

گزارش انجام پروژه

اداره امور مالی ساختمان

اعضای گروه:

میترا فرزام، هستی شاه حسینی، مهدیه رحیمی، عسل حجتی

استاد: استاد علیرضا کدیور

راهنمای پروژه: محمد مهدی حسام

فهرست

۱	مقدمه
۲	راهنمای استفاده از برنامه
۳	ترتیب چاپ خروجی در کنسول
۳	اشاره به کد
۴	بدنه اصلی کد:
۱۱	گزارش تقسیم کار
۱۱	توصیف روند انجام کار
۱۲	چالش های تیم شما در انجام پروژه و قسمت های جذاب کار
۱۴	نمونه خروجی بصری
۱۴	تراز مالی واحدها ها
۱۵	صورت حساب
۱۶	سهم بخشها
۱۶	سهم هزینه های زیرگروه های یک دسته بندی در آن دسته بندی
۱۶	سهم هزینه های دسته بندی نسبت به کل
۱۶	وضعیت موجودی ساختمان
۱۸	روند هزینه ها
۱۸	نمودار تجمعی هزینه های واحدهای مختلف
۱۸	نمودار تجمعی هزینه های مربوط به زیرگروه های مختلف
۱۹	پیش بینی هزینه ی پرداختی هر واحد در سال آینده
۲۰	بیان ایده های خالقانه و کارهای اضافی ای که انجام داده اید

محاسبه دقیق امور ساختمانی می‌تواند کاری بسیار دقیق و وقت‌گیر باشد. با بهره‌گیری از یک برنامه مناسب، می‌توان به راحتی امور مالی ساختمان را بدون نیاز به دانش حساب داری مدیریت کرد. چنین برنامه‌ای می‌تواند علاوه بر صرفه جویی در وقت، میزان دقت محاسبات را به طور چشمگیری افزایش دهد و امکان برنامه ریزی برای کارهای ساختمان را برای مدیر ساختمان بسیار راحت‌تر کند. برنامه ما قادر است رکوردهای حساب داری که توسط کاربر وارد می‌شود ثبت کند و با توجه با آن‌ها در هر بازه زمانی که مد نظر کاربر باشد، انواع گزارش‌ها از قبیل تراز تراکنش مالی واحدهای مختلف، صورت حساب‌های کلی ساختمان، سهم هزینه‌های گروه‌ها و زیرگروه‌های مختلف از کل تراکنش‌ها، روند هزینه‌ها در بازه‌های زمانی مختلف به صورت بصری (شامل نمودارها) و حتی پیشبینی شارژ سال آینده را به کاربر ارائه دهد. این برنامه همچنین قادر است هزینه را به روش‌های مختلفی بین چند واحد از ساختمان پخش کند که در این زمینه کاربر آزادانه تصمیم می‌گیرد که چگونه این روش‌ها را بکار بگیرد. برای مثال کاربر می‌تواند تصمیم بگیرد هزینه را به طور مساوی بین تمام واحدهای مربوطه تقسیم کند و یا بخواهد از انواع دیگر تقسیم بندی از جمله تقسیم بندی برحسب متراژ واحدها، تعداد ساکنین هر واحد، تعداد پارکینگ و مساحت هر واحد استفاده کند. در بحث تقسیم بندی برای موارد خاصی برنامه ما چند تقسیم بندی پیشفرض هم دارد که در قسمت "اشاره به کد" این موارد به طور کامل شرح داده شده است.

راهنمای استفاده از برنامه

در این بخش از گزارش، به دستور کار برنامه در مواردی مانند نحوه صحیح ورودی دادن به برنامه و گرفتن گزارش ها اشاره می شود. در ابتدا برنامه ما برای گرفتن ورودی ها و کار با آن ها از کاربر یک فایل **txt** می خواهد که اطلاعات تراکنش ها به صورت جمله هایی که با **enter** از هم جدا شده اند در آن ها نوشته شده است. اولین کلمه این جمله **append** است که مشخص می کند قرار است تراکنشی در برنامه ثبت شود. سپس تاریخ به صورت میلادی می آید و بعد از آن مبلغ تراکنش. در ادامه به ترتیب دسته، زیر دسته، واحد مسئول (همان مدیر ساختمان است)، واحد یا واحد های سهمیم در این تراکنش، **default**، نوع تقسیم بندی و **'nothing'** قرار می گیرند. برای مثال جمله زیر نمونه ای از ورودی مطلوب است:

```
append 2018/04/01 266 ghabz water 3 4,3,5,10,1,8,9 default water  
'nothing'
```

این مثال نشان می دهد که واحدهای ۴ و ۳ و ۵ و ۱ و ۸ و ۹ باید مبلغ ۲۶۶ هزارتومن را بابت هزینه ی قبض آب به واحد مدیر ساختمان (در این مثال واحد ۳) بپردازند و نوع تقسیم این هزینه بین واحد های دخیل در تراکنش به کمک تابع **water** باشد. همچنین تاریخ ثبت این تراکنش 2018/04/01 است.

البته بعد از اینکه داده های مربوط به یک ساختمان فرضی توسط دستیاران آموزشی در اختیار ما قرار گرفت برای اینکه بتوانیم از آن استفاده کنیم بخشی را به ابتدای کد خود اضافه کردیم که توانایی تبدیل داده هایی که در فایل **data1** بود به داده هایی که مطلوب برنامه ماست را دارا باشد از این رو برنامه ما می تواند با آن فرمت به درستی کار کند. یعنی اکنون برنامه **data1.xlsx** را می خواند (درست شبیه به همان فایلی که به ما داده شد) در ادامه **data2.xlsx** را می خواند و هر خط از **data2** را تبدیل به همان فایل **txt** که از اول برنامه ما با آن کار می کرد تبدیل می کند.

در قسمت های مختلف برنامه همواره توضیحاتی وجود دارد که کاربر را به خوبی راهنمایی می کنند تا بتواند گزارش های مورد نیازش را از برنامه اخذ کند. در هر قسمت از برنامه هم که کار با آن بخش تمام می شود، برای

خارج شدن از آن بخش کافی است کاربر کلمه "خروج" را در کنسول بنویسد. همچنین هنگامی که برنامه از کاربر می‌خواهد یک بازه زمانی دلخواه، چند واحد دلخواه و یا چند دسته و زیردسته دلخواه را انتخاب کند، کاربر می‌تواند با تایپ کردن "-" به برنامه اعلام کند که گزارش مربوط به همه را می‌خواهد و در این بخش نمی‌خواهد فیلتری را اعمال کند. برای مثال منظور از "واحدهای مورد نظر را وارد کنید: -" این است که گزارش ارائه شده به کاربر باید مربوط به تمام واحدها از ۱ تا ۱۰ باشد.

ترتیب چاپ خروجی در کنسول

۱. فایل CSV صورت حساب
۲. سهم بندی
۳. بیان وضعیت ساختمان
۴. تراز مالی
۵. نمودار تجمعی
۶. شارژ واحدها

اشاره به کد

Ourfunctions: در این ماژول تعدادی تابع نوشتیم که در بدنه اصلی کد از آن‌ها استفاده کردیم.

خط ۹ الی ۴۲: در قسمت بخش بندی با استفاده از توابع، توزیع را به گونه‌های مختلف تعریف کردیم. توابع a , r , $parking$ با استفاده از نسبت مساحت و تعداد ساکنین و تعداد پارکینگ توزیع را برای واحدهای مورد نظر انجام می‌دهند. تابع e توزیع به نسبت مساوی میان واحدها را انجام می‌دهد. هزینه قبض آب به نسبت تعداد افراد در هر واحد تقسیم بندی می‌شود. برای توزیع هزینه قبض گاز میان واحدها از توابع a , r (تعداد ساکنین و مساحت واحد) استفاده کردیم به طوری که تاثیر مساحت واحد روی هزینه قبض گاز بیشتر باشد در مورد ایده‌ی این تابع در قسمت "بیان ایده‌های خلاقانه" به طور کامل توضیح داده شده است. هزینه‌ی قبوض

برق و عوارض به یک نسبت بین واحد ها تقسیم شده است. هزینه ی مربوط به آسانسور به گونه ای تقسیم بندی شده است که طبقات بالاتر از سهم بیشتری از مبلغ برخوردار هستند.

خط ۴۶ الی ۵۴: ایجاد یک تابع با آرگومان های آرایه ای از تاریخ ها و دو تاریخ دیگر به عنوان شروع و پایان و برگرداندن آرایه ای از بول، که نشان دهنده بودن یا نبودن هر تاریخ درون بازه داده شده است.

خط ۵۸ الی ۱۴۵: توابع تبدیل تاریخ ها از شمسی به قمری و برعکس با جستجو بدست آمده است.

بدنه اصلی کد:

خط ۱ الی ۵: در این چند خط ماژول های مورد نیاز در برنامه را ایمپورت کرده ایم. ماژول `ourfunctions` را خودمان نوشته ایم و بیشتر توابع مورد نیاز در برنامه را در آن تعریف کرده ایم.

خط ۹: تاریخ فعلی را به صورت `YYYY-MM-DD`، که اولین قسمت رشته `datetime.datetime.now()` است میگیریم و سپس '-' را با '/' جایگزین می کنیم و سپس عنصر اول لیست ایجاد شده که شامل تاریخ می باشد را در متغیری به نام `current_date` ذخیره می کنیم.

خط ۱۱: خواندن `data1` که شامل اطلاعات ساختمان اعم از واحد ها و تعداد پارکینگ و تعداد ساکنین در هر واحد و... است.

خط ۲۸ و ۲۹: خواندن `data2` و در نظر نگرفتن سطرهایی که ستون نام افراد آن ها خالی است و اطلاعاتی در مورد اسامی آن ها داده نشده است.

خط ۳۲ الی ۳۵: تبدیل اسامی به کار رفته در ستون های "دسته" و "زیر دسته" به اسم دلخواه با استفاده از نوشتن دیکشنری.

خط ۳۸ و ۳۹: تغییر قالب تاریخ از `YYYY-MM-DD` به `YYYY / MM / DD` و ذخیره آن در ستون `"date"`، و ساختن ستون جدید برای ذخیره معادل تاریخ های شمسی این تقویم میلادی، به کمک تابعی به نام `togregorian` که از ماژول `ourfunctions` فراخوانی شده است.

خط ۴۲ تا ۵۲: به دلیل استفاده ما از فایل txt و فرمت نوشتاری مشخص برای دادن ورودی به برنامه،

فایل data2 که فرمت xlsx دارد را تبدیل به فایل با فرمت txt با فرمت نوشتاری مشخص کردیم. اینکه ساختار دقیق این جمله چیست پیش تر در بخش "راهنمای استفاده از برنامه" توضیح داده شده است.

خط ۶۰ تا ۶۴: (نحوه ورودی گیری برنامه به صورت ورودی گرفتن در کنسول نبود و با استفاده از فایل

txt ورودی را دریافت می کردیم. با توجه به داده های داده شده در تاریخ 99/11/14 به صورت xlsx مجبور به تبدیل این فایل به فایل txt با فرمت دلخواه شدیم.) فایل txt ایجاد شده را خوانده و اطلاعات آن را خط به خط و با استفاده از فاصله ی بین کلمات موجود در خط داخل لیستی به نام "vorudi" ذخیره می کنیم.

خط ۶۹ تا ۷۴: به منظور تحویل صورتحساب، با استفاده از تابع TRY EXCEPT، برنامه ابتدا سعی

میکند فایلی به اسم bills.csv را بخواند؛ وقتی برنامه برای اولین بار اجرا شده باشد، تبعاً چنین فایلی وجود ندارد و قسمت try با ارور مواجه میشود، در این صورت قسمت except اجرا میشود که در این قسمت bills به صورت یک دیکشنری خالی با کلیدهای معین تعریف میشود و با توجه به اینکه value این کلیدها از نوع لیست تعریف شده اند، برنامه در ادامه ورودی ها را به این لیست های خالی اضافه میکند و در نهایت این دیکشنری به صورت فایل bills.csv ذخیره میشود. در دفعات بعدی که برنامه اجرا میشود، قسمت try با ارور مواجه نمیشود (زیرا bills.csv وجود دارد و برنامه میتواند آن را بخواند)، و اطلاعات جدید به اطلاعات قبلی اضافه میشوند و برنامه در ادامه، با کل داده ها (داده های قدیمی و جدید) کار میکند.

خط ۷۸ تا ۹۲: تبدیل فرمت اعداد به کاربرده شده در ورودی به فرمت مناسب (به طور مثال تبدیل اعداد

از حالت str به int) و همچنین برای استفاده از داده ها نوع آن ها را تغییر داده ایم. برای مثال اگر کاربر بخواهد تمام واحدها را در یک تراکنش دخیل کند، "all" را تایپ کرده و کد آن را تبدیل به لیستی از تمام واحدها می کند و یا اگر چند واحد دلخواه را دخیل نماید، این داده با فرمت str را تبدیل به لیستی از شماره واحدها می کند.

خط ۹۶: ساختن یک کپی از لیست "vorudi"، بنابراین هنگام ایجاد تغییرات در "vorudi"، کپی آن

دست نخورده باقی خواهد ماند.

خط ۹۷ الی ۱۱۷: اگر تعداد واحدهای داده شده به برنامه توسط کاربر بیش از یک واحد باشد، برنامه به

هر یک از واحدها نسبت های مختلفی از مبلغ را با توجه به اینکه این پرداخت متعلق به کدام دسته/زیر دسته است ارائه می دهد و مبلغ مورد نیاز هر واحد را در خط های جداگانه به لیست "vorudi" اضافه می کند و خط اولیه را پاک می کند. یعنی برای مثال اگر در قسمت واحدها ۱،۴،۶ به برنامه داده شده باشد، ابتدا مبلغ کل را که قرار است بین این واحدها تقسیم شود در متغیر payment ذخیره می کند؛ سپس برای هر یک از واحدهای نوشته شده برای مثال واحد ۱، یک خط جدید ایجاد می کند که در آن به کمک دیکشنری تعریف شده و توابعی که از ماژول ourfunctions فراخوانی می شوند، سهم این واحد از کل واحدهای دخیل در تراکنش (در این مثال واحد ۱ و ۴ و ۶) حساب می شود (توجه کنید که این مقدار معمولاً به صورت یک عدد صحیح در نمی آید برای همین سقف آن را در نظر می گیریم زیرا اگر پولی که واحد ها به مدیر می دهند کمی بیشتر از مقدار دقیق سهمشان باشد مدیر می تواند بعداً در قسمت های دیگر از آن استفاده کند ولی اگر چنین نباشد مدیر باید مقدار تفاوت را از جیب خودش بپردازد!). در نهایت تمام خط هایی که در آن ها هزینه مربوط به بیش از یک واحد بود، حذف می شوند و این خط های جدید جای آن ها را می گیرند.

خط ۱۲۰ الی ۱۳۷: به منظور ارائه صورت حساب، اطلاعات موجود در "vorudi" (که لیستی از لیست

ها می باشد) را به طور مناسب در bills (دیکشنری) ذخیره می کنیم و در انتها این دیکشنری را تبدیل به DataFrame می کنیم. قبل از افزودن تاریخ ها به لیست زیر، کلمات "now" با تاریخ جاری جایگزین می شوند، بنابراین همه تاریخ ها از همان الگوی (YYYY / MM / DD) پیروی می کنند. به منظور جلوگیری از چاپ کلمه "undefined" در اطلاعاتی که زیر دسته ندارند (مانند دسته ی نظافت)، در ستون شرح نام دسته چاپ می شود. به منظور مشخص کردن بدهکار و یا بستانکار بودن یک واحد، با توجه به مثبت یا منفی بودن هزینه ی مشخص

شده در "vorudi" ، آن هزینه وارد ستون مربوطه در صورت حساب می شود. سپس اختلاف اندازه‌ی مقدار بدهکاری و بستانکاری ستون "mande" را مشخص می‌کند. در آخر bills را تبدیل به دیتا فریم می‌کنیم.

خط ۱۳۹ و ۱۴۰: به منظور تسهیل مرتب سازی صورتحساب بر حسب زمان ستونی به نام

"timestamps" به صورت حساب اضافه می‌نماییم. (در حقیقت تایپ تاریخ ها از حالت str به Timestamp تغییر یافته است). سپس این صورت حساب را بر حسب واحد، تاریخ و دسته‌ی آن مرتب می‌نماییم.

خط ۱۴۲ تا ۱۵۲: در این قسمت ابتدا برنامه یک کپی از bills به نام output می‌گیرد تا در ادامه

تغییراتی که ایجاد می‌شوند باعث به هم ریختن برنامه نشون. سپس برنامه از کاربر می‌خواهد بازه زمانی مد نظرش را وارد کند. در این حالت کاربر دو تاریخ را وارد می‌کند که ابتدا با کمک تابع "togregorian" از ماژول ourfunctions، تبدیل به میلادی می‌شوند. (اگر کاربر "now" را وارد کند با تاریخ جاری جایگزین می‌شود). سپس این دو تاریخ توسط دستور split از هم جدا می‌شوند و عضول اول با نام start و عضو دوم با نام end ذخیره می‌شود. سپس به کمک تابع tarikh_filtered از ماژول ourfunctions تمامی تراکنش های بین دو سر بازه‌ی شروع و پایان در tarikh_lists وارد می‌شوند بعد تبدیل به آرایه های نامپای شده و output به وسیله آن‌ها فیلتر می‌شود. در این صورت در output فقط اطلاعات مربوط به تراکنش های بین در تاریخ مد نظر کاربر قرار می‌گیرد.

خط ۱۵۴ تا ۱۵۸: صورت حساب نهایی متشکل از چند ستون مشخص با نام های مشخص می‌باشد که

از "output" بدست می‌آید. همچنین تاریخ های میلادی به تاریخ های شمسی تبدیل می‌شوند؛ و در نهایت این صورت حساب به صورت فایل xlsx ذخیره می‌شود.

خط ۱۶۴ تا ۱۹۷: برنامه قادر است سهم هزینه‌های هر کدام از زیر گروه‌ها نسبت به کل هزینه های آن

دسته را در قالب یک DataFrame چاپ کند. یعنی مجموع هزینه‌ی هر کدام از زیر دسته‌ها که فقط مربوط به دسته‌ی قبض است را نسبت به هزینه‌ی کل قبوض چاپ میکند. برنامه قادر است سهم هزینه‌های هر کدام از

دسته‌ها را نسبت به کل هزینه‌ها را در قالب یک DataFrame چاپ کند. یعنی مجموع هزینه‌ی هر کدام از دسته‌ها را نسبت به کل هزینه‌ها چاپ می‌کند.

خط ۲۰۰ الی ۲۱۳: دیتا فریم bills_copy از روی همان bills می‌سازیم و آن را با تاریخ‌ها مرتب می‌کنیم و در ستون جدید jalali Y/M سال و ماه هر تراکنش را قرار می‌دهیم. بعد دو ستون جدید به bills_copy اضافه می‌کنیم. یکی ماه تراکنش و دیگری سال آن را برای هر واحد نشان می‌دهد. bills_mohem را به گونه‌ای تعریف می‌کنیم که تمام هزینه‌های مربوط به دو ماه آخر را به ما بدهد. همچنین vase_koli را تعریف می‌کنیم تا نشان دهد کل تراز مالی ساختمان چگونه است. یعنی به طور کلی اعضای ساختمان چقدر بدهکار و یا بستانکارند.

خط ۲۱۶ تا ۲۳۲: این قسمت مدت زمان بین اولین تراکنش ثبت شده تا آخرین تراکنش را به ما نشان می‌دهد و در آخر به کمک format جمله‌ای را در کنسول می‌دهد که به کاربر اعلام کند وضعیت ساختمان را در چه بازه‌ای از زمان بررسی کرده است.

خط ۲۳۳ تا ۲۵۹: در اینجا برنامه جمع هزینه‌های کل این مدت را با جمع قبض‌های دو ماه آخر مقایسه می‌کند و یک بازه‌ی احتمال ۲۰ درصدی برای خرج‌های غیرمنتظره تعریف میکند و می‌گوید اگر موجودی ساختمان از ۸۰ درصد مجموع هزینه قبض دو ماه اخیر کمتر باشد، وضع ساختمان مطلوب است و در غیر این صورت یا اضطراری است و یا نسبتاً نامطلوب. علت این کار در بخش "ایده‌های خلاقانه" توضیح داده شده است. برنامه همچنین می‌تواند واحد یا واحدهایی که بیشترین بدهی را دارند مشخص کند تا مدیر بداند مشکل از کجاست. در اعلام یک یا دو واحدی که بیشترین بدهی را دارند، اگر بدهی همه واحدها یکی بود عبارت(همه‌ی واحدها به یک اندازه بدهکارند) چاپ می‌شود و در سایر مواقع اگر دومین واحد با بیشترین بدهی حداقل ۸۰ درصد بدهی واحد اول را داشته باشد در کنار واحد اول گزارش می‌شود و در غیر این صورت فقط واحد اول گزارش می‌شود.

خط ۲۶۴ تا ۲۸۸: در این قسمت برنامه قابلیت فیلتر کردن بر اساس تاریخ و واحد های مد نظر کاربر را

دارد. به طوری که ابتدا از کاربر می خواهد بازه زمانی، واحدها، دسته و زیرسته های دلخواهش را وارد کند. توجه کنید با وارد کردن '-' برنامه با تمام داده های در دسترس بررسی های خود را انجام می دهد. سپس به کمک تابع `tarikh_filter` تراکنش های مربوط به بازه وارد شده را جدا می کند و در دیتا فریم `bills_filtered` همه اطلاعات تراکنش هایی که کاربر تاریخ و واحد های مربوطه اش را انتخاب کرده ذخیره می شوند.

خط ۲۹۲ الی ۳۲۳: با توجه به اطلاعاتی که کاربر در قسمت قبلی وارد می کند، می خواهیم در این بخش

از کد، فیلترهایی را اعمال کنیم تا برنامه محاسبه تراز مالی به تفکیک واحدها و صورت حساب ها در یک بازه معین را به کاربر ارائه دهد. برای اینکه برنامه تراز مالی واحدهایی که مدنظر است در بازه زمانی مشخص خروجی دهد، کافی است کاربر عبارت "واحدها در کل" را وارد کند. در این صورت برنامه تراز مالی تک تک واحدها را در کنسول چاپ می کند. همچنین با وارد کردن عبارت "واحد به تفکیک شرح" برنامه با تعدادی جمله برای کاربر شرح می دهد که هر واحد بابت چه هزینه ای چه مقدار بدهکار است. یعنی این بار علاوه بر میزان بدهی هر واحد شرح آن را نیز به کاربر ارائه می دهد. در صورتی که کاربر عبارت "شرح به تفکیک واحد" را وارد کند برنامه در کنسول وضعیت تراز مالی همه واحدها را در رابطه با عوارض شهری، قبض برق، قبض گاز و سایر موارد نمایش می دهد.

خط ۳۲۹ الی ۴۴۸: وجود یک نمودار تجمعی که بتوان فیلترهایی را روی آن اعمال کرد می تواند کمک

زیادی در بصری سازی روند هزینه ها بکند. این برنامه مدیر ساختمان را قادر می سازد تا هزینه های مربوط به چند زیرگروه خاص یا چند دسته بندی خاص یا کل هزینه ها را برای چند واحد خاص در یک بازه خاص به صورت نمودار تجمعی ببیند. در ابتدا کاربر با وارد کردن "واحدها" به برنامه اطلاع می دهد که قصد دیدن نمودار تجمعی واحدها را دارد. در ادامه کاربر باید بازه زمانی، شماره واحد، دسته و زیر دسته مد نظرش را در کنسول وارد کند. سپس برنامه برای هر واحد به صورت جداگانه، دو نمودار در یک صفحه به کاربر ارائه می دهد. محور افقی این دو نمودار تاریخ است. در نمودار میله ای روی محور عمودی تمامی تراکنش ها نمایش داده می شوند و محور عمودی

نمودار اسکتر نشان دهنده جمع تجمعی است. در ادامه برنامه این امکان را به کاربر می دهد که اگر چند تراکنش در یک تاریخ اتفاق افتاده باشند، کاربر بتواند جدولی جداگانه مشاهده کند که این جدول جزئیات بیشتری در مورد تراکنش هایی که در یک روز اتفاق می افتند دارد. برای پیدا کردن چنین تراکنش هایی از `deduplicated` استفاده می شود که هر جا تاریخ ها یکسان بود گزاره ای منطقی با ارزش درست یا نادرست می دهد و این گزاره ها می توان در یک `series` به نام `is_duplicated` ذخیره کرد.

همچنین اگر کاربر در ابتدای این بخش عبارت "زیردسته ها" را وارد کند، در ادامه با وارد کردن زیردسته و تاریخ مد نظر نمودار مشابه بالا دریافت می کند با این تفاوت که این بار فیلترها بر اساس زیردسته و زمان هستند. همچنین کاربر می تواند با وارد کردن عبارت "همه" به جای نام چند زیرگروه، نمودار مربوط به تمام زیرگروه ها را جدا جدا دریافت کند.

خط ۴۵۲ الی ۴۷۵: این قسمت از کد به پیش بینی هزینه ی پرداختی هر واحد در سال آینده اختصاص دارد. یعنی برنامه می تواند با استفاده از رکوردهایی که در یک بازه که طول آن حداقل یک سال است ثبت شده اند، برآوردی از شارژ ثابت واحدها در آینده ارائه دهد. برای این کار، برنامه باید در ابتدا تشخیص دهد داده های ورودی مربوط به چند ماه است. برای این کار ابتدا یک دیتافریم جدید به نام `bills_est` تعریف می شود که در آن تراکنش ها طبق ترتیب رخ دادن آن ها مرتب می شوند؛ سپس دو ستون جدید به این دیتافریم اضافه می شود که یکی از آن ها `jalali Y/M` نام دارد و در آن سال و ماه هر تراکنش قرار دارد و دیگری `jalali y` است که فقط سال هر تراکنش در آن قرار داده شده است. برای یافتن تعداد ماه هایی که در آن ها تراکنش انجام شده است، کفایت از `unique` استفاده کنیم و تعداد داده های غیر تکراری موجود در `jalali Y/M` را بدست آوریم. سپس برای هر واحد ابتدا کل هزینه ها را جمع می شوند و در دیتا فریم `allyears_pay` ذخیره می شوند و بعد یک ستون به نام `sharj` به `allyears_pay` اضافه می شود که در آن کل میزان بدهکاری هر واحد به تعداد ماه هایی که این تراکنش ها در آن صورت گرفته تقسیم می شوند (به عبارتی میانگین هزینه برای هر واحد در هر ماه را بدست می آوریم). همچنین به اندازه ی ۱۰ درصد آن مقدار برای هزینه های غیرمنتظره به آن اضافه کردیم و این

مقدار را برای تمامی واحدها در کنسول چاپ کردیم. (توجه کنید در مواردی که عدد حاصل صحیح نباشد مقدار سقف آن را در نظر می گیریم.) این مقادیر به دست آمده می توانند تا حدودی نزدیک به شارژ میانگین این واحدها در سال آینده باشند.

گزارش تقسیم کار

در انجام این پروژه تک تک اعضای تیم سعی کردند تعامل خوبی با هم داشته باشند و کارها تا حد امکان به صورت تیمی انجام شود. از این رو اعضای تیم همگی در جلسات یکی دو ساعته که از تاریخ ۹۹/۱۰/۱۱ شروع شد و اکثر شبها حتی در طول امتحانات ترم برگزار می شد حضور می یافتند و در مورد جنبه های مختلف پروژه بحث می کردند و از این رو اکثر کدهای این برنامه با همفکری تمام اعضا و با به اشتراک گذاشتن اسکرین در محیط اسکای روم زده شد. با وجود فشردگی این ترم و سنگین بودن امتحانات ما از هیچ فرصتی برای کار روی این پروژه دریغ نکردیم و اکثر کارها با زمان بندی خوب و برنامه ریزی به درستی پیش رفتند و طی ۱۵ جلسه پایان یافت.

توصیف روند انجام کار

روند انجام کارهای پروژه همان طور که در بخش "گزارش تقسیم کار" بیان شد به صورت گروهی بود. برای نوشتن کدهای این برنامه در محیط اسکای روم یکی از اعضای گروه صفحه اش را به اشتراک می گذاشت و با فکری هم، کد بخش های مختلف را می زدیم. این روند کمک می کرد که نتایج به دست آمده همان چیزی باشند که همه اعضا روی آن توافق نظر داشته اند و هیچ کس از بخش های دیگر کد بی خبر نماند. ما معتقدیم این روش مزایایی را به همراه داشت از جمله یادگیری عمیقی که در روند مشارکت و بحث در مورد بخش های مختلف کد صورت گرفت. با توجه به مجازی بودن این ترم درس ها تا حد زیادی به صورت انفرادی در آمده بودند اما در طول انجام این پروژه همفکری ها و مشورت ها باعث شد وارد مرحله جدیدی از درک درس مبانی برنامه نویسی شویم. این پروژه برای یادگیری این درس بسیار مفید بود.

چالش های تیم شما در انجام پروژه و قسمت های جذاب کار

به طور کلی می توان گفت اصلی ترین چالش ما درک موارد خواسته شده در شیوه نامه بود. از آن جایی که ماهیت این پروژه با تمرین ها متفاوت بود و در بسیاری از موارد به جای وجود دستورالعمل مشخص باید از ایده های جدید استفاده می کردیم، در برخی از قسمت ها کمی دچار سردرگمی می شدیم. در مواجهه با چالش ها اول تلاش می کردیم به صورت گروهی با هم بحث کنیم و راه حل مناسبی را پیدا کنیم، در صورتی که نیاز بود حتما از منابع اینترنت کمک می گرفتیم و سعی می کردیم با جست وجوهای مناسب پاسخ سوالاتمان را بیابیم. اگر هم در مورد خواسته های طراحان پروژه ابهامی وجود داشت، از راهنمای پروژه سوال می کردیم که بسیار عالی ما را راهنمایی می کرد.

بجز موارد بالا گاهی با چالش های کوچکتر هم رو به رو می شدیم. یکی از این چالش ها مربوط به زمانی بود که برای امتحان کردن کد مجبور بودیم مدام آن را از اول ران کنیم و برای این کار مجبور بودیم هر بار از اول تعداد زیادی ورودی در کنسول وارد کنیم که زمان زیادی می برد. راه حل خلاقانه یکی از اعضای گروه (میترا فرازم) باعث حل این مشکل شد و البته منجر به تغییر نحوه کلی ورودی گرفتن برنامه شد. ما به این نتیجه رسیدیم که کاربر می تواند بدون نیازی به نوشتن تمام ورودی ها در کنسول به راحتی از یک فایل `txt` استفاده کند و در قالب چند جمله با ساختاری ساده و مشخص، تراکنش های مالی ساختمان را ثبت کند و برنامه ما با خواندن همین فایل ورودی های مورد نظر را دریافت می کند. که البته بعد از اینکه دستیاران آموزشی فایل های مربوط به داده های یک ساختمان را به ما تحویل دادند، به ناچار برای اینکه بتوانیم از آن داده ها استفاده کنیم، بخشی را به کد اضافه کردیم که بتواند این فایل را به متن `txt` که برنامه ما از قبل با آن کار می کرد تبدیل کند. قطعاً اگر این داده ها را زودتر دریافت می کردیم در بخش ورودی گرفتن دچار این چالش نمی شدیم.

چالش دیگری که داشتیم با نمودار تجمعی بود که ترسیم آن نزدیک به چند روز زمان برد. مفهوم نمودار تجمعی برای ما تازگی داشت و ابتدا مقداری در مورد آن تحقیق کردیم و البته در این زمینه از راهنمای پروژه هم

سوالاتی پرسیدیم. چالش اصلی ما تبدیل داده هایی که در یک جدول تجمعی به دست آورده بودیم به نمودار مناسب بود. پیدا کردن نموداری که بتواند روند تراز مالی ساختمان را به درستی نشان دهد تا حدی چالش برانگیز اما جالب بود.

یکی دیگر از چالش های اصلی ما این بود که داده های ساختمان فرضی که در اختیار ما قرار گرفت در بعضی از قسمت ها با برنامه ما تطابق نداشت. برای مثال در برنامه ما به طور مشخص هنگامی که واحد ها باید هزینه ای را پرداخت می کردند آن ها را بدهکار می کرد و بعد هنگامی که این هزینه را می پرداختند آن را از بدهی کم می کرد. در نتیجه برنامه با توجه به روندی که برای آن طراحی کرده بودیم می توانست وضعیت ساختمان را گزارش کند. اما در داده ای که به ما داده شد هرگز گفته نشده که واحدها بدهی خود را پرداخت می کنند بنابراین واحد ها فقط بدهکارتر می شوند و وضعیت ساختمان مدام نامطلوب تر می گردد. از این رو با توجه به تنظیمات مربوط به گزارش وضعیت ساختمان، وضعیت همواره اضطراری است. که البته اگر داده ها درست به برنامه داده شوند برنامه هم به درستی کار می کند. مشکل دوم ما با این داده ها، که مشکلات زیادی را در بخش پیشبینی هزینه های سال آینده برای ما به وجود آورد، این بود که در برخی از ماه ها بعضی از واحد ها اصلاً قبض آب یا برق نداشتند. همه ما می دانیم که در دنیای واقعی امکان ندارد که در یک ماه هزینه برق مشاع فقط مربوط به چند واحد خاص باشد.

می توان گفت مراحل که در این پروژه طی کردیم هم ما را به چالش می کشید و هم بسیار جالب بود. به کارگیری کد ها فراتر از یک محیط خیالی و استفاده از آن ها برای حل مسائل روزمره که در دنیای اطرافمان با آن روبه رو هستیم تجربه جالبی بود که در تمرین ها با آن مواجه نمی شدیم. این پروژه علاوه بر مهارت های برنامه نویسی، مهارت حل مسئله و مهارت های مشارکت تیمی ما را به چالش کشید.

نمونه خروجی بصری

ما با استفاده از صورت حساب های یک ساختمان فرضی به ما داده شد درستی برنامه خود را مورد آزمایش قرار دادیم. در این بخش گزارش خروجی های برنامه در ارتباط با این داده به صورت زیر است:

تراز مالی واحدها ها

در این بخش ما میزان بدهکاری و یا بستانکاری واحدهایی که کاربر می خواهد در بازه ی زمانی دلخواه به مدیر ساختمان تحویل میدهیم. ولی ما در این بخش از یک ایده ی خلاقانه و جالب استفاده کردیم که روند مدیریت را برای مدیر ساختمان راحت تر می کند؛ در این بخش ما سه نوع خروجی دلخواه به کاربر می دهیم. کاربر باید انتخاب کند که آیا تراز مالی کلی واحدها را می خواهد یا تراز مالی واحدها را به تفکیک دسته و زیر دسته (شرح) و یا تراز مالی واحدها به ترتیب شرح تراکنش ها. نمونه:

```
بازه زمانی: now 1397/01/12
واحدهای خواسته شده: 1
در صورت تمایل به مشاهده تراز مالی کلی عبارت 'واحدها در کل' را وارد کنید.
در صورت تمایل به مشاهده تراز مالی واحدها به تفکیک شرح عبارت 'واحد به تفکیک شرح' را وارد کنید.
و در صورت تمایل به مشاهده تراز مالی شرح به تفکیک واحدها عبارت 'شرح به تفکیک واحد' را وارد کنید.
در غیر این صورت 'خروج' را وارد کنید.
واحدها در کل
واحد 1. 4322 تومان بدهکار است.
```

۱. واحدها در کل

```
بازه زمانی: -
واحدهای خواسته شده: 3
در صورت تمایل به مشاهده تراز مالی کلی عبارت 'واحدها در کل' را وارد کنید.
در صورت تمایل به مشاهده تراز مالی واحدها به تفکیک شرح عبارت 'واحد به تفکیک شرح' را وارد کنید.
و در صورت تمایل به مشاهده تراز مالی شرح به تفکیک واحدها عبارت 'شرح به تفکیک واحد' را وارد کنید.
در غیر این صورت 'خروج' را وارد کنید.
واحد به تفکیک شرح
واحد 3. 258 تومان بابت هزینه ی avarex بدهکار است.
واحد 3. 281 تومان بابت هزینه ی electricity بدهکار است.
واحد 3. 776 تومان بابت هزینه ی elevator بدهکار است.
واحد 3. 432 تومان بابت هزینه ی gas بدهکار است.
```

۲. واحد به تفکیک شرح

توضیح: تفکیک واحد

1	واحد	584	تومان	بابت	هزینه	در	avarez	بدفکار	است.
2	واحد	645	تومان	بابت	هزینه	در	avarez	بدفکار	است.
3	واحد	258	تومان	بابت	هزینه	در	avarez	بدفکار	است.
4	واحد	971	تومان	بابت	هزینه	در	avarez	بدفکار	است.
5	واحد	1241	تومان	بابت	هزینه	در	avarez	بدفکار	است.
6	واحد	1241	تومان	بابت	هزینه	در	avarez	بدفکار	است.
7	واحد	971	تومان	بابت	هزینه	در	avarez	بدفکار	است.
8	واحد	1241	تومان	بابت	هزینه	در	avarez	بدفکار	است.
9	واحد	971	تومان	بابت	هزینه	در	avarez	بدفکار	است.
10	واحد	1241	تومان	بابت	هزینه	در	avarez	بدفکار	است.
11	واحد	224	تومان	بابت	هزینه	در	electricity	بدفکار	است.
12	واحد	269	تومان	بابت	هزینه	در	electricity	بدفکار	است.
13	واحد	201	تومان	بابت	هزینه	در	electricity	بدفکار	است.
14	واحد	241	تومان	بابت	هزینه	در	electricity	بدفکار	است.
15	واحد	186	تومان	بابت	هزینه	در	electricity	بدفکار	است.
16	واحد	166	تومان	بابت	هزینه	در	electricity	بدفکار	است.
17	واحد	189	تومان	بابت	هزینه	در	electricity	بدفکار	است.
18	واحد	178	تومان	بابت	هزینه	در	electricity	بدفکار	است.
19	واحد	309	تومان	بابت	هزینه	در	electricity	بدفکار	است.
20	واحد	275	تومان	بابت	هزینه	در	electricity	بدفکار	است.
1	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.
2	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.
3	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.
4	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.
5	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.
6	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.
7	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.
8	واحد	776	تومان	بابت	هزینه	در	elevator	بدفکار	است.

۳. شرح به تفکیک واحد

*توجه کنید با وارد کردن '-' برنامه با تمام داده های در دسترس بررسی های خود را انجام می دهد.

صورت حساب

در این بخش برنامه بازه ی زمانی دلخواه را از کاربر دریافت می کند، سپس صورت حساب را در قالب یک

فایل csv با نام output به کاربر تحویل می دهد.

```
In [3]: runfile('C:/Users/hs138/
Downloads/project0 (1).py', wdir='C:/
Users/hs138/Downloads')
Out[3]: 
برای تحویل فایل صورت حساب بازه زمانی
دلخواه را وارد کنید
بازه زمانی: 1398/01/12
Out[3]:
```

	A	B	C	D	E	F	G	H	I	J	K	L
1	vahed	date	date	zindaste	mablagh	sahm						
2		1 1398/01/2	neozafat	undefined	281	36						
3		1 1398/01/2	neozafat	undefined	293	30						
4		1 1398/02/1	ghabz	water	212	13						
5		1 1398/02/1	tamirat	undefined	253	26						
6		1 1398/02/1	ghabz	gas	487	39						
7		1 1398/02/1	neozafat	undefined	310	39						
8		1 1398/02/2	ghabz	electricity	110	14						
9		1 1398/02/2	elevator	undefined	309	31						
10		1 1398/02/3	parking	undefined	408	41						
11		1 1398/03/0	neozafat	undefined	284	48						
12		1 1398/03/1	neozafat	undefined	301	101						
13		1 1398/03/2	ghabz	gas	605	46						

۱. در کنسول

۲. در CSV

سهم بخش‌ها

سهم هزینه‌های زیرگروه‌های یک دسته بندی در آن دسته بندی

برنامه قادر است سهم هزینه‌های هر کدام از زیر گروه‌ها نسبت به کل هزینه‌های آن دسته را در قالب یک DataFrame چاپ کند. یعنی مجموع هزینه‌ی هر کدام از زیر دسته‌ها که فقط مربوط به دسته‌ی قبض است را نسبت به هزینه‌ی کل قبوض چاپ میکند.

سهم هزینه‌های دسته بندی نسبت به کل

برنامه قادر است سهم هزینه‌های هر کدام از دسته‌ها را نسبت به کل هزینه‌ها را در قالب یک DataFrame چاپ کند. یعنی مجموع هزینه‌ی هر کدام از دسته‌ها را نسبت به کل هزینه‌ها چاپ می‌کند.

```
سهم هزینه‌های زیرگروه‌های یک دسته‌بندی در آن دسته‌بندی به شکل زیر است
   gas    water  electricity  avarex
0  0.213664  0.325504    0.088669  0.372163

سهم هزینه‌های دسته‌بندی نسبت به کل به شکل زیر است
   ghabz  elevator  parking  nezafat  tamirat  sharj
0  0.479934      0.0    0.011063  0.263991  0.06142    0.0
```

۱. دو DataFrame

وضعیت موجودی ساختمان

این بخش یکی از چالش برانگیزترین بخش‌های این پروژه بود. اینکه چه زمانی وضعیت یک ساختمان طبیعی است و چه زمانی نامطلوب بحث برانگیز است اما دوست دارم توجیه راه این گروه را با یک سوال آغاز کنم (اگر هر کدام از این تراکنش‌ها توصیه نشود چه می‌شود؟) به نظر نمی‌آید که تعمیرکار آسانسور بعد از دو ماه توصیه نشدن به ساختمان برگردد و آن را خراب کند. اما برق یک ساختمان به راحتی بعد از همان دو ماه قطع می‌شود.

در این بخش برنامه بعد از بازه‌ی زمانی معینی که کاربر تراکنش‌های ساختمان را در آن مدت ثبت کرده وضعیت ساختمان و واحد(ها) با بیشترین بدهکاری را به مدیر معرفی می‌کند. روش بررسی وضع ساختمان به این شرح است:

مجموع هزینه و پرداختی قبض دو ماه اخیر که از تراکنش‌های مهم زمان حال است را حساب می‌کند و یک بازه‌ی احتمال ۲۰ درصدی برای خرج‌های غیرمنتظره تعریف میکند و میگوید اگر موجودی ساختمان از ۸۰ درصد مجموع هزینه و پرداختی قبض دو ماه اخیر کمتر باشد وضع ساختمان مطلوب است و در غیر این صورت یا اضطراری است و یا نسبتاً نامطلوب.

وضعیت این ساختمان بعد از ۲ سال و ۳ ماه از زمان اولین تراکنش به شرح زیر است
وضع ساختمان اضطراریست.
برای آگاهی از واحد(ها) با بیشترین بدهی 'ادامه' را وارد کنید.
ادامه
لطفاً به واحدهای ۵ و ۸ رجوع کنید.

در اعلام دو یا یک واحدی که بیشترین بدهی را دارند توجه داشته باشید اگر بدهی همه واحدها یکی بود عبارت(همه ی واحدها به یک اندازه بدهکارند) چاپ می‌شود و در سایر مواقع اگر دومین واحد با بیشترین بدهی حداقل ۸۰ درصد بدهی واحد اول را داشته باشد در کنار واحد اول گزارش می‌شود و در غیر این صورت فقط واحد اول گزارش می‌شود.

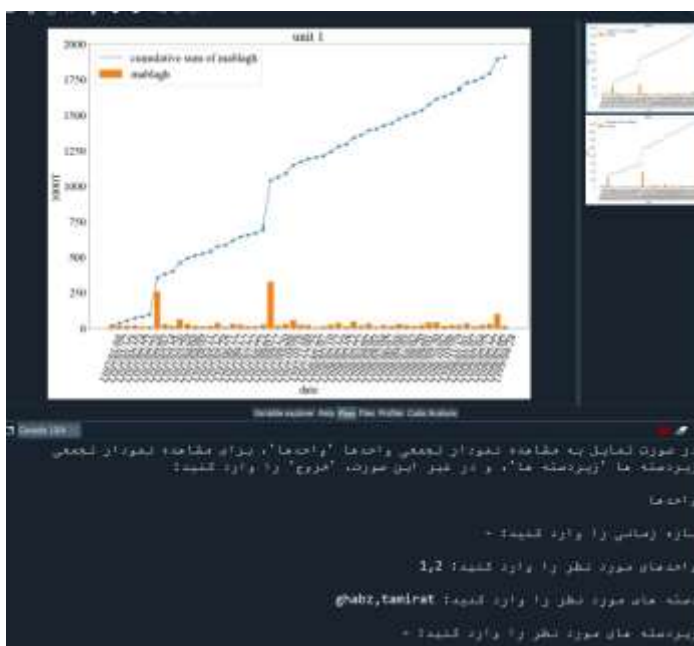
روند هزینه ها

نمودار تجمعی هزینه های واحدهای مختلف

برنامه می تواند نمودار تجمعی واحدهای دلخواه در بازه ی زمانی دلخواه و دسته و زیردسته ی دلخواه کاربر رسم کند و در شرایطی که در یک تاریخ بیش از یک تراکنش داشتیم جزئیات بیشتر را به کاربر نشان دهد و این کار به درک روند مالی ساختمان توسط مدیر کمک می کند.

در صورت تمایل به مشاهده زیر هزینه های تاریخ مابقی که شامل چند تراکنش میشود،
"مشاهده" را وارد کنید، و در غیر این صورت، "خروج" را وارد کنید: مشاهده

vahed	tarikh	mande	sharh
0	1	2018/11/28	24 gas
1	1	2018/11/28	25 water
2	1	2020/03/20	21 water
3	1	2020/03/20	16 electricity
4	2	2018/09/30	14 electricity
5	2	2018/09/30	12 water
6	2	2020/03/20	16 electricity
7	2	2020/03/20	11 water



نمودار تجمعی هزینه های مربوط به زیرگروه های مختلف

برنامه می تواند نمودار تجمعی زیردسته های دلخواه در بازه ی زمانی دلخواه کاربر رسم کند و در این بخش نیز اگر چند تراکنش در یک تاریخ رخ دهند، جزئیات آنها در یک جدول را به کاربر نشان داده می شود؛ این کار به مدیر کمک میکند تا روند مالی ساختمان را بهتر درک کند.



پیش بینی هزینه ی پرداختی هر واحد در سال آینده

برای پیش‌بینی شارژ ماهانه‌ی واحدها تمام هزینه‌ها در مدت زمان تراکنش‌ها را بر تعداد ماه‌های این مدت تقسیم کردیم و به اندازه‌ی ۱۰ درصد آن مقدار برای هزینه‌های غیرمنتظره به آن اضافه کردیم و این مقدار را برای تمامی واحدها در کنسول چاپ کردیم.

*چون تقسیم بندی‌ها را قبلا انجام داده بودیم دیگر نیاز به تقسیم بندی نبود و هزینه‌های واحدها همه

قبلا تقسیم شده بودند.

شارژ واحدها در سال آتی به شرح زیر است	
برآورد شارژ واحد 1	تقریبا برابر 171 است
برآورد شارژ واحد 2	تقریبا برابر 161 است
برآورد شارژ واحد 3	تقریبا برابر 186 است
برآورد شارژ واحد 4	تقریبا برابر 215 است
برآورد شارژ واحد 5	تقریبا برابر 278 است
برآورد شارژ واحد 6	تقریبا برابر 189 است
برآورد شارژ واحد 7	تقریبا برابر 176 است
برآورد شارژ واحد 8	تقریبا برابر 272 است
برآورد شارژ واحد 9	تقریبا برابر 207 است
برآورد شارژ واحد 10	تقریبا برابر 211 است

بیان ایده های خلاقانه و کارهای اضافی ای که انجام داده اید

در قسمت های مختلف این پروژه از ایده هایی متفاوت و تا حدی خلاقانه استفاده شده که در ادامه به شرح مفصلی از آن ها می پردازیم.

اولین ایده خلاقانه مربوط به تابع توزیع هزینه ی گاز بود. ما تصمیم گرفتیم که هزینه گاز را با توجه به مترآژ واحد و تعداد ساکنین توزیع کنیم؛ البته با توجه به اینکه قطعا مترآژ تاثیر بیشتری دارد، از تعداد ساکنین هر واحد جذر گرفتیم و جذر آن را در مساحت آن واحد ضرب کردیم. در نهایت عدد بدست آمده برای هر واحد را در ستونی جدید ذخیره کردیم و بعد برای تقسیم هزینه بین چند واحد، مبلغ کل را در نسبتی که در ستون جدید برای آن واحد تعریف شده است ضرب کرده و آن را به جمع کل اعداد موجود در ستون جدید تقسیم می کنیم. همچنین برای تقسیم هزینه های آسانسور تابعی تعریف کردیم که طبقات بالایی باید هزینه ی بیشتری بپردازند. که در قسمت "اشاره به کد" توضیح داده شده است.

یکی دیگر از ایده های خلاقانه مربوط به نوع ورودی گرفتن برنامه است. یعنی مدیر ساختمان بدون نیاز به نوشتن تمام ورودی ها در کنسول به راحتی از یک فایل `txt` استفاده می کند و تراکنش های مالی ساختمان را در آن می نویسد و برنامه ما با خواندن همین فایل ورودی ها را به فرم قابل استفاده خود تبدیل می کند. این کار می تواند از پیچیدگی فضای برنامه نویسی برای کاربری که کمتر با این محیط آشناست بکاهد و از نظر ما این روش کاربرپسندتر خواهد بود. البته بعد از اینکه دستیاران آموزشی فایل های مربوط به داده های یک ساختمان را به ما تحویل دادند بخشی را به کد اضافه کردیم که بتواند این فایل را به متن `txt` که برنامه ما از قبل با آن کار می کرد تبدیل کند.

به منظور تحویل صورتحساب، با استفاده از تابع `TRY EXCEPT`، برنامه ابتدا سعی میکند فایلی به اسم `bils.csv` را بخواند؛ وقتی برنامه برای اولین بار اجرا شده باشد، تبعا چنین فایلی وجود ندارد و قسمت `try` با ارور مواجه میشود، در این صورت قسمت `except` اجرا میشود که در این قسمت `bills` به صورت یک دیکشنری

خالی با کلیدهای معین تعریف می‌شود و با توجه به اینکه **value** این کلیدها از نوع لیست تعریف شده اند، برنامه در ادامه ورودی‌ها را به این لیست‌های خالی اضافه می‌کند و در نهایت این دیکشنری به صورت فایل **bills.csv** ذخیره می‌شود. در دفعات بعدی که برنامه اجرا میشود، قسمت **try** با ارور مواجه نمی‌شود (زیرا **bills.csv** وجود دارد و برنامه میتواند آن را بخواند)، و اطلاعات جدید به اطلاعات قبلی *اضافه* میشوند و برنامه در ادامه، با کل داده‌ها (داده‌های قدیمی و جدید) کار می‌کند.

نوشتن کل بخش مطلوب بودن وضع ساختمان و بخش پیشبینی شارژ سال آینده، بر اساس خلاقیت بود که شرح مفصل ایده آن‌ها در بخش "نمونه خروجی بصری" آمده است.

همچنین همانطور که در شیوه نامه گفته شده بود، به عنوان یک کار امتیازی کد نهایی را در **git hub** گذاشتیم.