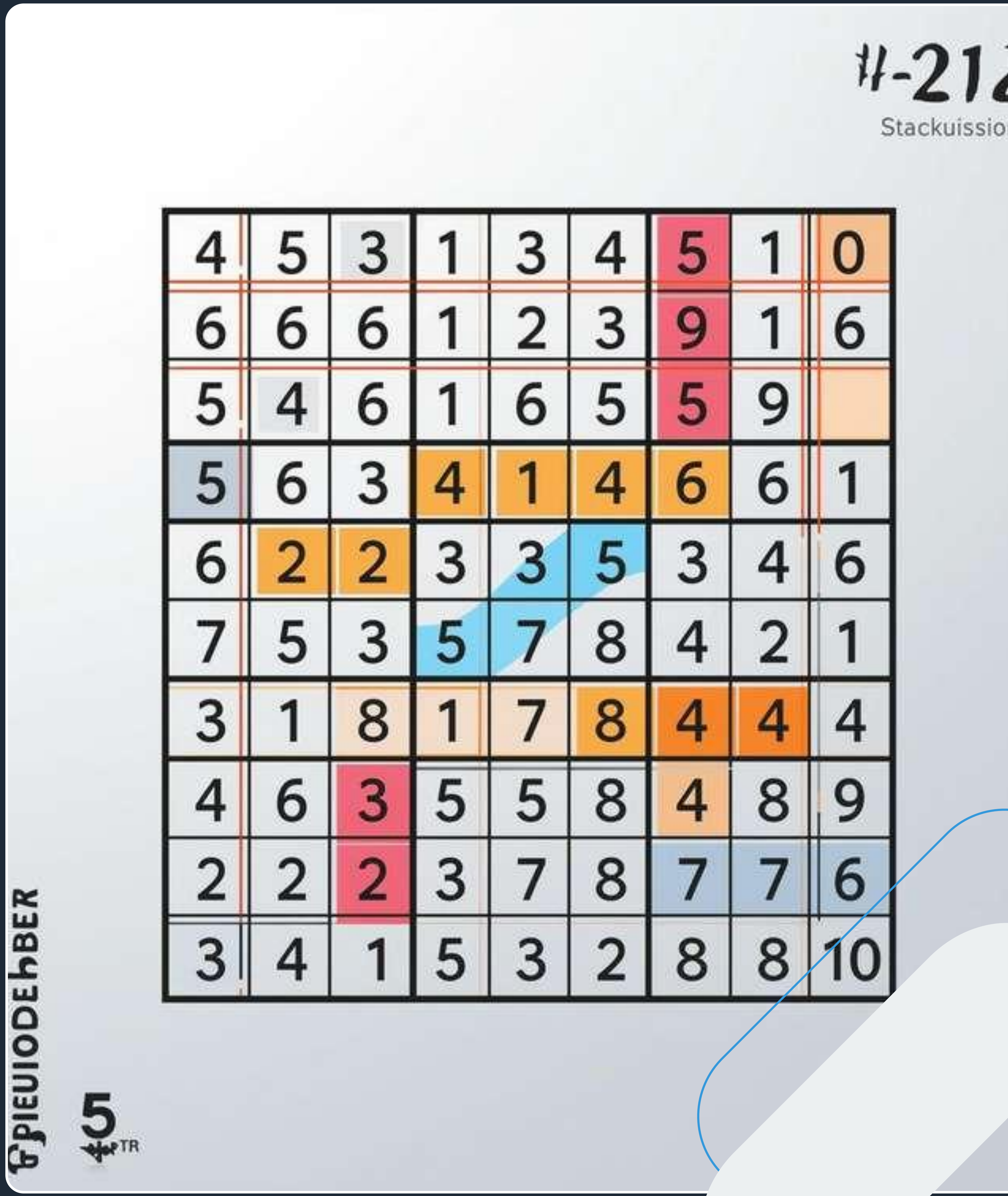
Sudoku Solver & Generator

MCA Project; Sameera Anjum



4	5	3	1	3	4	5	1	0
6	6	6	1	2	3	9	1	6
5	4	6	1	6	5	5	9	
5	6	3	4	1	4	6	6	1
6	2	2	3	3	5	3	4	6
7	5	3	5	7	8	4	2	1
3	1	8	1	7	8	4	4	4
4	6	3	5	5	8	4	8	9
2	2	2	3	7	8	7	7	6
3	4	1	5	3	2	8	8	10

Introduction: Sudoku - A Puzzle Phenomenon

Origin and Popularity in Various Formats

Sudoku has gained immense popularity as a brain-teasing puzzle, with various formats emerging globally. It challenges logic and enhances problem-solving skills, attracting enthusiasts of all ages.

History and Rules of Sudoku

Understanding Origins and Basic Gameplay

Sudoku originated in the late 18th century, gaining popularity worldwide. The rules are simple: fill a 9x9 grid with numbers 1-9 without repetition in rows, columns, and boxes.

Sudoku

Minta:: rendites cacenttle, kariple 7-1

1	2	3	9	4	5	1	9
1	6	3	1	5	1	8	9
7	2	5	8	2	6	4	7
1	8	0	1	2	1	7	8
5	5	6	7	4	5	6	7
5	2	1	9	6	6	1	5
4	5	2	7	3	2	4	5
7	2	2	6	3	7	2	9

any rephetceration wss ane autome fiall! ntiers, are boxes

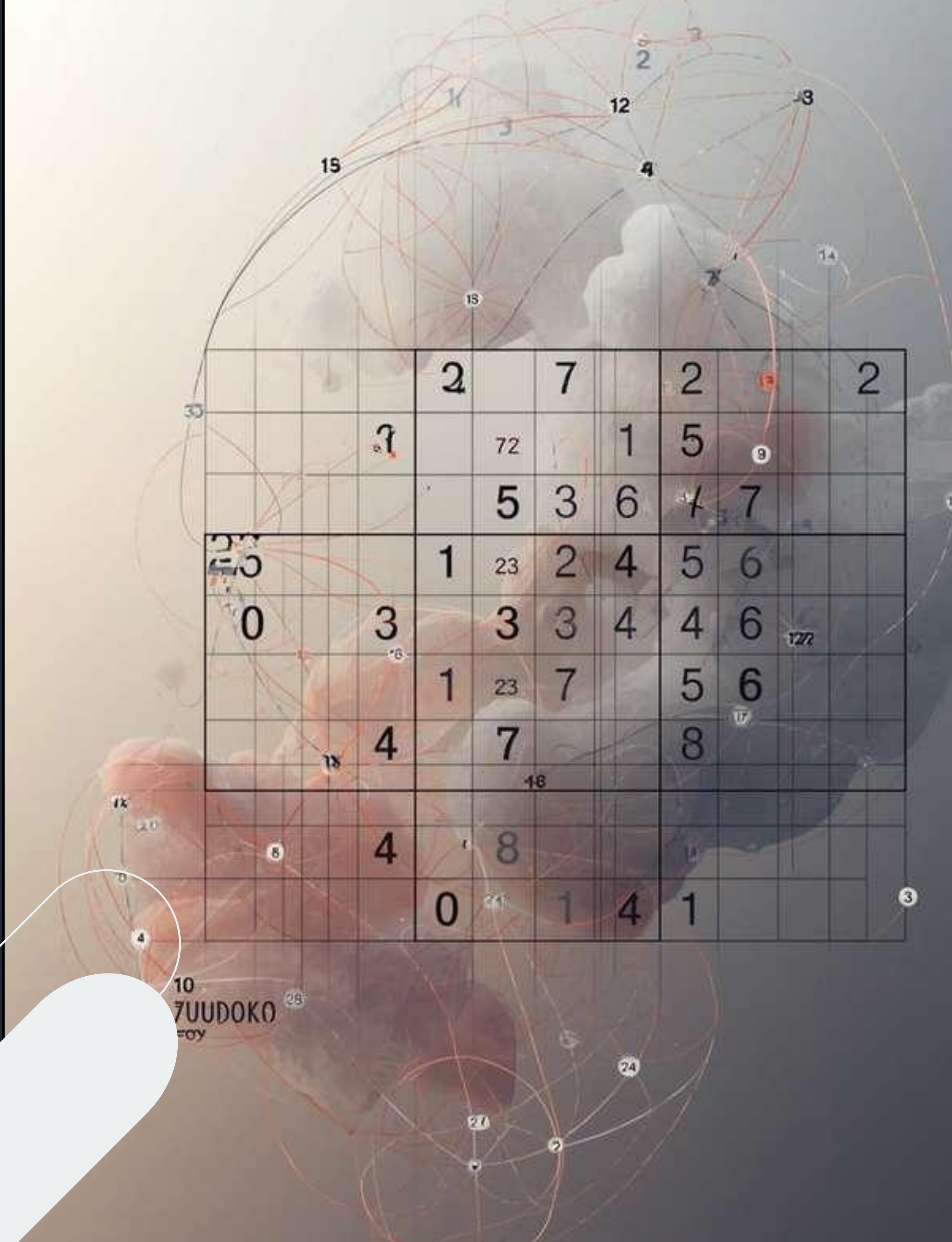
ane sunderrable the comokies pelience the notire addecded

Fape as the balude s of se colow to gaesi

The Computational Challenge of Sudoku Solving

Defining the complexities of automation

Solving Sudoku puzzles automatically involves addressing **multiple constraints** such as ensuring valid placements, maintaining grid integrity, and efficiently exploring potential solutions through algorithmic techniques.



Why Choose Python for Sudoku?

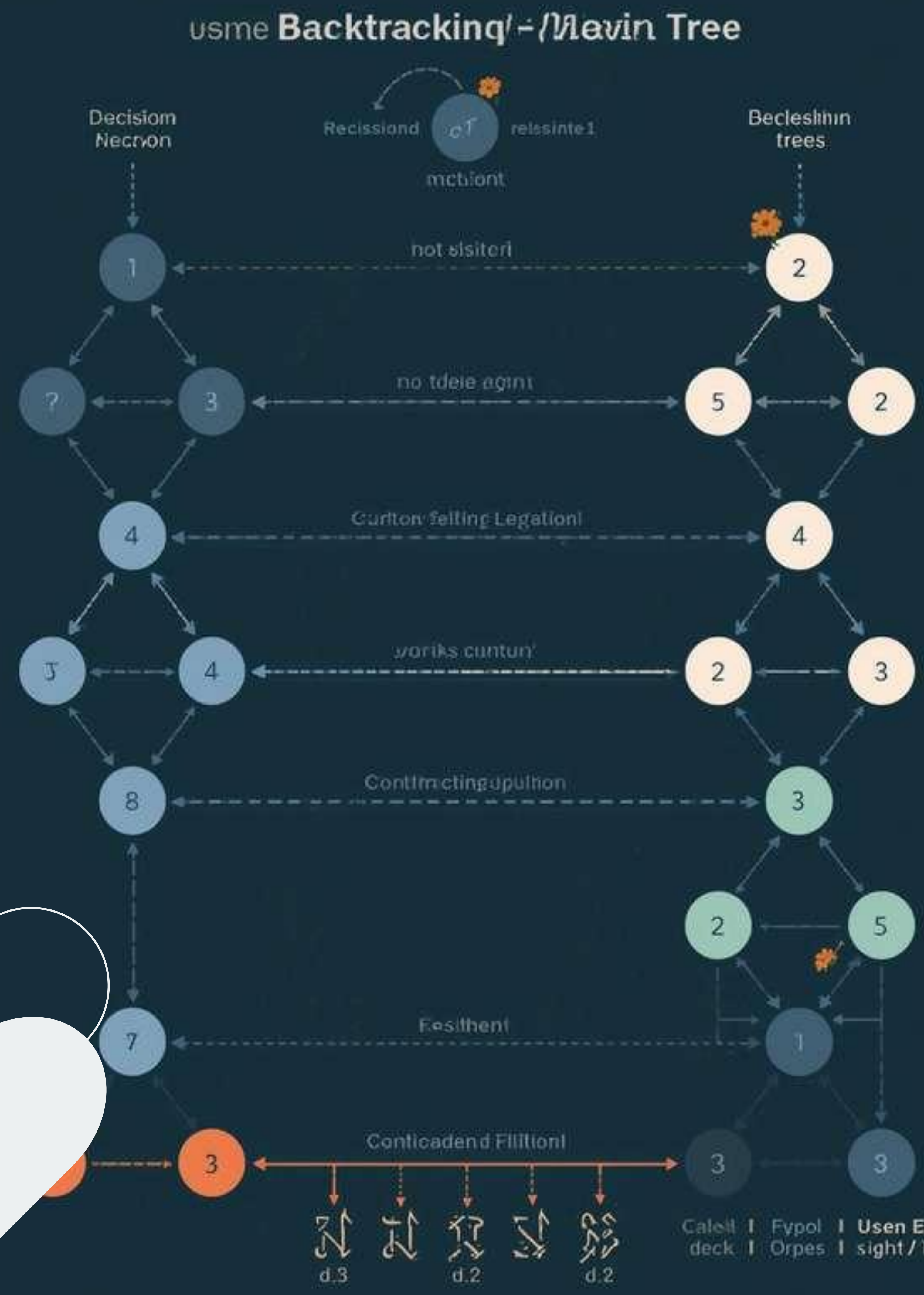
Simplicity and Readability for Students

Python's **simple syntax** and extensive libraries make it an ideal choice for students to learn programming concepts, especially while developing a Sudoku solver and generator.

Understanding the Backtracking Algorithm

Step-by-step logic for Sudoku solving

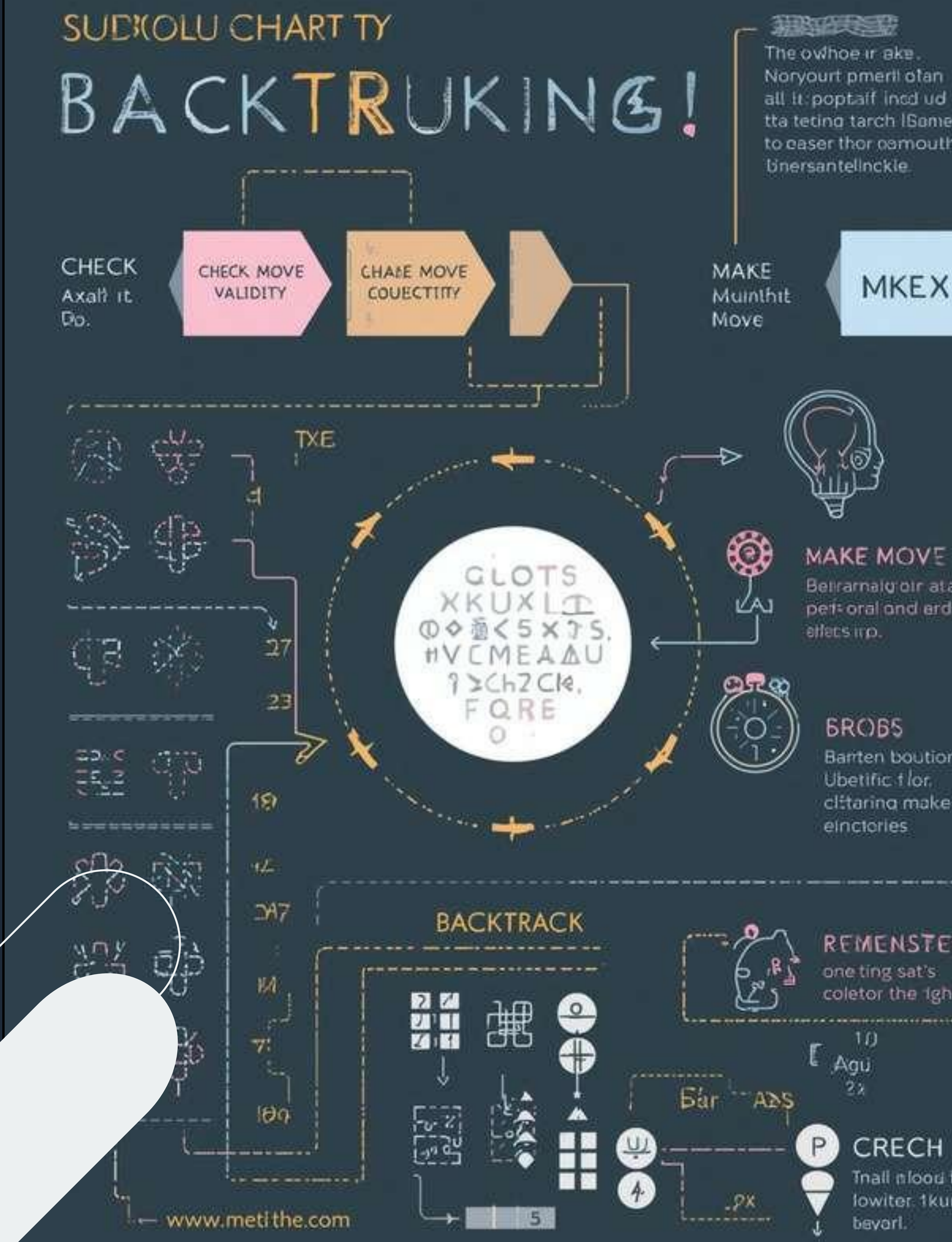
The backtracking algorithm systematically explores possible solutions by testing every option, backtracking when a conflict arises. This method suits Sudoku as it efficiently narrows down valid placements.



Understanding the Backtracking Process

Diagram of Sudoku Solver Logic

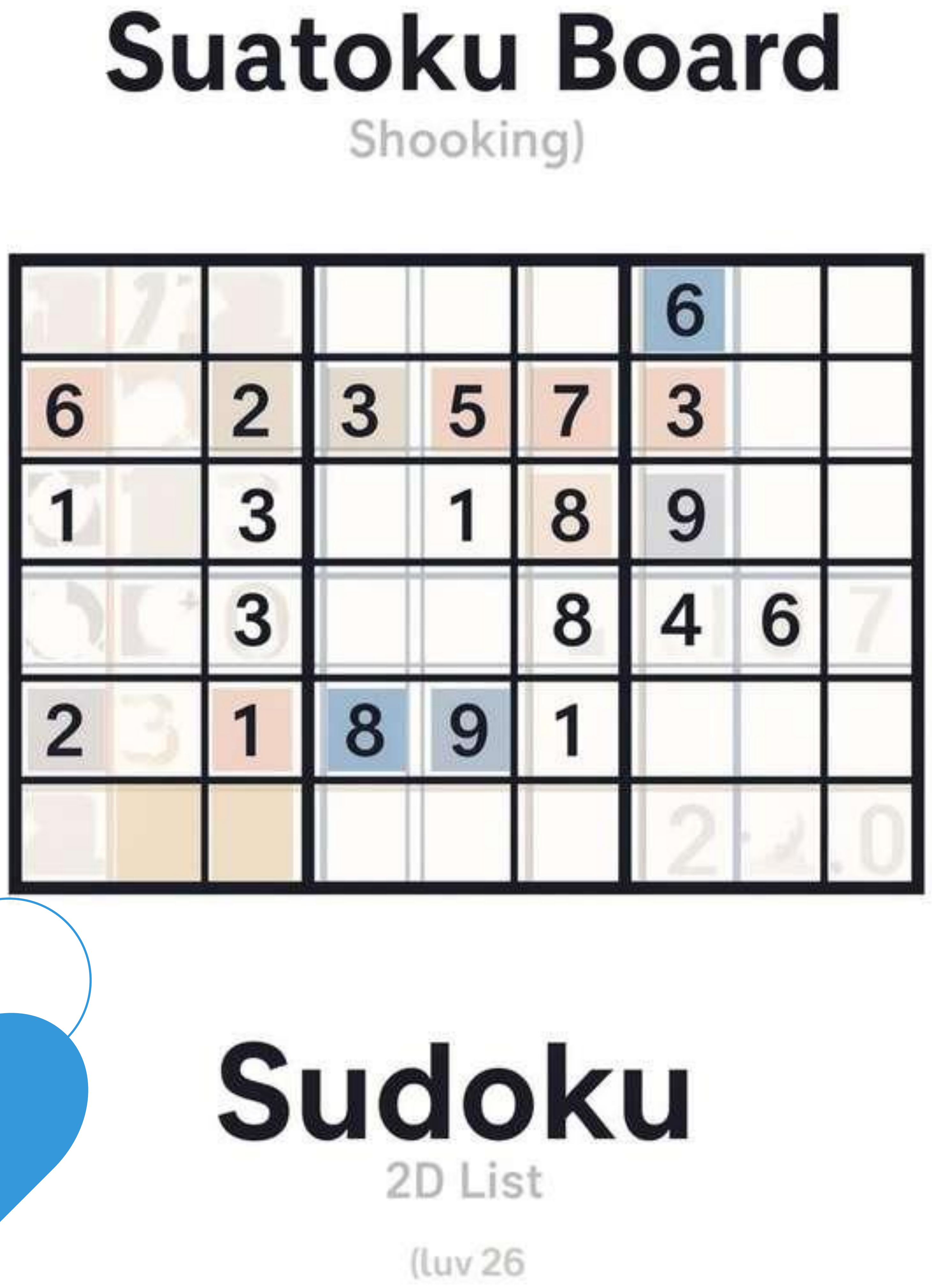
This flowchart illustrates the backtracking algorithm used in the Sudoku solver, demonstrating decision points for move validity checks and recursive calls, showcasing the problem-solving strategy employed.


SUDKOLU CHART TY
BACKTRUKING!
The owhoe ır ake. Noryourt pmeril otan all it poptaif incd ud tta teting tarch IBame to easer thor oamouth Unersantelnckle
CHECK Axalt it Do.
CHECK MOVE VALIDITY
CHAIE MOVE COUECTITY
MAKE Muinthit Move
MKEX
TXE
27
23
19
GLOTS XKUXLI
MAKE MOVE
Benarnalgoir ata pet oral and erd effets irp.
BROBS
Barten boution Ubetific flor. clitaring make einctories
BACKTRACK
REMENSTE
one ting sat's coletor the igh
Agu
Bar
ADS
CRECH
Tnall nlood lowiter 1ku bevorl.
www.metithe.com
5

Grid Representation in Python

Utilizing 2D Lists for Sudoku Boards

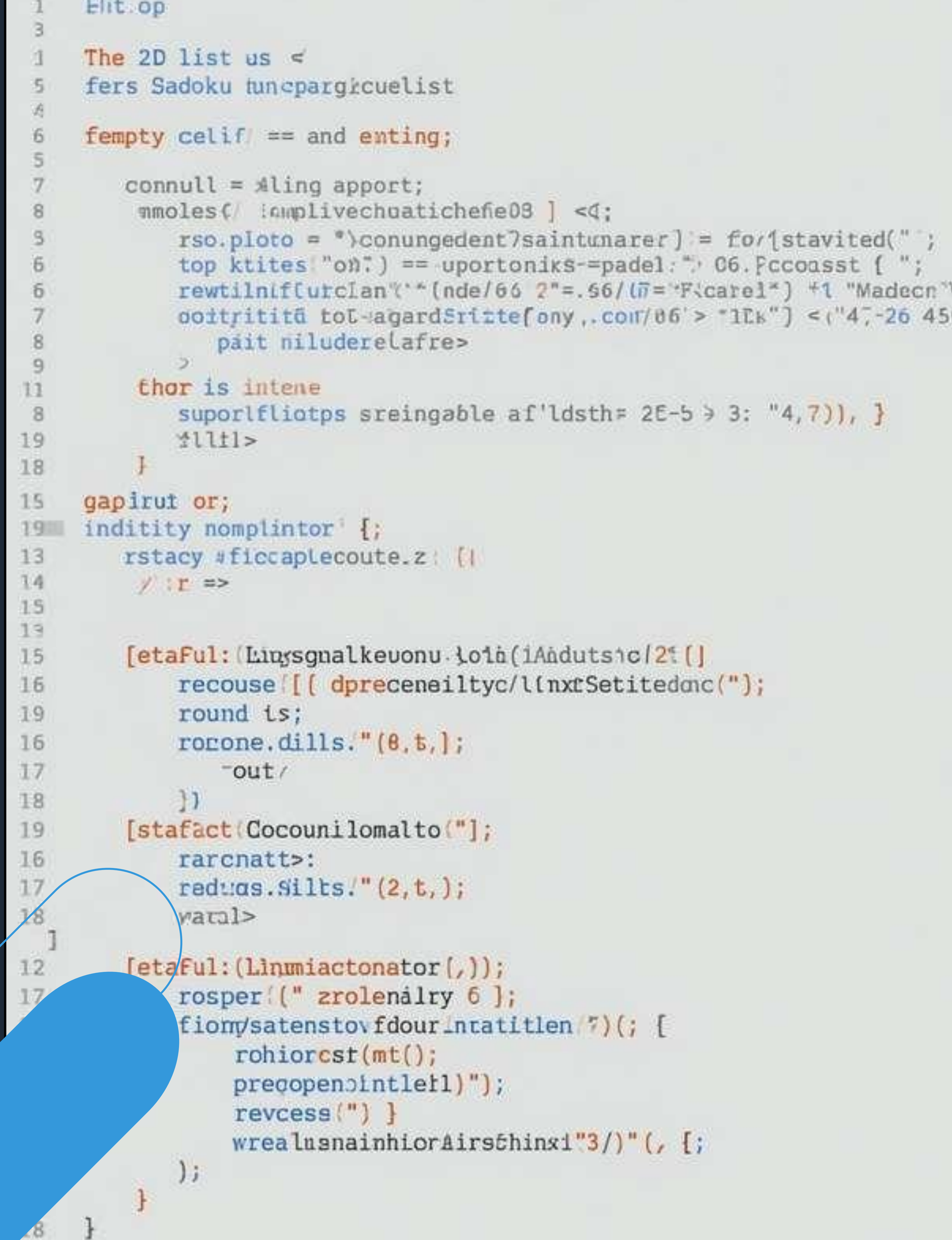
The Sudoku board can be efficiently represented using a **2D list structure** in Python, allowing easy access and manipulation of each cell for solving and generating puzzles.



Python Code for Board Initialization

Creating and setting up the Sudoku grid

This section covers how to create a **Sudoku grid** using a 2D list in Python, initializing both empty and partially filled cells to represent the puzzle's state.

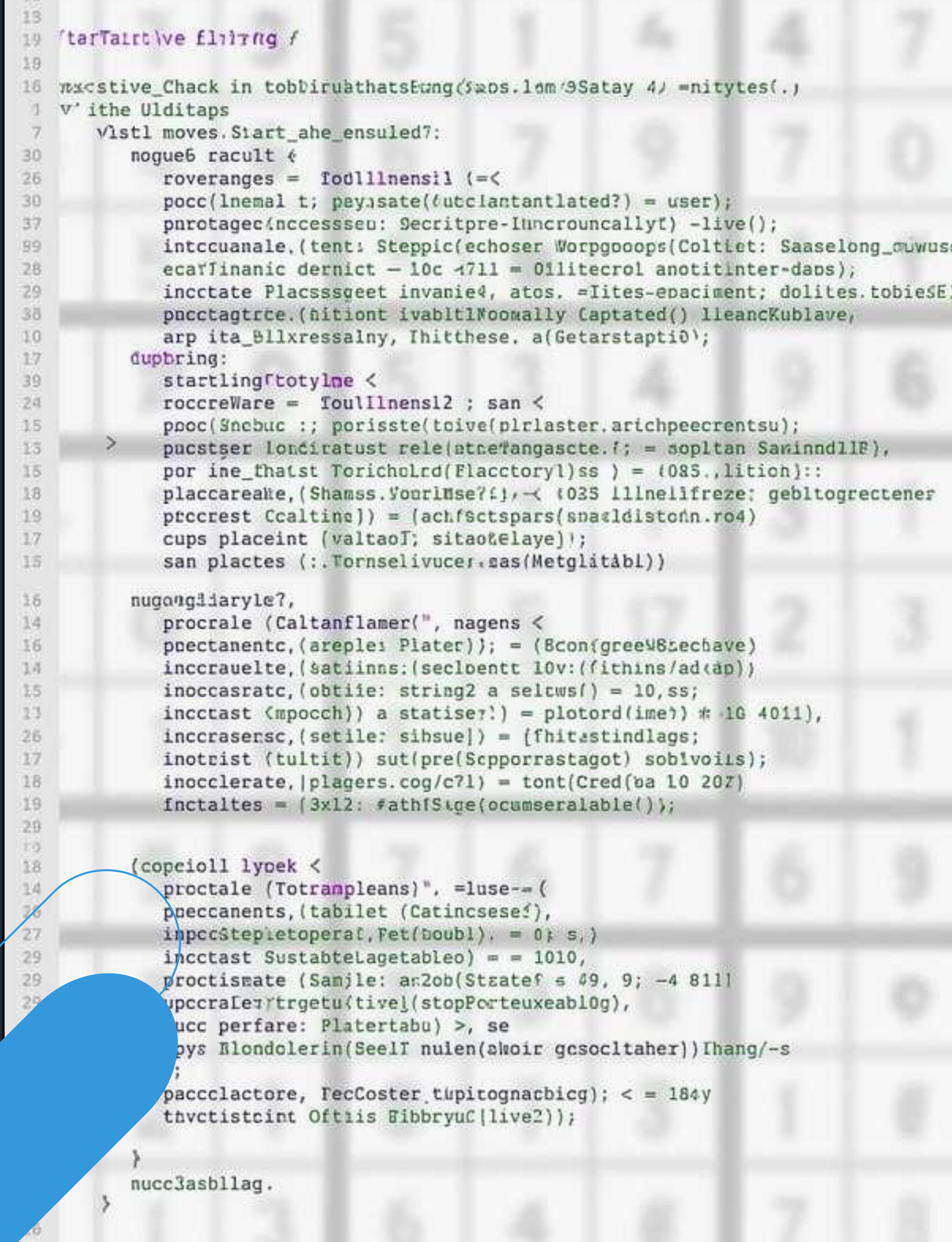


```
Eht.op

The 2D list us <
fers Sadoku tuncpargkcuelist

fempty celif == and enting;

   connull = 4ling apport;
    mmoles(/ !amplivechuatichefie08 ] <4;
       rso.ploto = *)conungedent?saintunarer] = fo/{stavited(" ;
       top ktites "oN:) == uportoniks-=padel:"› 06.Pccoasst { ";
       rewtilnifturclan(`*(nde/66 2"=.56/(ñ='F.carel*) +1 "Madecn
       ooitritiá tot-agardSrizte[ony,.com/06 > "1Ik"] <("4,-26 45
          pait niluderelafre>
       >
    thor is intene
       suporlfliotps sreingable af'ldsth= 2E-5 > 3: "4,7)), }
       tllil>
    }
gapirut or;
inditity nomplintor {;
   rstacy #ficcaplecoute.z {|
    / :r =>


   [etaFul:(Lungsgnalkeuonu.loin(iAndutsic/2t[]
       recouse[[ dpreceneiltyc/l[nxtSetitedanc(");
       round is;
       rorone.dills."(8,t,];
          -out/
       })
   [stafact(Cocounilomalto("];
       rarcnatt>:
       reduas.Silts."(2,t,);
       vatal>
]
   [etaful:(Linmiactonator(,));
       rosper(" zrolenalry 6 ];
       fiom/satenstov fdour.ntatitlen(7)(; [
           rohiorcst(mt();
           preqopenointletl)");
           revcess(") }
           wrealusnainhiorAirsEhinxi"3/)"(, [;
       );
    }
}
```

Validating Moves in Sudoku

Ensuring Correct Placement of Numbers

This section covers the **crucial logic** behind validating number placements in Sudoku, ensuring each move adheres to the rules of row, column, and 3x3 box checks.

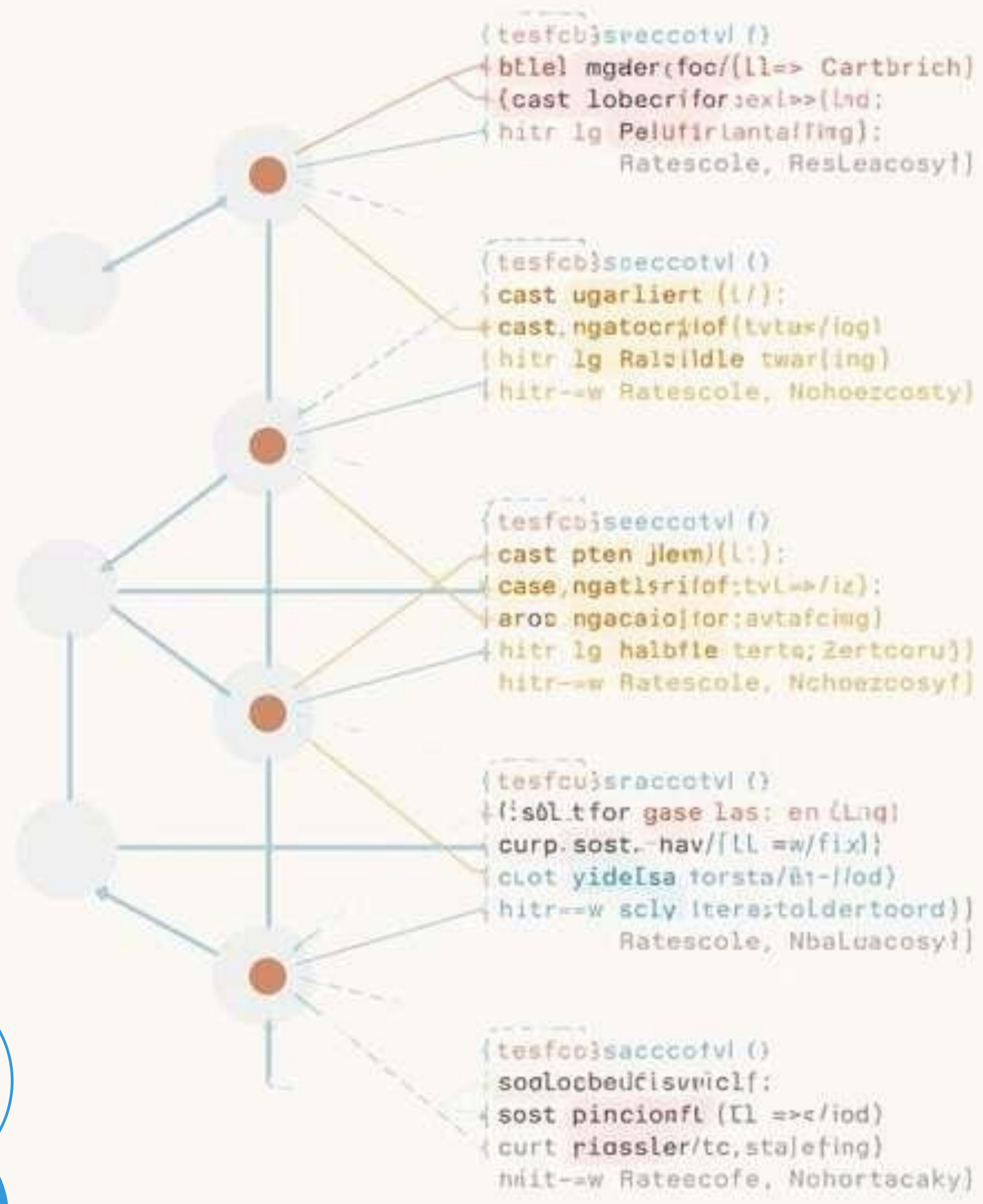


```
'tarTairt\ve flilrrg f

mxcstive_Chack in tobbiruathatsEung(Saos.lom/9Satay 4) =nitytes(.)
v' ithe Ulditaps
  vlstl moves.Start_ahe_ensuled7:
    nogue6 racult {
      roveranges =  Toulllnensil (=<
      pocc(lnemal t; peyısate((utclantantlated?) = user);
      pnrotagec(ncccessseu: 9ecritpre-Iimcrouncallyt) -live();
      intccuanale.(tent: Steppic(echoser Worpgooops(Coltlet: Saaselong_ouwus
      ecaTlinanic dernict — 10c 4711 = 0ilitecrol anotitinter-daps);
      incctate Placsssgeet invanie4, atos. =Iites-epaciment; dolites.tobiesE
      pncctagtrce.(hitiont ivabltlNoomally Captated() lleancKublave,
      arp ita_Bllxressalny, Ihitthese. a(Getarstaptio');
    dupbring:
      startlingrtotylne <
      roccreWare =  Toulllnensl2 ; san <
      pooc(Snebuc :; porisste(toive(plrlaster.arichpeecrentsu);
   >  pucstger lonciratust rele(atcefangascte.f; = sopltan SaninndllF),
      por ine_thatst Toricholrd(Flacctoryl)ss ) = (085.,lition)::
      placcareahe,(Shamss.YourlMse?{j,-< (035 illnelifreze: gebltogrectener
      pcccrest Ccaltine]) = (achfSctspars(snacldistorn.ro4)
      cups placeint (valtaoT; sitaotelaye)';
      san plactes (:.Tornselivucer.sas(Metglitabl))

    nugongiiaryle?,
      procrale (Caltanflamer(", nagens <
      pnectanentc,(areple: Plater)); = (8con{greeV8techave)
      inccrauelte,(satiinns:(secloentt 1ov:((ithins/ad(ap))
      inoccasratc,(obtiie: string2 a selcws() = 10,ss;
      incctast (mpocch)) a statiser:) = plotord(ime)) * 10 4011),
      inccrasersc,(setile: sibsue]) = {fhitastindlags;
      inotrist (tultit)) sut(pre(Scpporrastagot) sob1voiis);
      inocclerate,|plagers.cog/c?l) = tont(Cred(oa 10 207)
      fnctaltes = (3x12: #athfStge(ocumseralable());

    (copcioll lypek <
      proctale (Totrampleans)", =luse-=(
      peccanents,(tabilet (Catincsese!),
      inpccStepletoperat,Fet(toubl). = 0; s,)
      incctast SustabteLagetableo) = = 1010,
      proctismate (Samjle: ar2ob(Steatef = 49, 9; -4 811]
      upccraIerrtrgetu(tivel(stopPorteuxeablog),
      ucc perfare: Platertabu) >, se
      pys Blondolerin(SeelT nulen(aloir gcsocltaher))Ihang/-s
      ;
      paccclactore, TecCoster tupitognacbicg); < = 184y
      thvctistcint Oftiis Bibbryuc(live2));

    }
    nucc3asbllag.
  }
```

Python Code: Recursive Backtracking Logic

Key functions for solving Sudoku puzzles

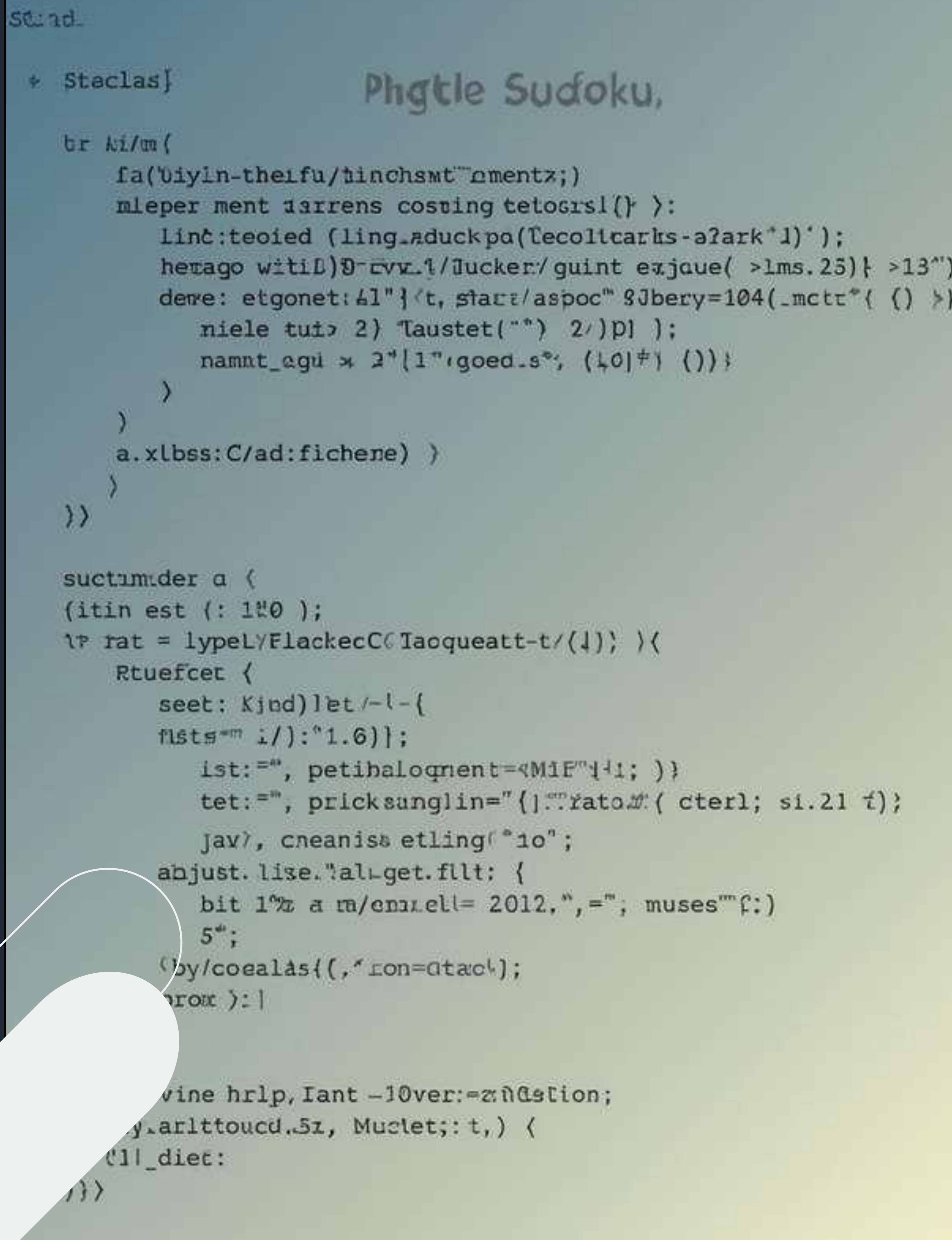
This section illustrates the **main recursive backtracking function** used to solve Sudoku puzzles. It highlights critical decision-making steps and emphasizes the logic employed in the solving process.


(tesfcb}sveccotvl ()
btlel mgder(foc/([Ll=> Cartbrich]
(cast lobecrifor:exl=>(Lnd:
hitr lg Pelufirlantalfing):
Ratescole, ResLeacosy!]
(tesfcb}sceccotvl ()
cast ugarliert (l/):
cast, ngatocrilof(tvtax/iog]
hitr lg Ralzildle twar(ing)
hitr-=w Ratescole, Nohoezcosty]
(tesfcb}seeccotvl ()
cast pten jlem)(L:):
case, ngatlsrilof:tvL=>/iz):
aroc ngacaio|for:avtafcimg]
hitr lg halbfle terte; Zertcoru})
hitr-=w Ratescole, Nchoezcosy!]
(tesfcu}sraccotvl ()
(!sôL.tfor gase Las: en (Lnq]
curp.sost. hav/[LL =w/fix]}
cLot yideLsa torsta/ê1-|/od)
hitr==w scly Itera:toldertoord})
Ratescole, NbaLuacosy!]
(tesfco}sacccotvl ()
soolocbeufisvwiclf:
sost pincionft (Tl =>c/iod)
curt riossler/tc,stalefing)
hmit-=w Rateecofe, Nohortacaky]

Python Code: Backtracking Logic Continued

Deep dive into the recursive function

This section elaborates on the **recursive backtracking structure** of the Sudoku solver. It demonstrates how the algorithm progresses through each potential solution until the puzzle is solved.

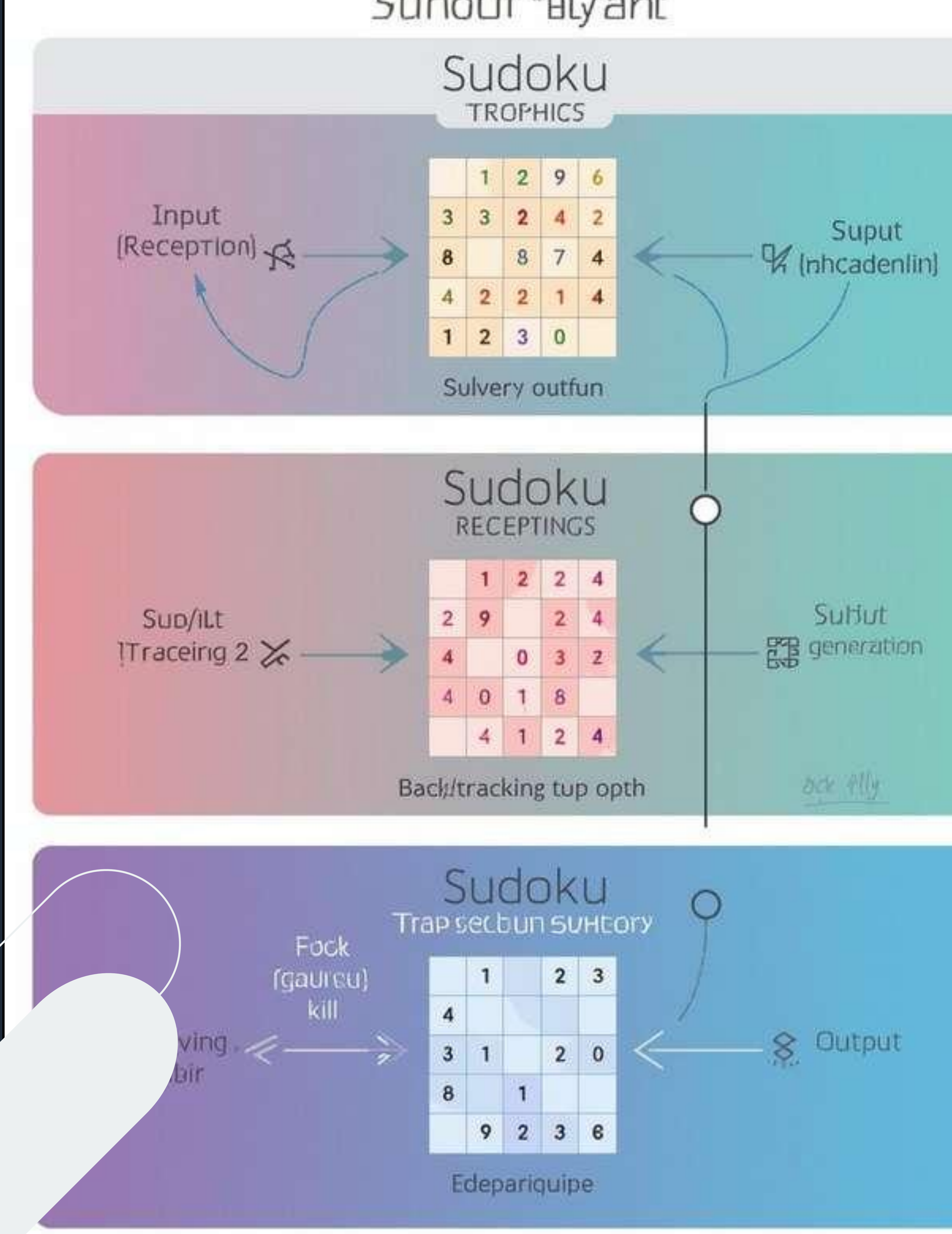


```
SGːad.

 ✦ Staclas]                 Phgtle Sudoku,

   br ki/m(
       fa('biyin-theıfu/binchsmt"ɷmentz;)
       mleper ment ɜɜrrens costing tetocrsl(}} ):
          Linĉ:teoied (ling.nduck pa(Tecoltcarks-a?ark'1)');
          hemago witiL)D-cvv.1/Jucker/guint exjaue( >lms.25)} >13")
          deme: etgonet:61"}'t, start/aspoc" 8Jbery=104(_mctt"( () >)
             niele tui> 2} Taustet("") 2/)p] );
             namnt_agu » 2"[1"'goed.s"; (lo|#) ())}
          )
       )
       a.xlbss:C/ad:fichene) }
       )
   })

   suctımːder a (
   (itin est (: 1ʺ0 );
   \P rat = lypeL/FlackecC(Iaoqueatt-t/(ɺ)} )(
       Rtuefcet (
          seet: Kjod)]et/-l-{
          flsts=" i/):"1.6)];
             ist:=", petihaloqnent=<MIF"{ʲ1; )}
             tet:=", pricksunglin="{|:"rato𝑧( cterl; si.21 t)}
             jav/, cneanisa etling("1o";
          abjust. lise.'al-get.fllt: {
             bit 1%z a m/enxcell= 2012.",="; muses"f:)
             5";
          'by/coealas{(,' ron=atac'];
          prox ):|

          vine hrlp,Iant -10ver:=zn0stion;
       y.arlttoucd.5z, Muclet;: t,) (
       C1|_diet:
   /})
```

Understanding Program Execution Flow

Input, Solving Process, and Output Explained

The program begins by receiving a Sudoku puzzle as input, processes the solving logic using the backtracking algorithm, and finally outputs the solved Sudoku grid for verification.



	1	2	9	6
3	3	2	4	2
8		8	7	4
4	2	2	1	4
1	2	3	0	

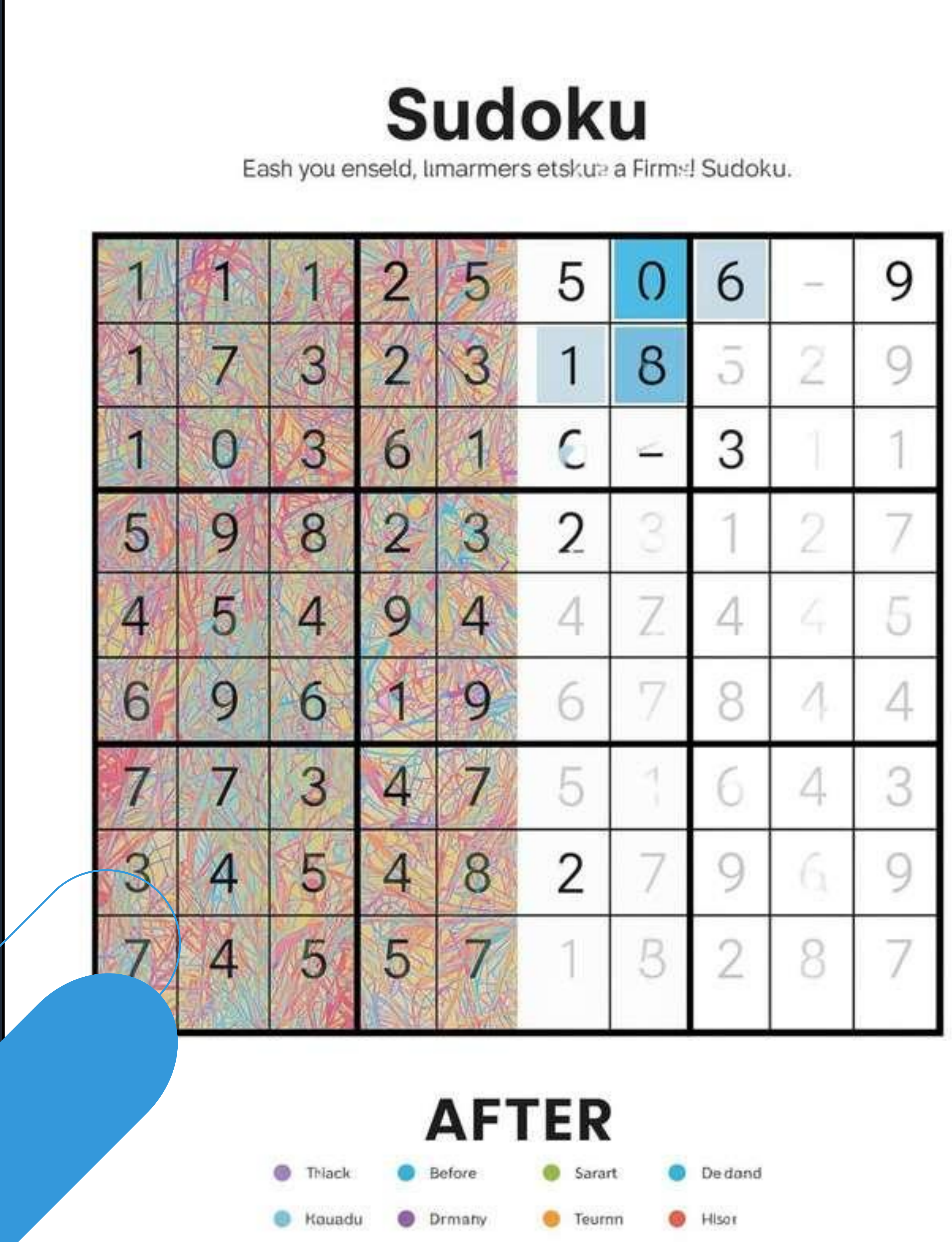
	1	2	2	4
2	9		2	4
4		0	3	2
4	0	1	8	
	4	1	2	4

	1		2	3
4				
3	1		2	0
8		1		
	9	2	3	6

Transforming Sudoku Grids: Before and After

Visualizing the Solution Process in Action

The images showcase the **transformation of complex Sudoku puzzles** into solved grids, illustrating the effectiveness of our automated solver and highlighting the process's efficiency and accuracy.



Sudoku

Eash you enseld, lımarmers etskuᴚ a Firm:! Sudoku.

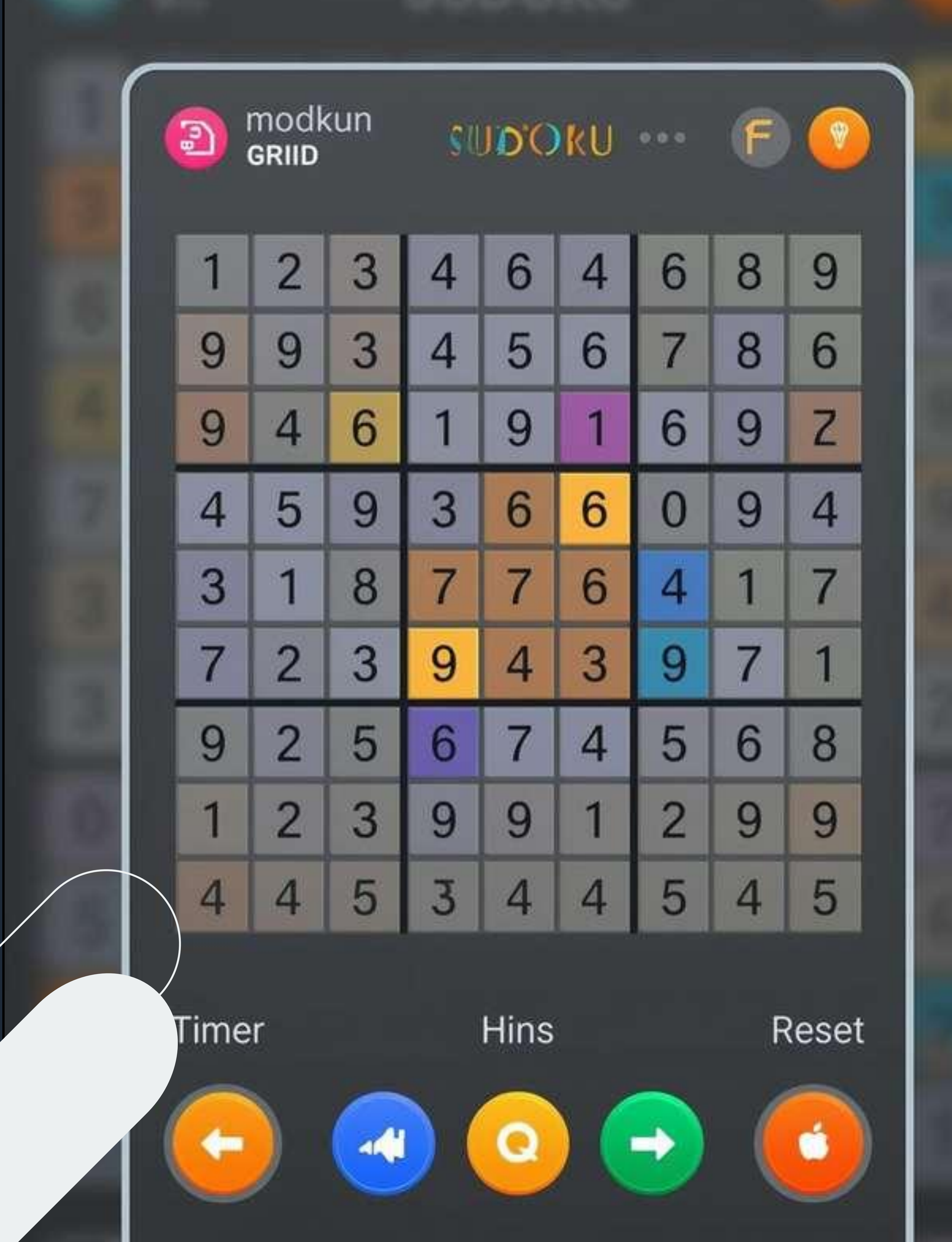
1	1	1	2	5	5	0	6	–	9
1	7	3	2	3	1	8	5	2	9
1	0	3	6	1	C	–	3	1	1
5	9	8	2	3	2	3	1	2	7
4	5	4	9	4	4	Z	4	4	5
6	9	6	1	9	6	7	8	4	4
7	7	3	4	7	5	1	6	4	3
3	4	5	4	8	2	7	9	6	9
7	4	5	5	7	1	8	2	8	7

AFTER

Future Enhancements for Sudoku Solver

Exploring GUI and AI Optimization Techniques

Future enhancements may include **developing a user-friendly GUI**, integrating heuristic algorithms for faster solving, and exploring AI-based techniques to improve problem-solving efficiency in Sudoku puzzles.



1	2	3	4	6	4	6	8	9
9	9	3	4	5	6	7	8	6
9	4	6	1	9	1	6	9	Z
4	5	9	3	6	6	0	9	4
3	1	8	7	7	6	4	1	7
7	2	3	9	4	3	9	7	1
9	2	5	6	7	4	5	6	8
1	2	3	9	9	1	2	9	9
4	4	5	3	4	4	5	4	5

Thank youuuu!!!!!