Derek Anderson
Jeremy Wright

Project Sudoku

Our project was to create an interactive Sudoku game so that you can play it on the computer instead of in the newspaper. We wanted to make it so that you could change the difficulty of the game based on your own skill level. We also incorporated an error counter that tell you how mistakes you have made if any. And it will tell you if your Sudoku is still solvable if you type in a correct answer.

This was a fun project to do because we were excited to see our outcome. Jeremy and I play Sudoku every day in the newspaper and we thought why not make it easier to do instead of erasing each time we make a mistake.

The first thing we did was make sage create a random 9x9 matrix with integers 1-9. Then we made it so it would fill in a certain number of boxes up to all 81 boxes. Then we need to figure out how to make so when it created our random matrix, it wouldn't put the same number in the same row, column, or 3x3 box. Lastly, we needed to make sure that it was still a solvable Sudoku and the solution wouldn't have any repeats in a row, column, or 3x3 boxes.

The next thing we did was make it so it would produce a solution to the randomly generated Sudoku. This way, people can look at the answer to see if they did it correctly or if they are making good progress. We had some help from the Sudoku program that is already in sage.

The next thing we did was add in the error counter so that it tells you how many mistakes you made. Who can do “El Escargot” without any mistakes? We also added in a slider so that you can change the difficulty without any problems or having to go back and change the code.

Here is an explanation of our code:

```
#This piece of code creates the slider that changes the difficulty setting ranging from 17 to 81. 17 is the lowest number of boxes that can be filled that has a unique solution. Anything below that will have more than one solution. And 81 will give you a complete solution.

global_starting_number = 30

@interact
def _(starting_number=(30,(17..81))):
    global global_starting_number
    global_starting_number = starting_number
    print "When you create a Sudoku below, the starting number will be %s"%global_starting_number
    html("<script>evaluate_cell(%s,0)</script>"%sudoku_cell_id)


#This is the entire code for our interactive Sudoku.


%hide
%auto

sudoku_cell_id = sagenb.notebook.interact.SAGE_CELL_ID


import math


# This checks the row and column and makes it so that when a Sudoku is generated, it won’t put a repeat in the row or column.


def check_row(x,y,z):
    j=0
```

```
    while j<9:
        if a[j,y] == z or a[x,j] == z:
            return false
        else:
            j=j+1
    return true
```

This line of code makes it so that it doesn't put a repeat in the same 3x3 box.

```
def check_box(x,y,z):
    x1=int(math.floor(x/3)*3)
    y1=int(math.floor(y/3)*3)
    for l in range(y1, y1+3):
        for k in range(x1, x1+3):
            if a[k,l]==z:
                return false
    return true
```

#This generates the random Sudoku with number 1-9 and makes it a solvable Sudoku.

```
a = copy(zero_matrix(ZZ,9))
i=0
while i in range(global_starting_number):
    x=ZZ.random_element(0,9)
    y=ZZ.random_element(0,9)
    z=ZZ.random_element(1,10)
    while a[x,y]!=0:
        x=ZZ.random_element(0,9)
        y=ZZ.random_element(0,9)
    while not(check_row(x,y,z) and check_box(x,y,z)):
        z=ZZ.random_element(1,10)
    a[x,y]=z
    i=i+1
    try:
        sage.all.sudoku(a)
    except:
        a[x,y]=0
        i=i-1
```

#This makes it an interactive Sudoku and will print the number of errors you made, it will tell you if it's still solvable, and it will tell you if you have made an error.

```
errors = 0
@interact
def f(sudoku=a):
    global errors
    try:
        sage.all.sudoku(sudoku)
    except:
        errors += 1
        print "ERROR!"
        print "Errors: %s"%errors
        for x in range(30):
            print "."
        print "Solution!"
    else:
        j = 0
        for x in range(0,9):
            for y in range(0,9):
                if sudoku[x,y] == 0:
                    break
                else:
                    j = j + 1
        if (j == 81):
            print "GOOD JOB!!"
        else:
            print "still solvable..."
            print "Errors: %s"%errors
            for x in range(30):
                print "."
            print "Solution!"


#This prints the solution using the Sudoku code already in sage.


s = sudoku(a)
show(s)
```