

## Lecture: 20-1

- Prerequisites for this lecture are: 19-1, 19-2 and 19-3.



## File export/File import/Next/Back

- improving the export function and coordinating the result of an unsuccessful export with the doom button
  - if the export should fail, then the doom3 button should be frozen

```
def save_map (name):  
    f = open (name, "w")  
    f = write_assets (f)  
    f.write ("\n") # add blank line for eye candy  
    f = write_map (f)  
    f.close ()  
  
def myexport (name, tap):  
    pygame.display.update ()  
    save_map (current_map_name)  
    check_export ()
```

## check\_export



```
def check_export ():  
    etc
```

## write\_map

- there is currently a problem with `write_map` and `chisel`
  - if the map has leading spaces `chisel` will fail
  - if the map has trailing spaces then `chisel` will also fail
  
- while this is a bug in `chisel`
  - we can avoid it by trimming spaces from our file in `write_map`
  - this is common practice - in software engineering

## write\_map

```
def write_map (f):  
    left, right = determine_range ()  
    m = ""  
    mdict = {"v":"#", "h":"#", "-":".", "|":".", " ":" ",  
            "H":"H", "S":"S", "T":"T"}  
    x, y = cell_array.high ()  
    for j in range (y):  
        for i in range (left, right+1):  
            if mdict.has_key (cell_array.get (i, j)):  
                m += mdict[cell_array.get (i, j)]  
            else:  
                m += cell_array.get (i, j)  
        # skip blank lines  
        m = m.rstrip ()  
        if len (m) > 0:  
            m += "\n"  
    f.write (m)  
    return f
```

## determine\_range

```
def determine_range ():
    left = -1
    x, y = cell_array.high ()
    right = x
    for j in range (y):
        for i in range (x):
            if cell_array.get (i, j) != " ":
                if (left == -1) or (i < left):
                    left = i
                if i > right:
                    right = i
    return left, right
```