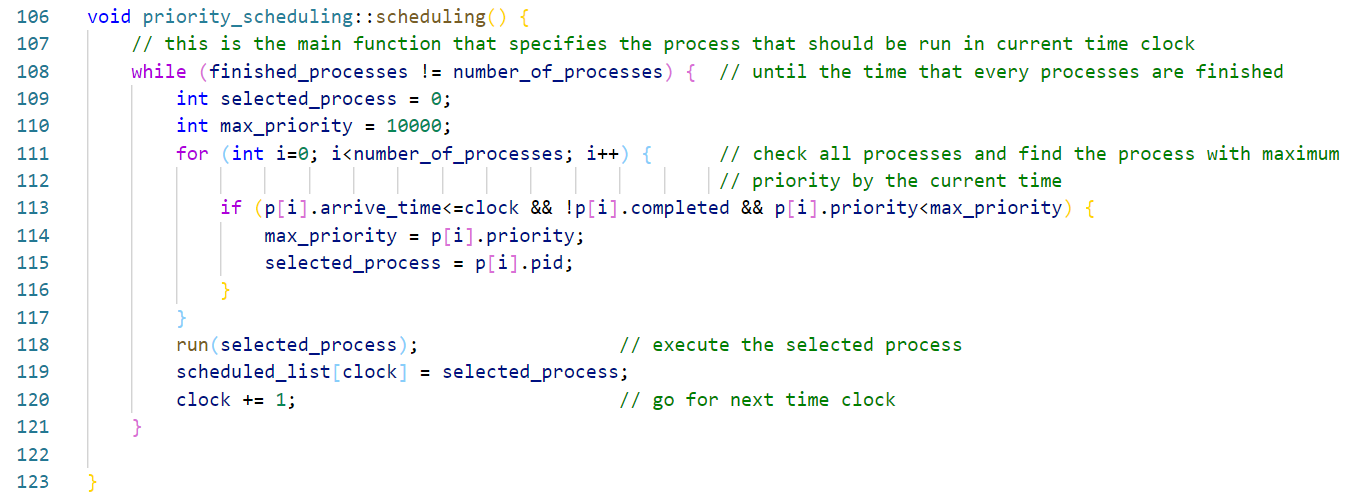
در این تابع مقدار و ویژگی های هر فرایند ذخیره میشوند. در خط 59، pid فرایند به شماره ی ورود ان فرایند ست میشود. سپس ان را خاتمه نیافته مقدار دهی میکنیم. پس از آن مقادیر زمان رسیدن، زمان انفجار و اولویت آن فرایند را از سیستم ورودی میگیریم و هرکدام را در ساختار مربوطه مقدار دهی میکنیم.



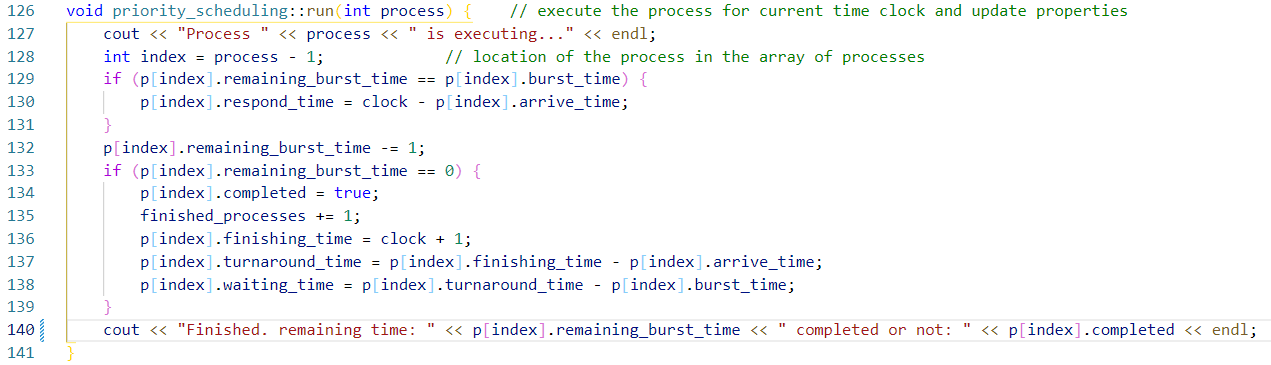
در این تابع برخی مقادیر و ویژگی های فرایند ها را نمایش میدهیم. این تابع برای راحتی کد زدن است و در الگوریتم بکار نمیرود اما کاربر ممکن است بخواهد همه ی اطلاعات را بدست اورد.



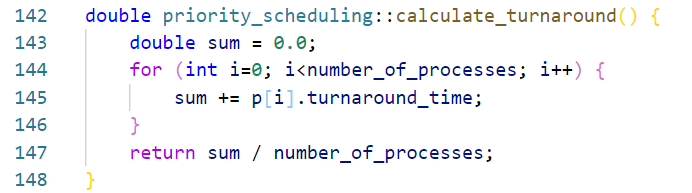
در این تابع نمودار gantt برای فرایند های برنامه ریزی شده ی نهایی رسم میشود. در خطوط 78 تا 80 خط بالای جدول نمایش داده میشود. سپس در خطوط 82 تا 92 فرایند انتخابی به نسبت زمان اجرایی که دارد فضای نمودار را به خود اختصاص میدهد. اگر در دو کلاک متوالی یک فرایند اجرا شود از نوشتن مجدد اسم آن خودداری میشود. سپس در خطوط 93 تا 103 خط پایین نمودار نشان داده میشود که در زمان هایی که فرایند اجرایی تغییر کرده است تایم نمایش داده شده است.



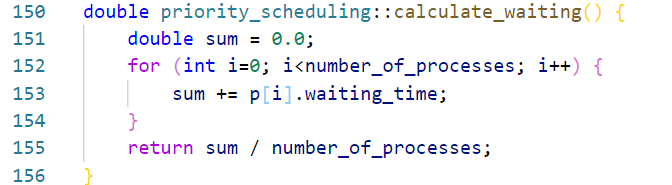
سپس به سراغ تابع اصلی زمانبندی میرویم. این تابع تا زمانی که همه ی فرایند ها خاتمه یابند اجرا میشود تا همه زمانبندی شوند. در حلقه ی for هربار روی همه ی فرایند های موجود حرکت میکنیم. سپس برای هر فرایند چک میکنیم. اگر به زمان رسیدن ان فرایند نرسیده باشیم یعنی هنوز ان فرایند وارد صف انتخاب نشده است پس از ان چشم پوشی کرده و به جست و جو ادامه میدهیم. اگر زمان رسیدن آن فرارسیده باشد اما فرایند پیش از ان خاتمه یافته باشد هم ان فرایند را در نظر نمیگیریم. سپس به سراغ شرط اخر میرویم. تنها درصورتی فرایند جدید جایگزین فرایند قبلی میشود که اولویت آن از قبلی بالاتر باشد. در غیر اینصورت کاری انجام نمیدهد. اما اگر فرایند عوض شود در نهایت فرایند خاتمه نیافته ی موجود با بیشترین اولویت انتخاب میشود و به تابع run داده میشود تا عملیات های لازم را روی آن اجرا کند. سپس برای آن بازه زمانی فرایند انتخاب شده را در لیست نهایی درج میکنیم. و در آخر زمان سیستم 1 واحد افزایش می یابد که به سراغ مرحله ی بعد میرود. از این روش چک کردن در هر واحد زمانی برای اجرای شرط preemptive بودن استفاده میشود. چون بین بازه های زمانی هیچ فرایند جدیدی وارد نمیشود.



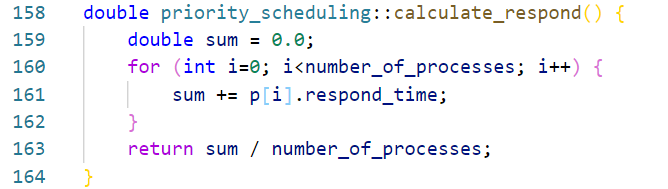
در این قسمت تابع run را توضیح میدهیم. این تابع به صورت واقعی قرار نیست فرایند را اجرا کند چون اجرا کار CPU است و مستقل از این الگوریتم زمانبندی است اما پس از اجرای هر فرایند لازم است عملیاتی روی آن انجام شود که این عملیات ها در این تابع هندل میشوند. ابتدا به کاربر نشان میدهد که کدام فرایند هم اکنون در حال اجراست. سپس ایندکس اصلی فرایند را با توجه به pid آن در صف بدست می آورد. در خطوط 128 تا 130 چک میکند که اگر فرایند برای بار اول است که دارد اجرا میشود زمان پاسخ آن را به زمان حاضر مقدار دهی کند. سپس چون در این واحد زمانی این فرایند اجرا شده است یک واحد از زمان انفجار آن کم میکند. سپس در خطوط 132 تا 138 چک میشود که اگر فرایند پس از این بازه خاتمه یافته است متغیر completed آن را برابر true قرار میدهیم و تعداد فرایند های خاتمه یافته را 1 عدد اضافه میکنیم و زمان پایان آن فرایندرا به کلاک بعدی ست میکنیم. در این قسمت میتوان turnaround time و waiting time نهایی را نیز برای فرایند خاتمه یافته مقدار دهی کرد. سپس به یوزر اعلام میکنیم که این بازه زمان با اجرای فرایند x خاتمه یافت و از آن y مقدار دیگر باقیمانده است و آیا تمام شده است یا خیر.



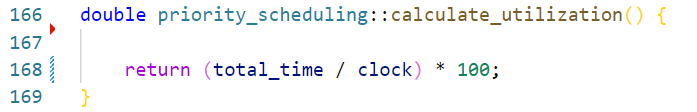
در این تابع برای همه ی فرایند ها مقدار turnaround time ای که هنگام خاتمه فرایند ست شده است را باهم جمع میکنیم و میانگین گیری را انجام میدهیم.



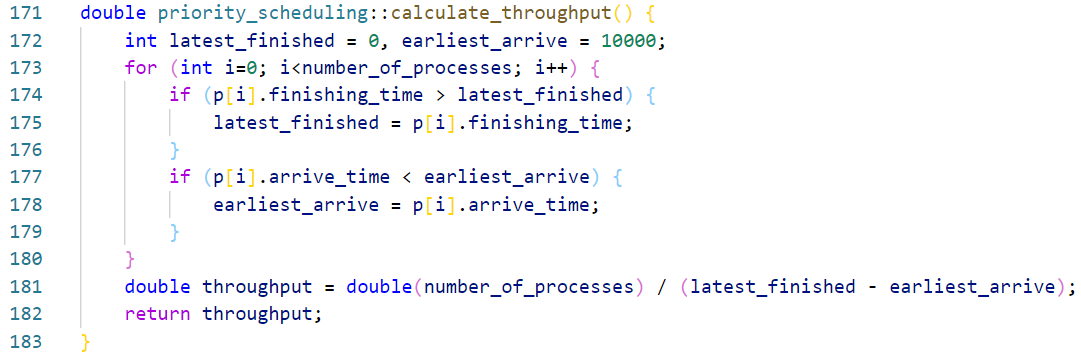
در این تابع برای همه ی فرایند ها مقدار waiting timeای که هنگام خاتمه فرایند ست شده است را باهم جمع میکنیم و میانگین گیری را انجام میدهیم.



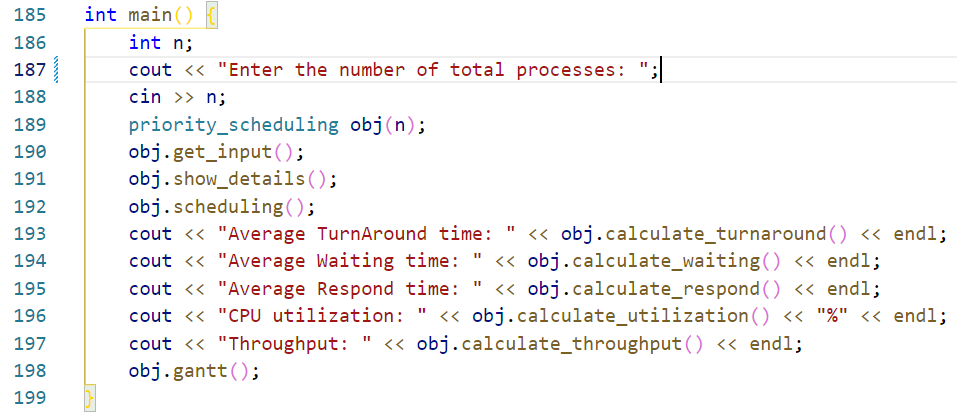
در این تابع برای همه ی فرایند ها مقدار respond timeای که هنگام شروع اجرای فرایند ست شده است را باهم جمع میکنیم و میانگین گیری را انجام میدهیم.



در این تابع بهره وری CPU را محاسبه می کنیم. بهره وری به مقدار زمانی که CPU در حال انجام اجرای فرایند ها بوده در طول زمانی که تا خاتمه ی همه ی فرایند ها است گویند. در متغیر total\_time مجموع زمان انفجار همه ی فرایند ها را ذخیره کرده ایم. سپس تقسیم بر زمان میکنیم و درصد گیری میکنیم.



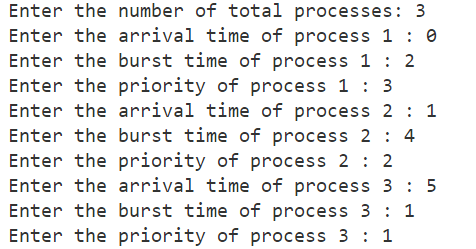
در این تابع throughput را محاسبه میکنیم. باید مشخص کنیم در یک واحد زمانی چند فرایند اجرا شده اند. این موضوع کاملا به فرایند ها و زمان اجرای آنها بستگی دارد. هرچقدر زمان اجرا کوتاه باشد این عدد بزرگتر خواهد شد. برای محاسبه از بین همه ی فرایند های اجرا شده اولین زمان رسیدن و اخرین زمان تمام شدن را پیدا میکنیم. سپس تعداد کل فرایند ها را بر بازه ی رسیدن اولین فرایند تا اتمام اخرین فرایند تقسیم میکنیم.



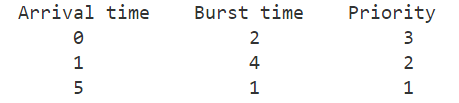
در اینجا تابع main دیده میشود. تعداد کل فرایند ها از ورودی گرفته میشود. سپس آبجکتی از کلاس زمانبندی ساخته میشود که کل عملیات ها روی آن انجام میشود. ساختن کلاس برای این است که میتوان آبجکت های متفاوتی ساخت و زمانبندی را روی صف فرایند های مختلفی به طور همزمان اجرا کرد. سپس فرایند ها را برای آن ابجکت ورودی میگیریم و مقدار دهی میکنیم. پس از آن اطلاعات فرایند های داده شده را نمایش میدهیم و سپس زمانبندی را روی آنها انجام میدهیم. بعد از آن میانگین turn around time, waiting time, respond time و بهره وری CPU و بازده را محاسبه کرده و نمایش میدهیم. در آخر هم نمودار گانت را برای زمانبندی انجام شده نمایش میدهیم.

مثال:

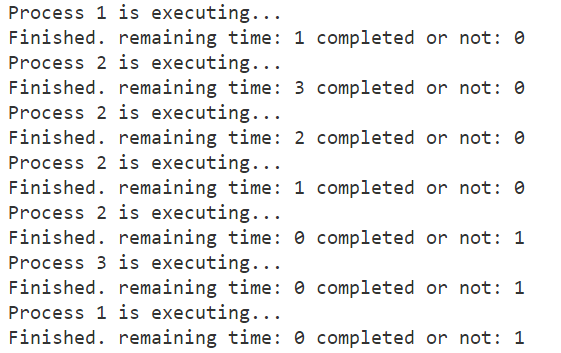
گرفتن ورودی(main):



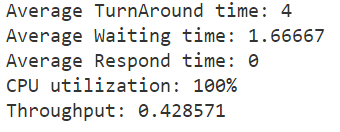
نمایش ویژگی ها(show\_details):



اجرای فرایند انتخابی(run):



نمایش نتایج توابع محاسبات:



نمایش نمودار گانت(gantt):

