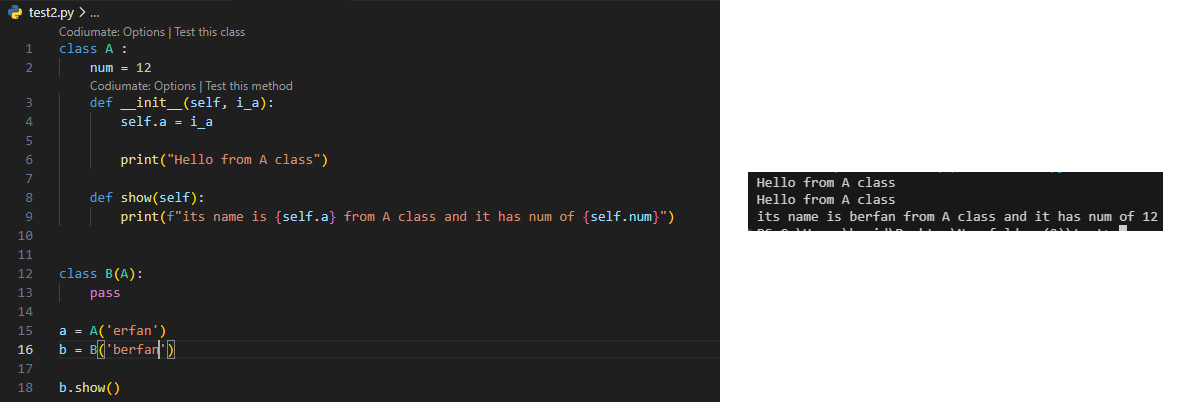
ارث بری در کلاس ها

---------------------------------------------------------------------------------------

وقتی یک کلاسی از یک کلاس دیگه ارث بری میکنه تمام ویژگی های اون رو از متغیر ها توابع و ... رو به ارث میبره :

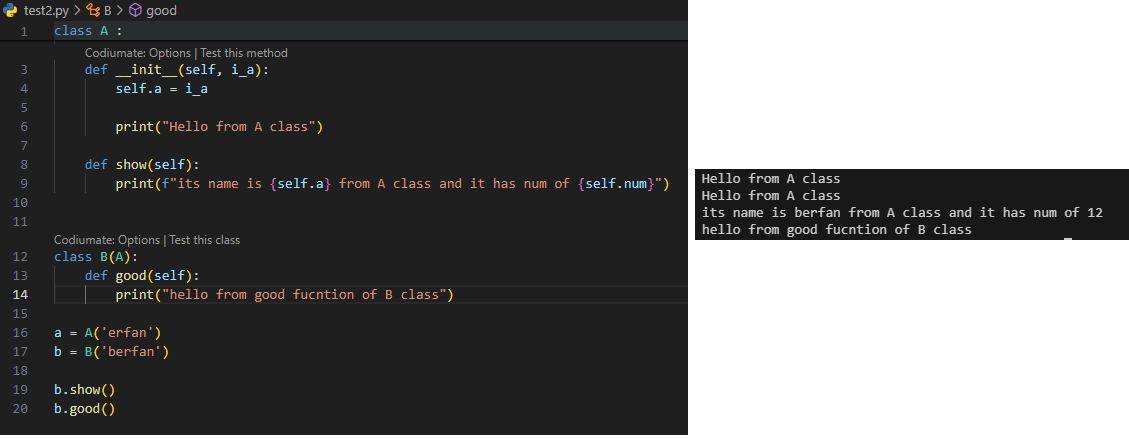


توی کلاس B من هیچی ننوشتم ولی تمام ویژگی های A رو داره ارث بری میکنه حتی موقعی که دارم b رو میسازم ازم ارگومان i\_a رو به عنوان ورودی میخاد ! . تابع show رو هم روی b اجرا کردم ولی توی B هیچی نیست ! به این میگن ارث بری

به کلاسی که ارث میبره فرزند میگن و به کلاسی که ارث میده میگن والد

---------------------------------------------------------------------------------------

حواستون باشه که کلاس فرزند از داشتن ویژگی های منحصر به فرد محروم نمیشه :



اینجا کلاس B که فرزنده یک ویژگی تابعی به اسم good هم گرفته که فقط برای خودشه

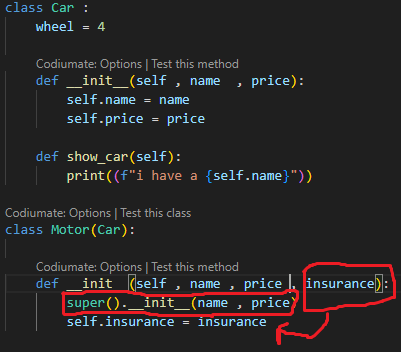
---------------------------------------------------------------------------------------

حالا شاید ما بخواییم فقط بخشی از ویژگی های والد رو بگیریم , یعنی فقط یکسری ویژگی ها به ارث برن :

فرض کنید که یک کلاس ماشین داریم که مقادیر اسم و قیمت رو از ورودی میگیرن , حالا میخاییم علاوه بر گرفتن اون ویژگی های والد ( اسم و قیمت ) وضعیت بیمه رو هم مشخص کنیم ( فقط برای کلاس موتور ) :

Syntax : super().adjectives

Overwrite :



دقت کنید :

Super().\_\_init\_\_(name , price) -> میگه که از باباعه ! برو سراغ داندر اینیتش و از اون اسم و قیمتش رو به ارث ببر

حواستون باشه که اون name و price باید توی ارگومان تابع فرزند هم قرار بگیره تا به \_\_init\_\_ خودشم برسه ! به الاوه ویژگی های منحصر به فرد خودش

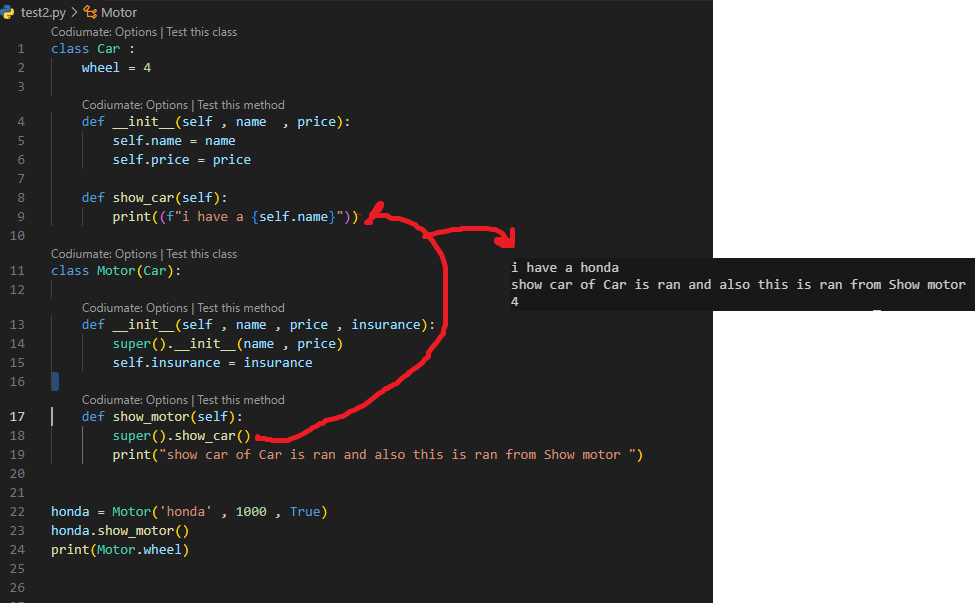
Insurance یک ویژگی منحصر به فرد برای خود کلاس Motor هست

ما الان اومدیم و متود \_\_init\_\_ تابع فرزند رو بازنویسی (overwrite) کردیم

یکبار دیگه توضیح بدم چون نفهمیدید :   
  
میگم اهای ای کلاس Motor تو برو متود \_\_init\_\_ خودتو بساز که ویژگی name و price و insurance داره , خط بعد super() : اون ویژگی name و price تو از داندر اینیت تابع والدت بگیر . self.insurance = insurance : این ویژگی رو هم منحصر به فرد برای خودت اضافه کن

-----------------------------------------------------------------------------------------------

پس وقتی داریم یک ویژگی والد رو overwrite میکنیم , از داخل سوپر صداش میزنیم و اون ویژگی هایی که میخایم رو ازش میاریم . هرچی رو میشه به ارث برد :



داخل تابع show\_motor اومدم گفتم که : توی کلاس والدت یه تابع show\_car هست اول اونو اجرا کن بعد اون بیل بیلک رو پرینت کن . اخرم که پایین کد صداش زدم

!! حواستون باشه : توی تابع Car که موتور ازش ارث بری کرده نوشتم :

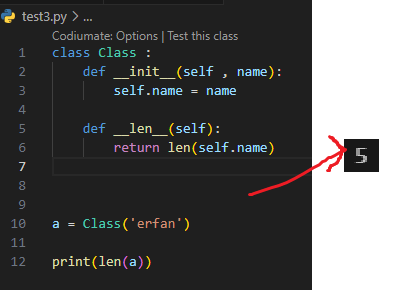
Print(f’i have a {self.name}’)

این self.name که هست چون توی کلاس Motor داره اجرا میشه پس هوندا چاپ میکنه نه اون self.name که توی کلاس Car هست مثلا پراید .

---------------------------------------------------------------------------------------

\_\_len\_\_ :

میگه هروقت از روی ابجکت ها len گرفته شد فلان رفتارو نشون بده :

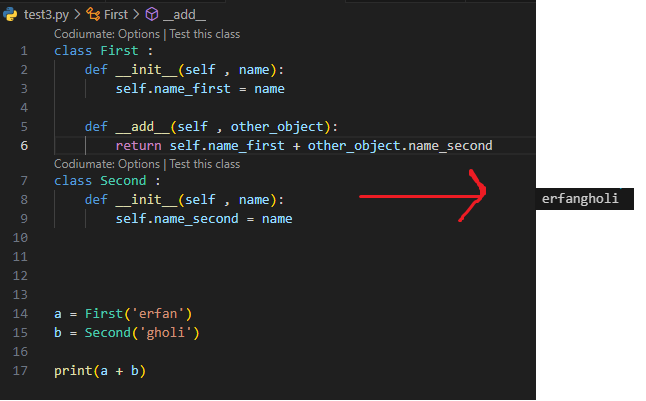


گفتم اگر از روی ابجکت len گرفتن , len اون self.name رو بهش بده

---------------------------------------------------------------------------------------

\_\_add\_\_ :

میگم وقتی ابجکت ما با یه ابجکت دیگه ای جمع شد فلان رفتارو نشون بده :



با دقت بخونید : گفتم اگرآبجکتی از کلاس First با آبجکتی از کلاسی دیگر جمع شد فلان رفتار رو نشون بده : توی ارگومان هاش یدونه سلف داریم که هیچی . یدونه other\_object هم گذاشتیم چرا ؟ وقتی 2 تا ابجکت جم میشن , داندر add اولی اجرا و دومی به عنوان ارگومان به داندر add اولی ارسال میشه . حالا : گفتم که return کن : self.name\_first که یعتی اون متغیر رو از همون ابجکت ( که داندر صدا زده شده ازش ) به اضافه اون یکی ابجکته که یه متغیری داره به اسم name\_second بکن .

Try except :

بعضی از مواقع ما میخاهیم اگر اروری دریافت شد اون رو شخصی سازی کنیم یا اصلن اجازه ندیم که ارور برنامه رو متوقف کنه :

Try :

Statement

Except :

Statement

حالا : میاد اون کد داخل try رو اجرا میکنه و اگر اروی دریافت کرد طبق رفتار کد های داخل except عمل میکنه :



a متدی به اسم upper ندارد پس ارور برمیگردونه در حالت عادی ولی من گفتم که اگر ارور برگردوند (except) : بیا چاپ کن اون بیل بیلکه رو . ( برنامه متوقف نمیشه )

بعضی از مواقع ما میخاییم که ارور هامون برنامه رو متوقف بکنن ! مثل موقعی که طرف ورودی اشتباه داد ولی بعضی از مواقع مثل اینجا من نمیخام برنامه متوقف بشه و فقط یه پیغامی چاپ بشه و به راه خودش ادامه بده .

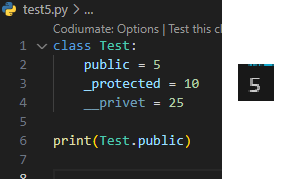
Name mangling:

متغیر ها در کلاس ها به 3 دسته تقسیم میشوند :

1-public : متغیر های عمومی معممولی که تعریف میکنیم

2-protected : متغیر هایی که از لحاظ ظاهری حفاظت شدن که یک اندرسکور دارند

3-privet : متغیر هایی که خارج از کلاس قابل دسترس نیستند که دو اندرسکور دارند

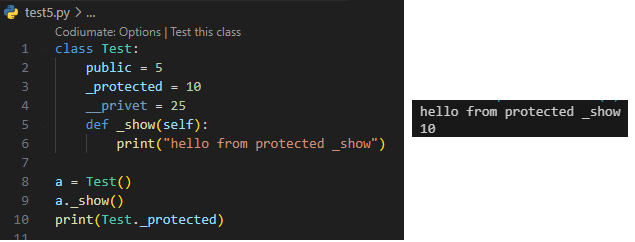


X

متغیر پابلیک که قضیه ای نداره

--------------------------------------------

حالا پروتکتد رو چجور باید صدا زد ؟

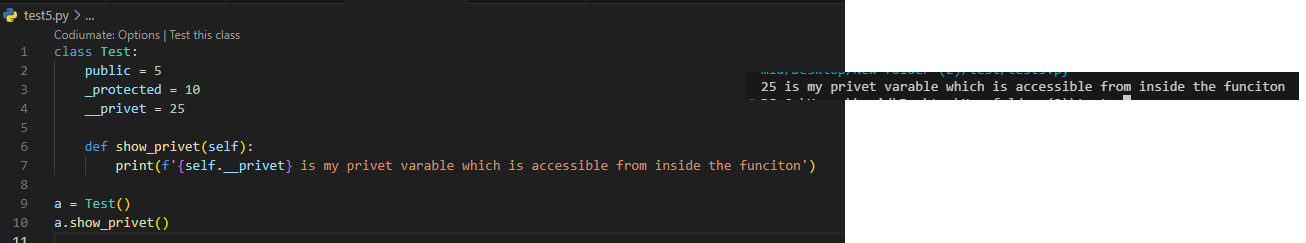


فقط کافیه اسم کامل متود یا متغیر رو ( همراه با \_ اش ) صدا بزنی

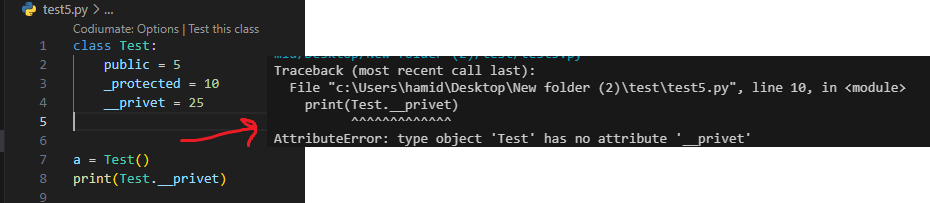
\*\* این فقط جنبه ظاهری داره که وقتی داره کد شمارو میخونه بدونه که این یه متغیری هست که نباید دست بهش بزنه یا تغییر اساسی توش بده \*\*

--------------------------------------

متغیر privet : این متغیر یا متود هارو هیج جوره خارج از کلاس نمیشه دسترسی داشت بهشون :



همون طور که دارید میبینید از داخل کلاس در دسترسه اومااا :



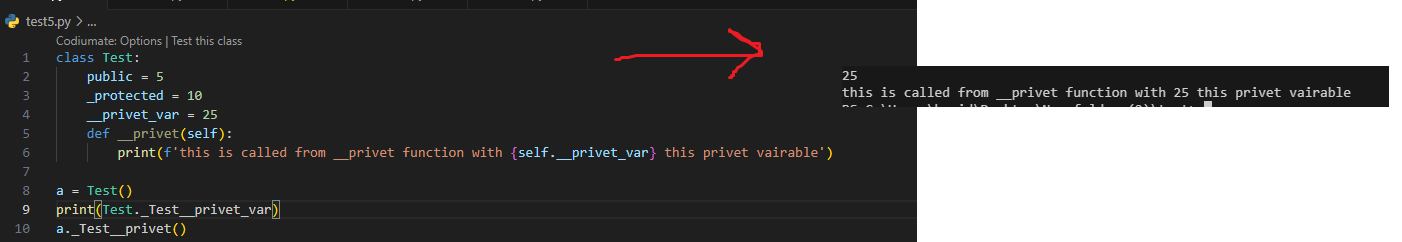
اینجا از خارج از تابع ولی در دسترس نیست ارور میده .

حالا چو کنیم برای دسترسی ؟

Syntax for variable : Class.\_Class\_\_variable

Syntax for Function: Object.\_Class\_\_function

مثال :



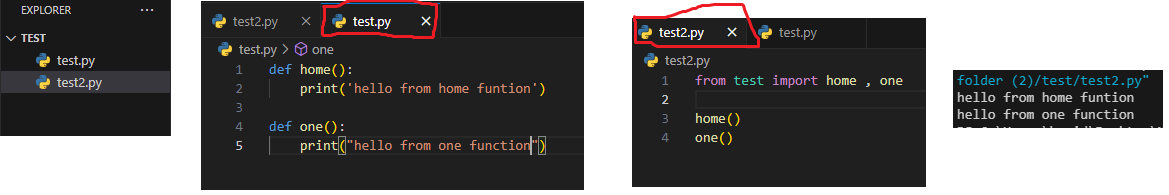
هم اون متغیر رو چاپ کردم هم اون تابع رو صدا زدم

\*\*\* تابع از روی ابجکت صدا زده میشه نه خود کلاس \*\*\*

File management

-----------------------------------------------------------------------------------------------

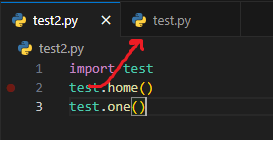
ما همیشه از یک فایل برای پایتون استفاده نمیکنیم و چندین فایل دخیل هستند در پروژه های ما . ما همیشه توابع و کلاس هایمون رو جدا مینویسم و داخل فایل اصلی پروژه وارد میکنیم :



در اینجا من داخل فایل test اومدم توابع رو نوشتم ولی در فایل test2 اون هارو Import کردم و صدا زدم . توی تصویر آخر هم اگر ببینید اون آبیه رو فایل test2 رو اجرا کنم ( چون توی فایل 2 صدا زدم ) و توابع اجرا میشوند.

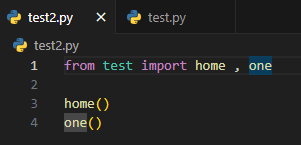
روش ها صدا زدن :

1- کلن هرچی داره وارد کنی و بعد قبل از صدا زدن اسم فایل رو هم بیاری :



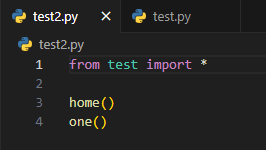
وقتی مینیوسم test.home() یعنی برو توی فایل home و test رو اجرا کن

2- ایمپورت دونه دونه :



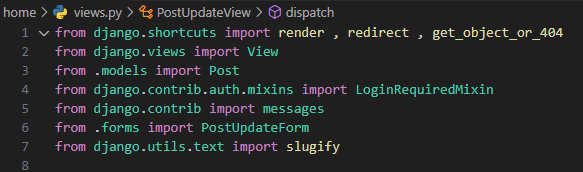
\*\* دیگه نیازی به صدا زدن فایل نیست چون خودشو اوردیم \*\*

مشابه روش 1 کاری میشه کرد که همه رو بیاریم ولی دیگه نخاد اسم فایل رو صدا بزنیم :



وقتی میگم ستاره یعنی هرچی توشه بردار بیار

!!!!! در دنیای واقعی ما هیچوقت همه چی رو برنمیاریم بیاریم چون الان پروژه کوچیکه ولی مثلا کتابخونه بزرگی باشه یا فریم ورک بزرگی باشه اگر هرچی داره برداریم بیاریم پروژه بهینه نمیشه و سرعت خیلیییییییییییی میاد پایین :



این یک پروزه جنگوعی هست . میبینید که هر کدام از ابزار ها دونه دونه وارد میشه

---------------------------------------------------------------------------------------

درباره اونی که نوشته .model یا .forms که داره به فایل پایتونی اشاره میکنه هم نپرسید که الان وقتش نیست

---------------------------------------------------------------------------------------