1. نوشتن مسئله:

با استفاده از Z3 جدول سودوکو ارائه شده را حل نمایید. منابع پیشنهادی در ادامه بیان شده است. کد پیاده سازی شده را در گیت هاب به صورت مرحله به مرحله قرار دهید (مبانی مهندسی نرم افزار). مراحل تحلیل و پیاده سازی بایستی مطابق بیان روشهای رسمی باشد و ارائه کد و توضیح آن بدون رعایت مفاهیم روش های رسمی مورد پذیرش نمی باشد.

				6	1			2
	7						6	
9	2							
		4	5	2		9		
	8	2	1		4	6	3	
		3		7	6	1		
							9	8
	3						4	
6			3	8				

2. بیان مسئله:

جدول سودوکویی ارائه شده است که ما باید با استفاده از روش Z3، آن را حل کنیم و کد نهایی، مراحل پیاده سازی و تحلیل را (با توجه به مبانی روش های رسمی) در گیت هاب آپلود کنیم.

3. نیازمندی ها (Requirements):

- **3-1 سودوکو:**

سودوکو یک پازل منطقی است که معمولاً در روزنامه‌ها، مجلات، کتاب‌ها و برنامه‌های موبایل یافت می‌شود. این بازی شامل یک جدول دو بعدی مربعی 9 در 9 است که به 81 خانه کوچکتر تقسیم شده است. هدف در سودوکو، پر کردن این جدول با اعداد از 1 تا 9 است به گونه‌ای که شرایط زیر برقرار شود:

a. هر سطر باید شامل اعداد 1 تا 9 باشد و هر عدد فقط یک بار در سطر ظاهر شود.
b. هر ستون باید شامل اعداد 1 تا 9 باشد و هر عدد فقط یک بار در ستون ظاهر شود.
c. هر بلوک 3 در 3 که درون جدول وجود دارد، باید شامل اعداد 1 تا 9 باشد و هر عدد فقط یک بار در بلوک ظاهر شود.

پازل سودوکو با شروع از یک جدول ناقص شروع می‌شود، که برخی از خانه‌ها در آن با اعداد پر شده‌اند و برخی خالی هستند. بازیکن باید خانه‌های خالی را با اعداد مناسبی پر کند تا شرایط سودوکو برقرار شود و پازل حل شود.

حل سودوکو نیازمند استفاده از استدلال منطقی و تحلیل است. بازیکن باید الگوها و قواعد منطقی را در نظر بگیرد تا به ترتیب صحیح اعداد را در جدول قرار دهد. این می‌تواند شامل تعیین اعداد محتمل برای خانه‌های خالی، استفاده از اصول منطقی مانند "عدد منحصر به فرد" در سطرها و ستون‌ها، و شناسایی الگوهای تکراری در بلوک‌های 3 در 3 باشد.
هر سودوکو دارای یک راه‌حل یکتا است، که به وسیلهٔ استدلال منطقی و قواعد سودوکو قابل یافتن است. بازیکنان می‌توانند از روش‌های مختلفی برای حل سودوکو استفاده کنند، از جمله استفاده از تکنیک‌های پیشرفته‌تر مانند "تکنیک XYZ-Wing" یا استفاده از روش‌های آزمایش و خطا.
سودوکو یک بازی پرفایده است. این بازی نه‌تنها سرگرم‌کننده است، بلکه مهارت‌هایی مانند تمرکز، تحلیل منطقی، حافظه و صبر را تقویت می‌کند.

- **Z3 3-2:**

- SMT چیست؟

SMT یک مسئله تصمیم گیری است که بررسی می کند آیا یک فرمول منطقی با توجه به قوانین (نظریه های) پس زمینه خاص قابل تحقق است. این تئوری ها می توانند شامل مواردی مانند حساب (اعداد)، بردارهای بیتی (داده های باینری)، آرایه ها (ساختارهای داده) و حتی توابع تعریف نشده باشند.

- Z3 چیست؟

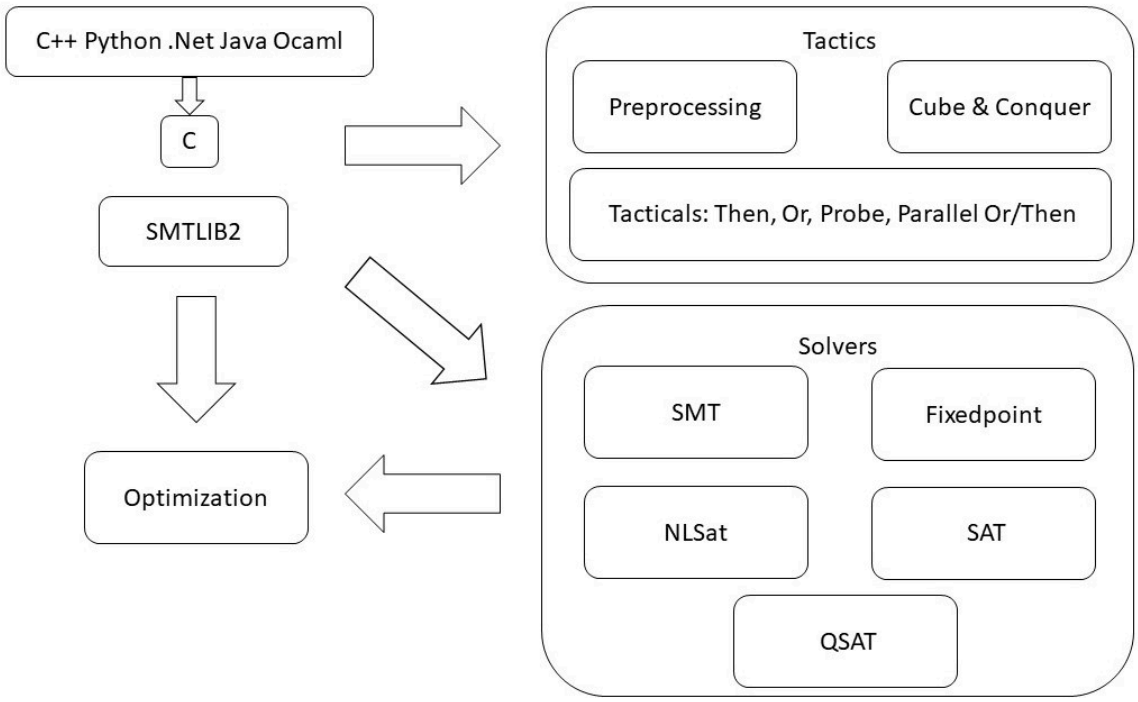
Z3 یک ابزار قدرتمند است که مشکلات SMT را حل می کند. این ابزار دارای الگوریتم های تخصصی برای مدیریت کارآمد تئوری های پس زمینه مختلف است.

- چرا SMT مفید است؟

SMT نقش مهمی در تحلیل، تایید و اجرای نمادین نرم افزار ایفا می کند. این به این دلیل است که SMT می تواند انواع داده ها و عملیات رایج در برنامه ها و مشخصات آنها را مدیریت کند.
ابزارهای نرم افزاری چگونه از SMT استفاده می کنند؟
ابزارها می توانند بخش هایی از وظایف تجزیه و تحلیل خود را به فرمول های SMT تبدیل کنند. درک منطق های پشتیبانی شده (نظریه ها) و نحوه حل فرمول های SMT برای توسعه دهندگان این ابزارها ضروری است.

- فراتر از فرمول های تکی:

گاهی اوقات، حل یک مشکل ممکن است نیاز به تعاملات متعدد با یک حل کننده SMT داشته باشد، نه فقط یک فرمول واحد. توسعه دهندگان ابزار باید با روش ها و گزینه های موجود (کلیدها) در حل کننده های SMT آشنا باشند تا به نتایج مطلوب برسند.
به عبارت ساده، SMT با بررسی اینکه آیا شرایط خاصی تحت قوانین خاص قابل تحقق است، به ابزارهای نرم افزاری برای تجزیه و تحلیل و تایید برنامه ها کمک می کند. Z3 یک ابزار قدرتمند است که به حل کارآمد این مشکلات کمک می کند. درک نحوه عملکرد SMT برای توسعه دهندگان ابزارهای تحلیل و تأیید نرم افزار ضروری است.



تصویر ارائه شده در بالا، نموداری کلی از معماری Z3 (نسخه 4.8) را نشان می‌دهد. اجزای کلیدی آن عبارتند از:

- رابط ها (بالا سمت چپ):

اسکریپت‌های SMT-LIB2: فایل‌های متنی حاوی فرمول‌هایی که Z3 می‌تواند آنها را پردازش کند.
تماس‌های API: تعامل با Z3 از طریق زبان‌های برنامه‌نویسی سطح بالا (مانند پایتون).

- رابط کاربری پایتون:

روش اصلی مورد استفاده در سند برای تعامل با Z3.

- نحو انتزاعی(Abstract Syntax):

ساختار فرمول‌ها و اصطلاحاتی را که Z3 می‌تواند آنها را بپذیرد، توضیح می‌دهد.

- نظریه‌ها(Theories):

مجموعه قوانین منطقی پشتیبانی‌شده را که تفسیر نمادها را در فرمول‌ها تعریف می‌کنند، تعیین می‌کند.

- حل‌کننده‌ها(Solvers):

Z3 حل‌کننده‌های مختلفی را برای تعیین اینکه آیا یک فرمول با توجه به نظریه‌های مشخص شده قابل تحقق است یا خیر، ارائه می‌دهد.

- تاکتیک‌ها(Tactics):

Z3 تاکتیک‌هایی را برای دستکاری فرمول‌ها قبل از حل آنها ارائه می‌دهد. این می‌تواند شامل ساده‌سازی یا ایجاد زیرهدف‌ها باشد.

- بهینه‌سازی(Optimization):

Z3 می‌تواند مشکلاتی را حل کند که شامل یافتن مقادیر بهینه (حداقل یا حداکثر) بر اساس یک تابع هدف داده شده و محدودیت‌های خاص باشد.

- برشمردن پیامدها:

Z3 دارای الگوریتم‌های تخصصی برای شناسایی پیامدهای مهم (backbone literals) است که از یک فرمول ناشی می‌شود.

به طور کلی، این نمودار نمایی کلی از معماری Z3 را ارائه می‌دهد و اجزای مختلفی را که برای حل مشکلات SMT با هم کار می‌کنند، برجسته می‌کند.

- **3-3 منابع پیشنهادی:**

https://microsoft.github.io/z3guide/
https://theory.stanford.edu/~nikolaj/programmingz3.html
https://github.com/Z3Prover/doc/blob/master/examples/TutorialISPRAS2019.ipynb

- **3-4 Visual Studio Code :**

که به اختصار VS Code نیز شناخته می‌شود، یک ویرایشگر کد منبع است که توسط مایکروسافت برای سیستم‌عامل‌های ویندوز، لینوکس، مک و حتی وب‌سایت‌هایی که از وب‌گردگر پشتیبانی می‌کنند، توسعه یافته است. این نرم‌افزار قابلیت‌های زیادی را ارائه می‌دهد که در اینجا به برخی از مهم‌ترین آن‌ها اشاره می‌کنیم:
پشتیبانی از چندین زبان برنامه‌نویسی: ویژوال استودیو کد از زبان‌های برنامه‌نویسی مختلفی از جمله جاوا، جاوا اسکریپت، پایتون، C++ و بسیاری دیگر پشتیبانی می‌کند.
رنگ‌آمیزی سینتاکسی: VS Code کد شما را با رنگ‌های مختلف هایلایت می‌کند تا خواندن و درک آن را آسان‌تر کند.
تکمیل کد هوشمند: این قابلیت با تایپ کردن شما کلمات کلیدی و توابع را پیشنهاد می‌دهد و در بسیاری از موارد کل خط کد را به‌صورت خودکار کامل می‌کند.
دیباگ کردن: ویژوال استودیو کد به شما امکان می‌دهد تا کد خود را خط به خط اجرا کنید و اشکالات آن را پیدا کنید.
کنترل نسخه با Git: VS Code به طور مستقیم با Git که یک سیستم کنترل نسخه متن‌باز است، ادغام می‌شود و به شما امکان می‌دهد تا تغییرات کد خود را مدیریت کنید.

شخصی‌سازی: شما می‌توانید با تغییر تم‌ها، فونت‌ها و میانبرهای صفحه‌کلید، محیط ویرایش را مطابق با میل خود شخصی‌سازی کنید.
افزونه‌ها: VS Code دارای یک بازار پر رونق از افزونه‌ها است که قابلیت‌های بیشتری را به نرم‌افزار اضافه می‌کند. این افزونه‌ها می‌توانند برای پشتیبانی از زبان‌های برنامه‌نویسي خاص، فریم‌ورک‌ها و ابزارهای مختلف باشند.

برخلاف برخی از ویرایشگرهای کد ساده، Visual Studio Code یک محیط توسعه یکپارچه (IDE) کامل نیست. IDE ها معمولاً ویژگی‌های بیشتری مانند ابزارهای طراحی رابط کاربری گرافیکی و کامپایلرها را ارائه می‌دهند. با این حال، VS Code با قابلیت‌های غنی و انعطاف‌پذیری که دارد، به یکی از محبوب‌ترین ویرایشگرهای کد در بین توسعه‌دهندگان تبدیل شده است.

- **3-5 لپ تاپ:**

حداقل سیستم مورد نظر برای اجرای VSCode از نظر سخت افزاری باید به صورت زیر باشد:
رم : 1 گیگابایت
سی پی یو (CPU): پردازنده دو هسته ای
کارت گرافیک (GPU): رزولوشن 720*1280
فضای مورد نیاز: حدود 250 مگابایت

- **3-6 حساب github:**

حساب گیت‌هاب (GitHub account) یک حساب کاربری است که شما می‌توانید در پلتفرم گیت‌هاب ایجاد کنید. گیت‌هاب یک سرویس میزبانی مخزن کدهای منبع باز است که توسط توسعه‌دهندگان و تیم‌های برنامه‌نویسی برای مشارکت در پروژه‌های نرم‌افزاری استفاده می‌شود.
با ایجاد یک اکانت گیت‌هاب، شما می‌توانید مخزن‌های خود را بسازید، کدهای خود را بارگذاری (push) و به‌روزرسانی (commit) کنید، با دیگران در پروژه‌های مشترک همکاری کنید، مشکلات را گزارش دهید و برنامه‌های دیگر را مشاهده و تحلیل کنید. علاوه بر این، شما می‌توانید پروفایل کاربری خود را سفارشی کنید و نمایه‌ای از تجربه و توانایی‌های خود به دیگران ارائه دهید.
حساب گیت‌هاب به شما امکاناتی مانند مدیریت نسخه‌های کد، انجام تغییرات به صورت توزیع‌شده، مشارکت در پروژه‌های گروهی، بررسی و تصحیح کدها، ایجاد برنچ‌ها (branches) برای توسعه ویژگی‌های جدید و دسترسی به ابزارهای مدیریت پروژه مانند برگه‌های مسئولیت (issue trackers) را می‌دهد.
با داشتن یک حساب کاربری گیت‌هاب، شما می‌توانید به عنوان یک توسعه‌دهنده فعالیت کنید، کدهای خود را به صورت عمومی یا خصوصی به اشتراک بگذارید و با جامعه برنامه‌نویسی گیت‌هاب در ارتباط باشید. همچنین در این پروژه، برای بارگذاری کدها، احتیاج به حساب کاربری گیت هاب داریم.

4. اختصاصی سازی(Spesification):

- **4-1 دلایل استفاده از VS Code برای حل Z3:**

1. پشتیبانی از افزونه‌ها:
VSCode دارای یک اکوسیستم گسترده از افزونه‌ها است که امکانات مختلفی را به ویرایشگر اضافه می‌کنند. افزونه‌های متعددی برای Z3 نیز در دسترس هستند که کار با این ابزار را آسان‌تر و کارآمدتر می‌کنند. این افزونه‌ها می‌توانند شامل موارد زیر باشند:

برجسته‌سازی نحو Z3: این افزونه‌ها فرمول‌های Z3 شما را با رنگ‌های مختلف هایلات می‌کنند تا خواندن و درک آنها آسان‌تر شود.

پیشنهاد فرمول Z3: این افزونه‌ها با تایپ کردن شما فرمول‌های Z3 را پیشنهاد می‌دهند و در بسیاری از موارد کل فرمول را به‌صورت خودکار کامل می‌کنند.

اشکال‌زدایی Z3: این افزونه‌ها به شما امکان می‌دهند تا فرمول‌های Z3 خود را خط به خط اجرا کنید و اشکالات آنها را پیدا کنید.

ادغام با سایر ابزارها: این افزونه‌ها Z3 را با سایر ابزارهای توسعه، مانند ویرایشگرهای کد دیگر یا سیستم‌های کنترل نسخه، ادغام می‌کنند.

2. رابط کاربری بصری:
VSCode دارای یک رابط کاربری بصری و کاربر پسند است که کار با آن را آسان می‌کند. شما می‌توانید به راحتی فایل‌های Z3 خود را باز کنید، فرمول‌ها را ویرایش کنید، و نتایج را مشاهده کنید.

3. قابلیت‌های ویرایش قدرتمند:
VSCode دارای قابلیت‌های ویرایش قدرتمندی است که برای کار با فرمول‌های Z3 مفید است. این قابلیت‌ها شامل موارد زیر هستند:

جستجوی قدرتمند: شما می‌توانید به راحتی در بین فایل‌های Z3 خود و همچنین در فرمول‌هایتان جستجو کنید.

پشتیبانی از میانبرهای صفحه‌کلید: شما می‌توانید میانبرهای صفحه‌کلید را برای انجام وظایف رایج مانند ذخیره، باز کردن و جستجو سفارشی کنید.

کنترل نسخه با Git: VSCode به طور مستقیم با Git ادغام می‌شود و به شما امکان می‌دهد تا تغییرات کد خود را مدیریت کنید.

4. جامعه بزرگ:
VSCode دارای یک جامعه بزرگ و فعال از کاربران و توسعه‌دهندگان است. این به این معنی است که شما می‌توانید به راحتی در صورت بروز مشکل کمک پیدا کنید.

5. رایگان و منبع باز:
VSCode یک نرم‌افزار رایگان و منبع باز است. این به این معنی است که شما می‌توانید آن را به صورت رایگان دانلود و استفاده کنید و همچنین می‌توانید کد منبع آن را بررسی و اصلاح کنید.

در مجموع، استفاده از VSCode برای کار با Z3 به دلیل پشتیبانی از افزونه‌ها، رابط کاربری بصری، قابلیت‌های ویرایش قدرتمند، جامعه بزرگ و رایگان بودن آن، انتخابی عالی است.

- **4-2 نحوه حل سوال:**

در صورت سوال، یک جدول سودوکو در اختیار ما قرار داده شده است. حال ما باید کدی را با استفاده از Z3 بنویسیم که این جدول را (با توجه به تعاریف سودوکو) برای ما کامل کند.

0. نصب Z3:

برای استفاده از کتابخانه z3 کافیست که دستور :

pip install z3-solver

را در shell و یا ترمینال vscode وارد کنیم تا کتابخانه z3 در سیستم ما نصب شود.

سپس یک فایل به نام sudoku.py میسازیم و آن را در vscode باز میکنیم.

1. ایمپورت کتابخانه ها و تعریف نمونه سودوکو:

- from z3 import * :

تمام توابع کتابخانه Z3 را ایمپورت می‌کند.

- instance:

یک متغیر است که لیستی 9x9 را تعریف می‌کند. این لیست وضعیت اولیه معمای سودوکو را نشان می‌دهد. سلول‌های خالی با 0 مشخص شده‌اند.

لیست سودوکوی ما به صورت زیر تعریف می شود:

```
instance = ((0,0,0,0,6,1,0,0,2),
            (0,7,0,0,0,0,0,6,0),
            (9,2,0,0,0,0,0,0,0),
            (0,0,4,5,2,0,9,0,0),
            (0,8,2,1,0,4,6,3,0),
            (0,0,3,0,7,6,1,0,0),
            (0,0,0,0,0,0,0,9,8),
            (0,3,0,0,0,0,0,4,0),
            (6,0,0,3,8,0,0,0,0))
```

2. ایجاد متغیرهای صحیح برای هر سلول:

- X:

لیستی 2 بعدی از متغیرهای صحیح ایجاد می‌کند. هر متغیر X[i][j] مقدار سلول در ردیف i+1 (شماره‌گذاری از 1)، ستون j+1 (شماره‌گذاری از 1) در جدول سودوکو را نشان می‌دهد.

- Int("x_%s_%s" % (i+1, j+1)):

با استفاده از قالب‌بندی رشته، نام منحصر به فردی برای هر متغیر سلول تعریف می‌کند.

3. تعریف محدودیت ها:

- cells_c: تضمین می‌کند که هر سلول مقداری بین 1 تا 9 دارد.
- rows_c: از تابع Distinct برای اطمینان از اینکه هر ردیف شامل اعداد منحصر به فرد (1 تا 9) است استفاده می‌کند.
- cols_c: مشابه rows_c است اما برای اطمینان از اینکه هر ستون شامل اعداد منحصر به فرد است.
- sq_c: زیر شبکه‌های 3x3 را مدیریت می‌کند. از حلقه‌های لانه‌گذاری برای پیمایش هر زیر شبکه استفاده می‌کند و Distinct را اعمال می‌کند تا اطمینان حاصل شود که هر زیر شبکه شامل اعداد منحصر به فرد است.

4. ترکیب محدودیت ها و آغاز حل کننده:

- sudoku_c:

تمام محدودیت های تعریف شده قبلی (cols_c، rows_c، cells_c و sq_c) را ترکیب می‌کند.

- instance_c:

سلول‌های از پیش پر شده از لیست instance را به محدودیت تبدیل می‌کند. این تضمین می‌کند که این سلول‌ها مقادیر اولیه خود را حفظ کنند.

- s = Solver():

یک نمونه حل کننده از کتابخانه Z3 ایجاد می‌کند.

- s.add(sudoku_c + instance_c):

تمام محدودیت‌ها (هم قوانین سودوکو و هم سلول‌های از پیش پر شده) را به حل کننده اضافه می‌کند.

5. حل و چاپ نتیجه:

- if s.check() == sat:

بررسی می‌کند که آیا حل کننده می‌تواند راه حلی پیدا کند که تمام محدودیت‌ها را برآورده کند. sat یک ثابت در Z3 است که نشان دهنده یک حالت قابل قبول است.

    - `m = s.model()`:

اگر راه حلی وجود داشته باشد، مدل را از حل کننده بازیابی می‌کند. مدل شامل انتساب‌هایی برای همه متغیرهایی است که الزامات را برآورده می‌کنند.

- `r = [ ... ]`:

با ارزیابی مقادیر اختصاص داده شده برای هر متغیر سلول در مدل، شبکه سودوکو حل شده را ایجاد می‌کند.

- `print_matrix(r)`:

این خط احتمالاً یک تابع تعریف شده توسط کاربر برای چاپ شبکه سودوکو حل شده به روشی قالب‌بندی شده است (تعریف تابع ارائه نشده است).

- `else`:

اگر هیچ راه حلی پیدا نشد، "failed to solve" را چاپ می‌کند.

به طور کلی، این کد به طور موثر مشکل سودوکو را با استفاده از توابع Z3 به مجموعه ای از محدودیت ها تبدیل می کند. سپس حل کننده سعی می کند ترکیبی از مقادیر را برای هر سلول پیدا کند که تمام محدودیت ها را برآورده کند و منجر به راه حلی صحیح برای معمای سودوکو شود.

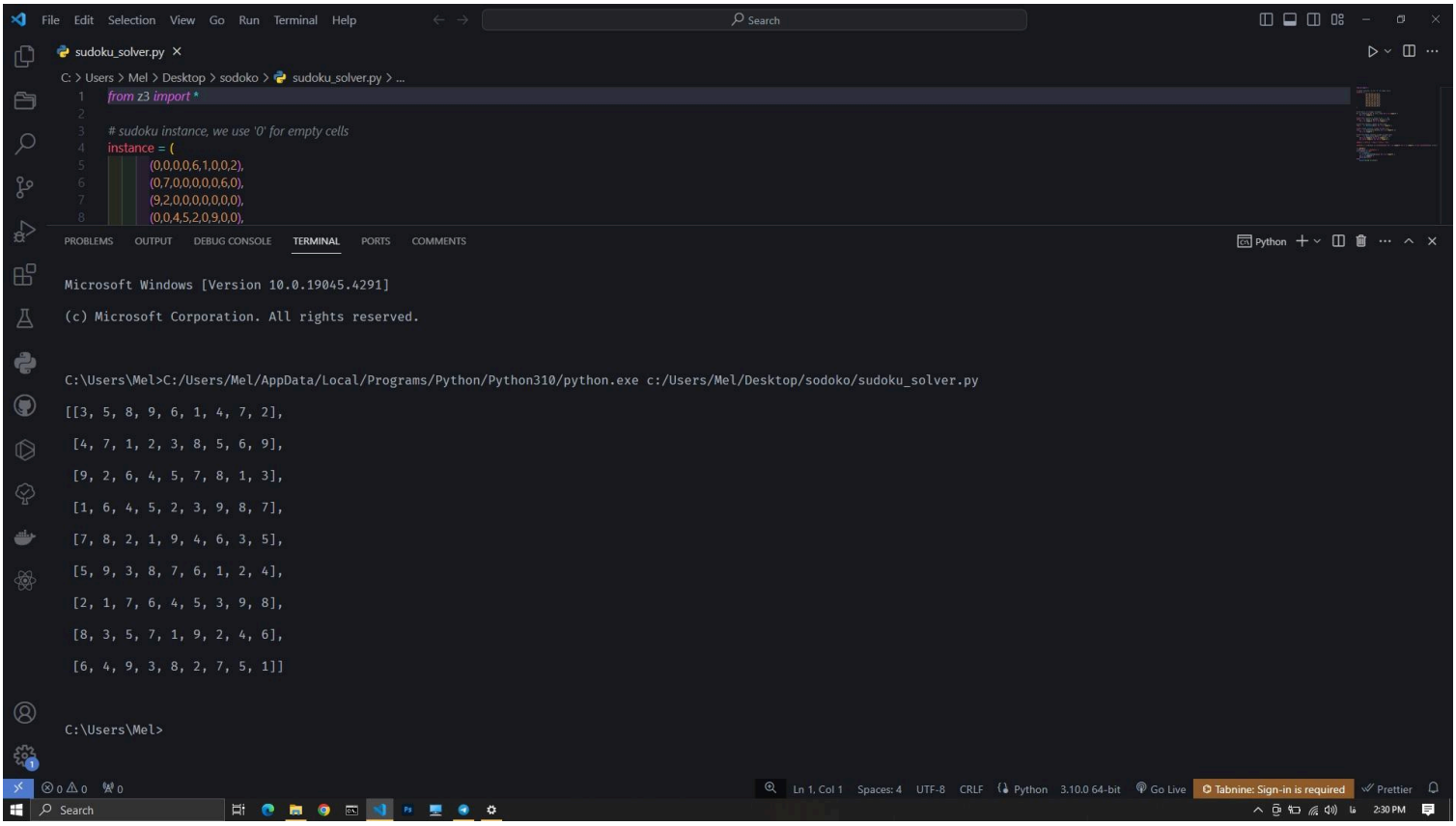
5. پیاده سازی(Implementation):

```
from z3 import *

# sudoku instance, we use '0' for empty cells
instance = (
        (0,0,0,0,6,1,0,0,2),
        (0,7,0,0,0,0,0,6,0),
        (9,2,0,0,0,0,0,0,0),
        (0,0,4,5,2,0,9,0,0),
```

```
        (0,8,2,1,0,4,6,3,0),
        (0,0,3,0,7,6,1,0,0),
        (0,0,0,0,0,0,0,9,8),
        (0,3,0,0,0,0,0,4,0),
        (6,0,0,3,8,0,0,0,0)
)


# 9x9 matrix of integer variables
X = [ [ Int("x_%s_%s" % (i+1, j+1)) for j in range(9) ]
  for i in range(9) ]

# each cell contains a value in {1, ..., 9}
cells_c  = [ And(1 <= X[i][j], X[i][j] <= 9)
  for i in range(9) for j in range(9) ]

# each row contains a digit at most once
rows_c   = [ Distinct(X[i]) for i in range(9) ]

# each column contains a digit at most once
cols_c   = [ Distinct([ X[i][j] for i in range(9) ])
  for j in range(9) ]

# each 3x3 square contains a digit at most once
sq_c     = [ Distinct([ X[3*i0 + i][3*j0 + j]
  for i in range(3) for j in range(3) ])
  for i0 in range(3) for j0 in range(3) ]

sudoku_c = cells_c + rows_c + cols_c + sq_c

instance_c = [ X[i][j] == instance[i][j] for i in range(9) for j in range(9) if not
(instance[i][j] == 0) ]

s = Solver()
s.add(sudoku_c + instance_c )
if s.check() == sat:
  m = s.model()
```

```
    r = [ [ m.evaluate(X[i][j]) for j in range(9) ]
    for i in range(9) ]
    print_matrix(r)
else:
    print("failed to solve")
```

- **Run:**



و اینگونه می توان یک جدول سودوکو را با استفاده از Z3 حل کرد.

(تفاوت سایز فونت ها و رنگ متن ها فقط به علت سهولت در خواندن است)