**R Grader Map**

**Init()**:

one\_time\_setup()

setup\_for\_new\_dir()

init\_gui()

update\_gui(True, True)

**one\_time\_setup()**:

global\_general\_config = construct\_config()

global\_general\_config =

update\_config\_from\_file(...)

general\_config = None

global\_specific\_config = construct\_config()

global\_specific\_config =

update\_config\_from\_file(...)

specific\_configs = None

cf\_index = filename\_separator = roster\_id =

None

**setup\_for\_new\_dir()**:

dir = os.getcwd()

general\_config = global\_general\_config

update\_general\_config()

codefiles = get\_codefiles()

if self.codefiles is None:

current\_codefile = None

specific\_config = None

return

current.codefile = 0

[drop excess codefiles]

set\_specific\_configs()

get\_codefile\_files()

**update\_general\_config():**

general\_config = global\_general\_config

general\_config = update\_config\_from\_file()

set\_file\_format\_info()

read\_roster()

**set\_file\_format\_info():**

set filename\_separator, file\_format\_items[]

**read\_roster():**

set roster\_firstname, roster\_lastname,

roster\_email, roster\_fullname

**get\_codefiles():**

file\_list from general\_config['codefiles']

expand '\*' if needed, updating config file

and general\_config

return None or codefile list

**set\_specific\_configs():**

specific\_configs = {}

for each codefile:

specific\_config\_inner =

global\_specific\_config

specific\_config\_inner =

update\_config\_from\_file()

specific\_configs[codefile] =

specific\_config\_inner

**construct\_config(setup\_tuple):**

**write\_config\_file(config\_dict):**

**parse\_one\_filename(name):**

**init\_gui():**

Set menubar, cf\_index, cf\_radio\_dict,

cf\_text\_dict

Callbacks: path\_select,

general\_config\_dialog,

specific\_config\_dialog, on\_quit,

set\_specific, run\_one

**update\_gui(new\_codefiles, new\_codefile\_files):**

if new\_codefiles:

update\_codefiles\_in\_gui()

if new\_codefile\_files:

update\_codefile\_rfiles\_in\_gui()

**update\_codefiles\_in\_gui():**

Fix codefile radio buttons

cf\_index.set(0) (or None for no codefiles)

Fix menu item for specific codefile

configuration

**update\_codefile\_files\_in\_gui():**

reset dropdownMenu of file ids

reset chosen\_file

reset file\_count

**general\_config\_dialog():**

save old values as 'gconfig'

gconfig = copy(general\_config)

ConfigDialog(gconfig)

general\_config = conf\_dialog\_result

update\_for\_general\_config\_non\_gui(new...)

write\_config\_file(general\_config)

update\_gui(...)

**update\_for\_general\_config\_non\_gui**():

new\_file\_format, new\_code\_files,

new\_course\_id)

set\_file\_format()

codefiles = get\_codefiles()

general.config['codefiles'] = ", ".join(...)

get\_f\_files()

read\_roster()

**parse\_codefile\_names(names):**

set fullname, filename, version, timestamp,

student\_name, email, filelabel for each

unique student (using parse\_one\_filename)

**get\_student\_files():**

codefile = filenames matching current

general\_config['codefiles']

rtn = files with correct names

parse\_codefile\_names(rtn)

**path\_select():**

dir = filedialog.askdirectory()

new\_dir(dir)

**new\_dir(dir):**

set dir, change dir, and set GUI title

setup\_for\_new\_dir()

update\_gui(True, True)

**f\_path(fname, add\_dir=False):**

return list of filenames

**pre\_analyze(text):**

...

**post\_analyze(file):**

...

**submit\_code():**

...

**run\_one():**

file = f\_path(chosen\_file)

text = myfile.read()

pre\_analyze(text)

submit\_code(file)

post\_analyze(file)