




CS506 Lecture

<input checked="" type="checkbox"/> Override filters	<input type="checkbox"/>
 Room	KCB 104

Notes: Code better

"Software systems get replaced not when they wear out but when they crumble under their own weight because they have become too complex"

Setting yourself up for success:

- Try to have a plan and prioritize according to your plan
- Structure
 - When you declare a function make sure it does that one thing it tries to do
 - Compartmentalize: Make sections of files, and different files to portion everything
 - Many Functions with small bodies > one function with a large body

Top-Down Approach:

- You write almost what is pseudocode about you want to program
 - Is nice with typed languages
 - bugs can arise that you may have not thought of
 - You write the body of your code disregard the functions until your program is done and go back and implement the functions

Bottom-up:

- You implement the functions first then work on the program
 - Everything you need will already be implemented by the time you work on your main program
 - You may overdo it and create functions you don't need

Debugging:

- Don't look for a quick fix
- Read the error
 - What is the error saying
 - remember one bug can hide another
 - Re-read your code
 - Chances are that your code maybe to complex to be able to find the error and you may need to simplify it
- Is everything set up properly, like APIs and stuc
- Look for help Office hours etc.

Live Coding:

```
class Board:

    def __init__(self):
        self.board = [[" " for _ in range*8] for _ in range(8)]

    def __repr__(self):
        res= ''
        for row in range(8):
            for col in range(8):
```

```

        res+= self.board[row][col]
        res+=" "
        res+="\n"

    return res

def set_queen_at(self, row, col):
    self.board[row][col]= "Q"

def unset_queen_on(self, row):
    self.board[row]= ["-" for _ in range(8)]

def is_valid_col(self, row, col):
    for i in range(8):
        if i != row and self.board[i][col]== "Q":
            return False

    return True

def is_valid_row(self, row, col):
    for j in range(8):
        if j != col and self.board[row][j]== "Q":
            return False

    return True

def is_valid_move(self, row, col):
    if not self.is_valid_row(self, row, col):
        return False

    if not self.is_valid_col(row, col):
        return False

```

```

        return True

def get_queen_on_row(self, row):
    for i in range(8):
        if self.board[row][i] == 'Q':
            return i
    raise ValueError("No queen on this row")

def find_solution(self):
    row = 0
    col = 0

    while row < 8:
        if self.is_valid_move(row, col):
            self.set_queen_at(row, col)
            row += 1
            col = 0

    else:
        col += 1
        if col >= 8:
            col = self.get_queen_on_row(row - 1)
            col += 1
            row -= 1

```