# Coding 4Ward

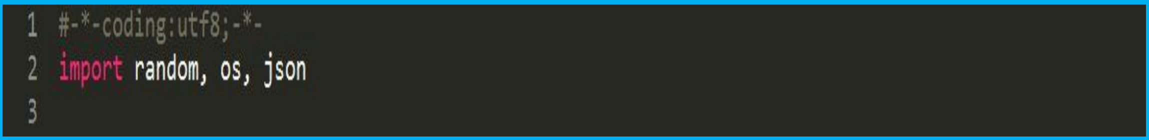
Sudoku Solver

Jose M.Sixpenze

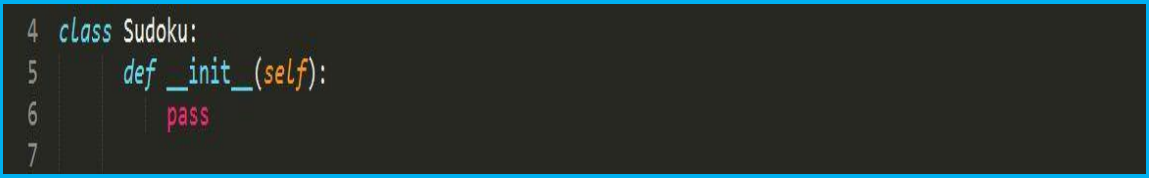
# {CODING 4WARD}

## 1.Sudoku Solver:

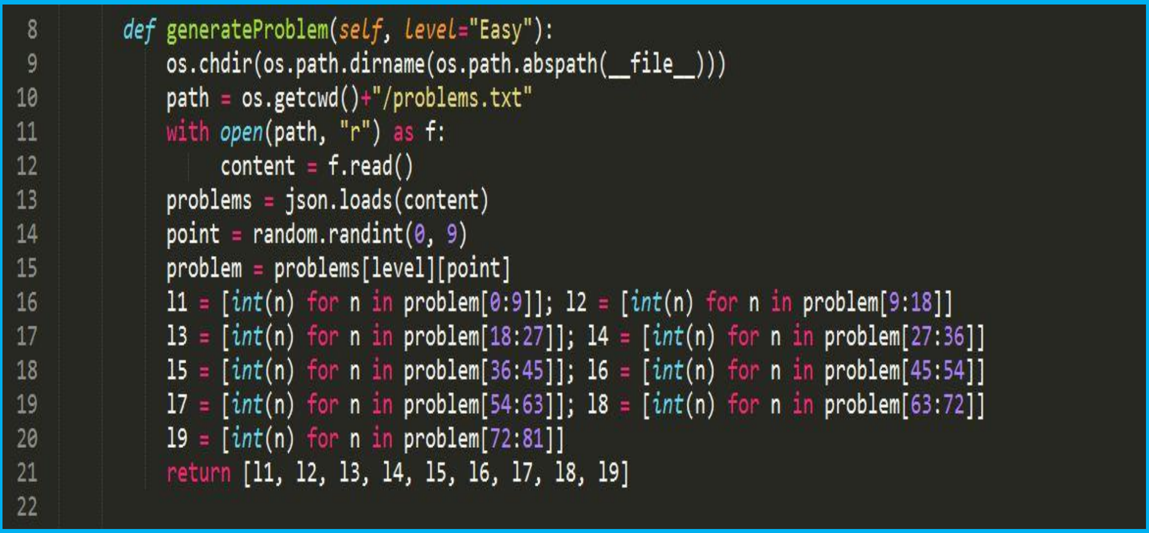
**Sudoku.py**



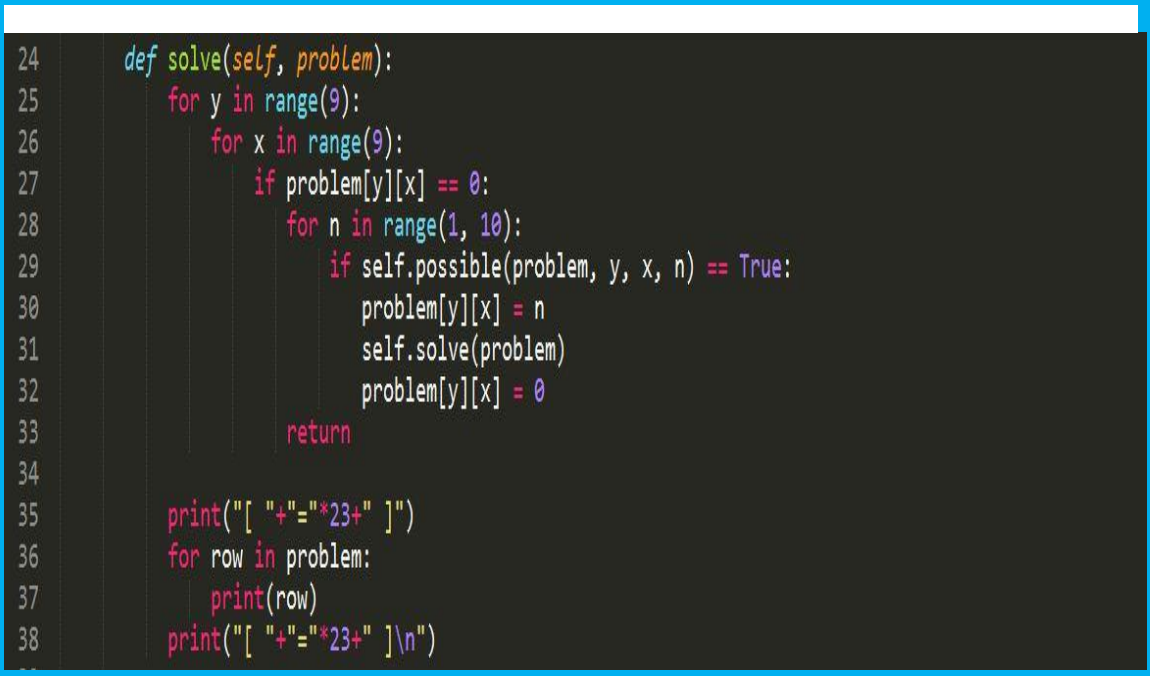
```python
#-*-coding:utf8;-*-
import random, os, json

```



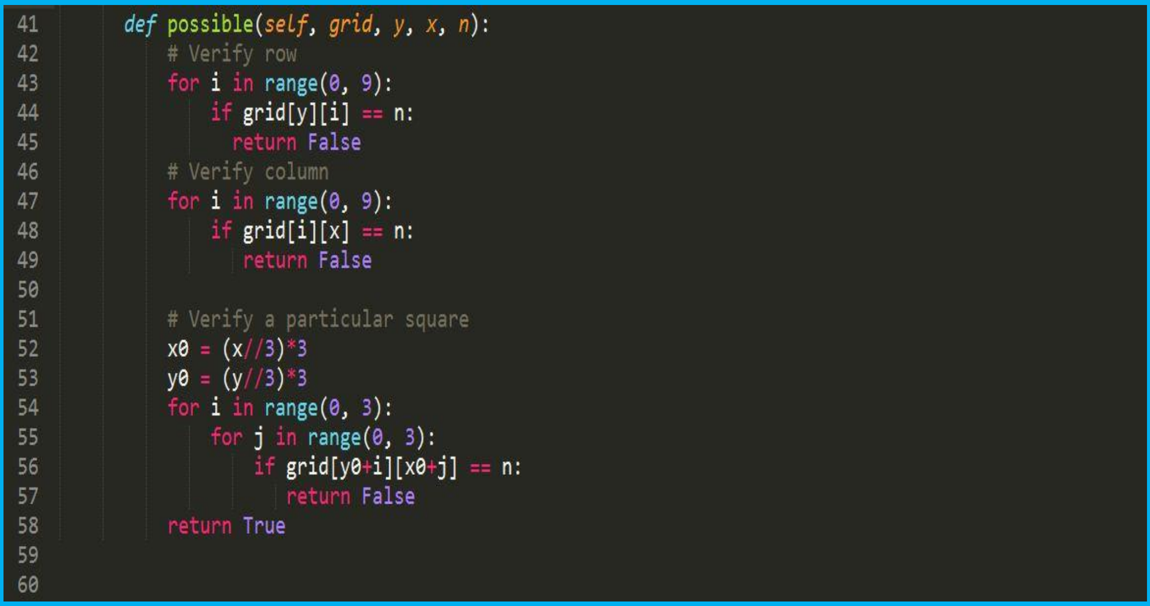
```python
class Sudoku:
    def __init__(self):
        pass

```



```python
    def generateProblem(self, level="Easy"):
        os.chdir(os.path.dirname(os.path.abspath(__file__)))
        path = os.getcwd()+"/problems.txt"
        with open(path, "r") as f:
            content = f.read()
        problems = json.loads(content)
        point = random.randint(0, 9)
        problem = problems[level][point]
        l1 = [int(n) for n in problem[0:9]]; l2 = [int(n) for n in problem[9:18]]
        l3 = [int(n) for n in problem[18:27]]; l4 = [int(n) for n in problem[27:36]]
        l5 = [int(n) for n in problem[36:45]]; l6 = [int(n) for n in problem[45:54]]
        l7 = [int(n) for n in problem[54:63]]; l8 = [int(n) for n in problem[63:72]]
        l9 = [int(n) for n in problem[72:81]]
        return [l1, l2, l3, l4, l5, l6, l7, l8, l9]

```



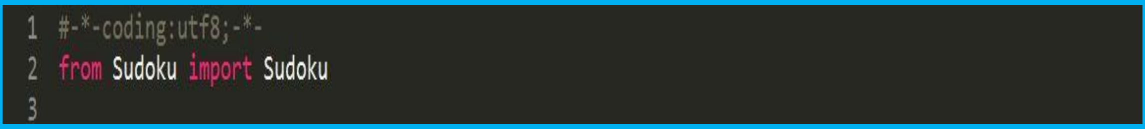
```
    def solve(self, problem):
        for y in range(9):
            for x in range(9):
                if problem[y][x] == 0:
                    for n in range(1, 10):
                        if self.possible(problem, y, x, n) == True:
                            problem[y][x] = n
                            self.solve(problem)
                            problem[y][x] = 0
                    return

        print("[ "+"="*23+" ]")
        for row in problem:
            print(row)
        print("[ "+"="*23+" ]\n")
```



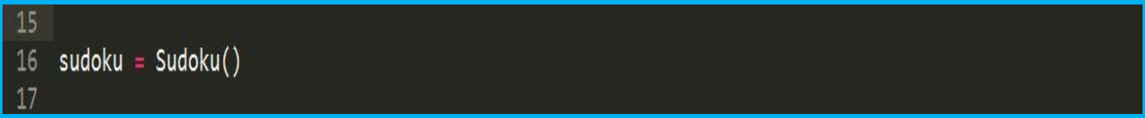
```
    def possible(self, grid, y, x, n):
        # Verify row
        for i in range(0, 9):
            if grid[y][i] == n:
                return False
        # Verify column
        for i in range(0, 9):
            if grid[i][x] == n:
                return False

        # Verify a particular square
        x0 = (x//3)*3
        y0 = (y//3)*3
        for i in range(0, 3):
            for j in range(0, 3):
                if grid[y0+i][x0+j] == n:
                    return False
        return True


```

## 2. HOW TO USE THE SUDOKU SOLVER?

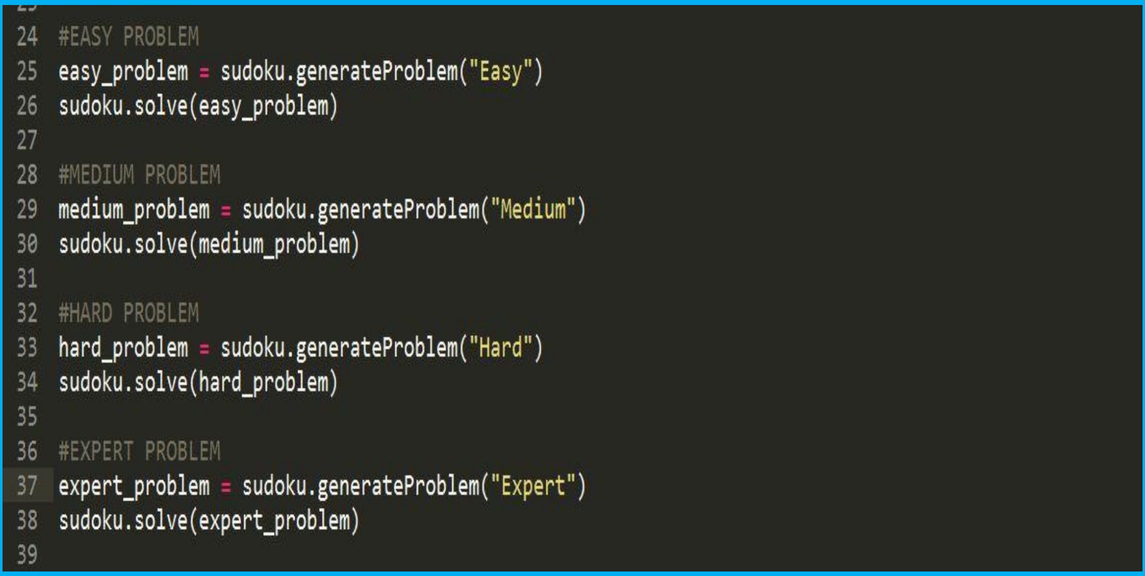
### main.py



```python
#-*-coding:utf8;-*-
from Sudoku import Sudoku

```



```python

sudoku = Sudoku()

```



```python
#EASY PROBLEM
easy_problem = sudoku.generateProblem("Easy")
sudoku.solve(easy_problem)

#MEDIUM PROBLEM
medium_problem = sudoku.generateProblem("Medium")
sudoku.solve(medium_problem)

#HARD PROBLEM
hard_problem = sudoku.generateProblem("Hard")
sudoku.solve(hard_problem)

#EXPERT PROBLEM
expert_problem = sudoku.generateProblem("Expert")
sudoku.solve(expert_problem)

```