**CSE1100 – Intro to Programming Concepts with Python**

**Final Project Report**

**Fall 2021**

Cael Shoop

Computer Science, B.S.

cshoop2018@my.fit.edu

11/5/21

# 1. Background & Problem Statement

My friends and I like to practice playing different agents in Valorant, and we use a random agent selector to pick the agent we will play during the next game. It chooses between all of the agents each time it is used, so it often chooses agents we don’t have unlocked and has to be redone. It also has to be used at least one time for each person, which could mean up to 10 attempts to get a list of agents selected. This program will solve this issue by selecting an agent that it knows is unlocked for each person.

# 2. Requirements

1. The software shall have a GUI menu.
   1. Before launching the GUI, the software will run specific file checks.
   2. The menu shall have multiple options.
      1. Checkboxes for all saved players.
      2. A ‘Select Agents’ button
      3. A ‘Select Map’ button
      4. An ‘Add Agent to Player’ button
      5. An ‘Add New Player’ button
      6. An ‘Add New Agent’ button
      7. An ‘Add New Map’ button
      8. An ‘Exit’ button
   3. Buttons are to be separated with horizontal lines.
   4. The program shall run successfully even if none of the config files are present, and prompt the user to add agents, maps, and/or players.
2. The software shall not repeat agents during a single selection. Although the agent selection is random, a single agent cannot be given to more than one player.
3. The software shall have a button to allow the user to copy the text to their clipboard.
4. The software shall allow for randomly selecting agents for the checked players.
5. The software shall allow for randomly selecting a map.
6. The software shall allow for updating which agents a player has. This includes both adding (checking the checkbox) and removing (unchecking the checkbox) agents from a player.
7. A .ini config file will be stored containing all players and their unlocked agents.
   1. The .ini file will be named ‘players.ini.’
   2. Unlocked agents are saved under each section (player) as options and have a value of ‘1.’
   3. If an agent has a value of ‘0’, the option will be removed from the section on program start.
8. A .ini file shall be stored containing all agents available for selection in the game.
   1. This shall be appended to through an option in the menu.
   2. This file shall be called ‘agents.ini.’
9. A .ini file shall be stored containing all maps available for selection in the game.
   1. This shall be appended to through an option in the menu.
   2. This file shall be called ‘maps.ini.’

# 3. Software Construction

*'''Random Agent Selector by Cael Shoop.'''*

import tkinter as tk # GUI

from tkinter import ttk

from tkinter.messagebox import showerror, showinfo

import random # random number selector for choosing agents

import configparser as cp # Saving player info in .ini config file

import os

import sys

import time

# Second part of selecting agents function

def select\_agents(select\_win, playersCheckedVars):

picked\_win = tk.Toplevel(select\_win)

picked\_win.resizable(False, False)

picked\_win.configure(bg='black')

picked\_win.title('Agents Selected')

playersChecked = []

counter = 0

for player in playerConfig.sections(): # Creates a list of all player names

if playersCheckedVars[counter].get():

playersChecked.append(player)

counter += 1

if len(playersChecked) < 1:

showerror(title='Error', message='Please select at least one player.')

refresh()

usedAgents = [] # List to prevent repeated agents

output = []

longestOutput = 0

for player in playersChecked: # Selects and displays the random agent for each player

random.seed(round(time.time() \* 1000))

picked = random.randint(1, len(playerConfig.options(player))) - 1

infinite = 0

while agents[picked] in usedAgents or not playerConfig.has\_option(player, agents[picked].lower()):

picked = random.randint(1, len(agentConfig.sections())) - 1

infinite += 1

if infinite > 1000000:

showerror(title='Error', message='Failed to pick agents. Please try again.')

refresh()

usedAgents.append(agents[picked])

newOutput = str(player) + ': ' + str(agents[picked]) + '\n'

if len(newOutput) + 2 > longestOutput: # Updates longest output, +2 is for brackets

longestOutput = len(newOutput) + 2

output.append(newOutput)

picked = -1

text = tk.Text(picked\_win, width=longestOutput, height=len(output))

for ii in range(len(output)):

line = str(ii + 1) + '.0'

outputStep = output[ii].strip('{').strip('}')

text.insert(line, outputStep) # Outputs selected agents in text fields

text['state'] = 'disabled'

text.pack(padx=5, pady=5)

# Button to copy output to clipboard

tk.Button(picked\_win, text='Copy', bg='gray20', fg='white', relief='ridge', command=lambda: copy(picked\_win, text)).pack(pady=5)

ttk.Separator(picked\_win, orient='horizontal').pack(fill='x', pady=5)

# Closes this window

tk.Button(picked\_win, text='Close', bg='gray20', fg='white', relief='ridge', command=lambda: picked\_win.destroy()).pack(pady=5)

# Copies everything in the text field to clipboard

def copy(picked\_win, text):

picked\_win.clipboard\_clear()

picked\_win.clipboard\_append(text.get('1.0', 'end').strip('\n\n'))

picked\_win.update()

# Selects a random map

def select\_map(window):

maps = []

for map in mapConfig.sections():

maps.append(map)

random.seed(round(time.time() \* 1000))

map = random.randint(1, len(maps)) - 1

showinfo(title='Map Selection', message=f'Selected map: {maps[map]}')

# First part of adding agents to a player

def add\_agent(window):

add\_a\_win = tk.Toplevel(window)

add\_a\_win.resizable(False, False)

add\_a\_win.configure(bg='black')

add\_a\_win.title('Add Agent(s) to Player')

tk.Label(add\_a\_win, bg='black', fg='white', text='Select a Player:').pack(padx=75, pady=5)

selPlayer = tk.StringVar() # Selected player variable

menu\_button = tk.Menubutton(add\_a\_win, bg='grey75', disabledforeground='grey20', text='Click Here') # Dropdown list

menu = tk.Menu(menu\_button, tearoff=0)

players = playerConfig.sections()

for player in players: # Adds each player to the dropdown list

# When a radiobutton is selected, it calls show\_agents()

menu.add\_radiobutton(label=player, value=player, variable=selPlayer, command=lambda: show\_agents(add\_a\_win, selPlayer.get(), menu\_button))

menu\_button['menu'] = menu

menu\_button.pack(padx=50, pady=5)

# This shows each of the agents. I figured out the other way that I used for the players

# later on after a couple hours of trial and error, but I already wrote it this way

# and it works so I don't want to change it until I have a reason to.

def show\_agents(add\_win, player, menu\_button):

menu\_button['state'] = 'disabled'

menu\_button['text'] = player

ttk.Separator(add\_win, orient='horizontal').pack(fill='x', pady=5)

agentsCheckedVars = []

counter = 0

for agent in agentConfig.sections(): # Creates a list of tk variables

if playerConfig.has\_option(player, agent):

agentCheckVars = tk.IntVar(add\_win, playerConfig.getint(player, agent))

else:

agentCheckVars = tk.IntVar(add\_win, 0)

agentsCheckedVars.append([agents[counter], agentCheckVars])

counter += 1

counter = 0

for agent in agentConfig.sections(): # Displays a checkbox for each player

tk.Checkbutton(add\_win, bg='black', activebackground='gray20', fg='red', activeforeground='white', text=agent, variable=agentsCheckedVars[counter][1], onvalue=1, offvalue=0).pack()

counter += 1

# Saves player name and agents to config file

tk.Button(add\_win, text='Save', bg='gray20', fg='white', relief='ridge', command=lambda: a\_config(add\_win, player, agentsCheckedVars)).pack(pady=5)

# Function to specifically add agents to players

def a\_config(add\_a\_win, player, agentsCheckedVars):

for agent in agentsCheckedVars:

# Adds every agent (and value, could be 0 for not having the agent) to each player

playerConfig.set(player, agent[0], str(agent[1].get()))

with open('players.ini', 'w') as configfile: # Write to the config file

playerConfig.write(configfile)

# Closes the toplevel window

refresh()

# Function to add a player and their unlocked agents to the player config file

def add\_player(window):

add\_win = tk.Toplevel(window)

add\_win.resizable(False, False)

add\_win.configure(bg='black')

add\_win.title('Add New Player')

tk.Label(add\_win, bg='black', fg='white', text='Player Name:').pack(pady=5) # Label for input field

entryName = tk.StringVar() # For taking player's name as input

intake = tk.Entry(add\_win, textvariable=entryName, width=40)

intake.pack(padx=10, pady=5) # Text entry field

agentsCheckedVars = []

if agentConfig.sections():

ttk.Separator(add\_win, orient='horizontal').pack(fill='x', pady=5)

tk.Label(add\_win, bg='black', fg='white', text='Select this Player\'s Unlocked Agents:').pack(pady=5)

for ii in range(len(agentConfig.sections())): # Creates a list of tk variables

if ii == 2 or ii == 5 or ii == 9 or ii == 10 or ii == 12:

agentCheckVars = tk.IntVar(add\_win, 1)

else:

agentCheckVars = tk.IntVar(add\_win, 0)

agentsCheckedVars.append(agentCheckVars)

counter = 0

for agent in agentConfig.sections(): # Displays a checkbox for each player

tk.Checkbutton(add\_win, bg='black', activebackground='gray20', fg='red', activeforeground='white', text=agent, variable=agentsCheckedVars[counter], onvalue=1, offvalue=0).pack()

counter += 1

# Saves player name and agents to config file

tk.Button(add\_win, text='Save', bg='gray20', fg='white', relief='ridge', command=lambda: p\_config(add\_win, intake, agentsCheckedVars)).pack(pady=5)

ttk.Separator(add\_win, orient='horizontal').pack(fill='x', pady=5)

tk.Button(add\_win, text='Back', bg='gray20', fg='white', relief='ridge', command=lambda: add\_win.destroy()).pack(pady=5)

# Appends new section to the player config file

def p\_config(add\_win, intake, agentsCheckedVars):

agentsChecked = []

counter = 0

for agent in agentConfig.sections(): # Creates a list of all player names

if agentsCheckedVars[counter].get():

agentsChecked.append(agent)

counter += 1

name = intake.get() # Retrieves string from variable

if playerConfig.has\_section(name): # If player exists in config file, close window

showerror(title='Error', message='Player already exists.')

add\_win.destroy()

return

playerConfig\_add = cp.ConfigParser()

playerConfig\_add.add\_section(name)

for agent in agentsChecked: # creates the section in a new config instance

playerConfig\_add.set(name, agent, '1')

with open('players.ini', 'a') as configfile: # Appends the new instance to config file

playerConfig\_add.write(configfile)

# Destroy toplevel window and refresh

add\_win.destroy()

refresh()

# For adding a new agent to the agent config file

def new\_agent(window):

new\_win = tk.Toplevel(window)

new\_win.resizable(False, False)

new\_win.configure(bg='black')

new\_win.title('Add New Agent')

tk.Label(new\_win, bg='black', fg='white', text='Agent Name:').pack(pady=5) # Label for input field

entryName = tk.StringVar() # For taking player's name as input

intake = tk.Entry(new\_win, textvariable=entryName, width=40)

intake.pack(padx=10, pady=5) # Text entry field

tk.Button(new\_win, text='Save', bg='gray20', fg='white', relief='ridge', command=lambda: a\_n\_config(new\_win, intake)).pack(pady=5)

ttk.Separator(new\_win, orient='horizontal').pack(fill='x', pady=5)

tk.Button(new\_win, text='Back', bg='gray20', fg='white', relief='ridge', command=lambda: new\_win.destroy()).pack(pady=5)

# Appends new section to agent config file

def a\_n\_config(new\_win, intake):

name = intake.get()

if agentConfig.has\_section(name): # If agent exists in config file, close window

showerror(title='Error', message='Agent already exists.')

new\_win.destroy()

return

agentConfig\_add = cp.ConfigParser()

agentConfig\_add.add\_section(name)

with open('agents.ini', 'a') as configfile:

agentConfig\_add.write(configfile)

new\_win.destroy()

refresh()

# Adds new map to map config file

def new\_map(window):

new\_win = tk.Toplevel(window)

new\_win.resizable(False, False)

new\_win.configure(bg='black')

new\_win.title('Add New Map')

tk.Label(new\_win, bg='black', fg='white', text='Map Name:').pack(pady=5) # Label for input field

entryName = tk.StringVar() # For taking player's name as input

intake = tk.Entry(new\_win, textvariable=entryName, width=40)

intake.pack(padx=10, pady=5) # Text entry field

tk.Button(new\_win, text='Save', bg='gray20', fg='white', relief='ridge', command=lambda: m\_config(new\_win, intake)).pack(pady=5)

ttk.Separator(new\_win, orient='horizontal').pack(fill='x', pady=5)

tk.Button(new\_win, text='Back', bg='gray20', fg='white', relief='ridge', command=lambda: new\_win.destroy()).pack(pady=5)

# Appends new section to map config file

def m\_config(new\_win, intake):

name = intake.get()

if mapConfig.has\_section(name): # If map exists in config file, close window

showerror(title='Error', message='Map already exists.')

new\_win.destroy()

return

mapConfig\_add = cp.ConfigParser()

mapConfig\_add.add\_section(name)

with open('maps.ini', 'a') as configfile:

mapConfig\_add.write(configfile)

new\_win.destroy()

refresh()

# Refreshes the window by using execv to run the program again

def refresh():

os.execv(sys.executable, ['python3'] + sys.argv)

# Closes the window and exits the program

def close(window):

window.destroy()

exit()

def main():

# GUI

window = tk.Tk()

window.resizable(False, False)

window.configure(bg='black')

window.title('Random Agent Selector')

# Global config instances and list of Valorant agents

global agentConfig

agentConfig = cp.ConfigParser()

try:

agentConfig.read\_file(open('agents.ini'))

except: # Throws non-fatal error if file doesn't exist

showerror(title='Error', message='agents.ini not found. Please add agents.')

agentConfig.read('agents.ini')

global agents

agents = [] # List of agents

for agent in agentConfig.sections():

agents.append(agent)

agents.sort()

newAConfig = cp.ConfigParser()

for agent in agents:

newAConfig.add\_section(agent) # Sort all agents into new config file and write it

with open('agents.ini', 'w') as configfile:

newAConfig.write(configfile)

global playerConfig

playerConfig = cp.ConfigParser()

try:

playerConfig.read\_file(open('players.ini'))

except:

showerror(title='Error', message='players.ini not found. Please add players.')

playerConfig.read('players.ini')

changed = False

for player in playerConfig.sections():

for option in playerConfig.options(player): # Remove all agent options

if not playerConfig.getint(player, option): # with a value of 0

playerConfig.remove\_option(player, option)

changed = True

if changed:

with open('players.ini', 'w') as configfile:

playerConfig.write(configfile)

sortingList = []

for player in playerConfig.sections():

sortingList.append([len(playerConfig.options(player)), player, playerConfig.options(player)])

sortingList.sort()

newPConfig = cp.ConfigParser() # Same as agents sorting,

for item in sortingList: # but also has options

newPConfig.add\_section(item[1]) # with unlocked agents

item[2].sort()

for option in item[2]:

newPConfig.set(item[1], str(option), '1')

with open('players.ini', 'w') as configfile: # It actually sorts it by number of

newPConfig.write(configfile) # unlocked agents to make selection

global mapConfig # more successful

mapConfig = cp.ConfigParser()

try:

mapConfig.read\_file(open('maps.ini'))

except:

showerror(title='Error', message='maps.ini not found. Please add maps.')

mapConfig.read('maps.ini')

mapList = mapConfig.sections()

mapList.sort()

with open('maps.ini', 'w') as configfile:

mapConfig.write(configfile)

xPadding = 100 # Makes adjusting window sizing easier

# Config

if playerConfig.sections() and agentConfig.sections():

# Buttons

tk.Label(window, bg='black', fg='white', text='Select current players:').pack(padx=xPadding, pady=5)

playersCheckedVars = []

for ii in range(len(playerConfig.sections())): # Creates a list of tk variables

playerCheckVars = tk.IntVar(window, 0)

playersCheckedVars.append(playerCheckVars)

counter = 0

for player in playerConfig.sections(): # Displays a checkbox for each player

tk.Checkbutton(window, bg='black', activebackground='gray20', fg='red', activeforeground='white', text=player, variable=playersCheckedVars[counter], onvalue=1, offvalue=0).pack()

counter += 1

tk.Button(window, text='Select Agents', bg='gray20', fg='white', relief='ridge', command=lambda: select\_agents(window, playersCheckedVars)).pack(padx=50, pady=5)

ttk.Separator(window, orient='horizontal').pack(fill='x', pady=5)

if mapConfig.sections():

tk.Button(window, text='Select Map', bg='gray20', fg='white', relief='ridge', command=lambda: select\_map(window)).pack(padx=50, pady=5)

ttk.Separator(window, orient='horizontal').pack(fill='x', pady=5)

if playerConfig.sections() and agentConfig.sections():

tk.Button(window, text='Add Agent to Player', bg='gray20', fg='white', relief='ridge', command=lambda: add\_agent(window)).pack(padx=xPadding, pady=5)

ttk.Separator(window, orient='horizontal').pack(fill='x', pady=5)

tk.Button(window, text='Add New Player', bg='gray20', fg='white', relief='ridge', command=lambda: add\_player(window)).pack(padx=xPadding, pady=5)

ttk.Separator(window, orient='horizontal').pack(fill='x', pady=5)

tk.Button(window, text='Add New Agent', bg='gray20', fg='white', relief='ridge', command=lambda: new\_agent(window)).pack(padx=xPadding, pady=5)

ttk.Separator(window, orient='horizontal').pack(fill='x', pady=5)

tk.Button(window, text='Add New Map', bg='gray20', fg='white', relief='ridge', command=lambda: new\_map(window)).pack(padx=xPadding, pady=5)

ttk.Separator(window, orient='horizontal').pack(fill='x', pady=5)

tk.Button(window, text='Exit', bg='gray20', fg='white', relief='ridge', command=lambda: close(window)).pack(padx=xPadding, pady=5)

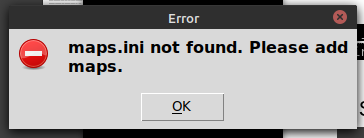
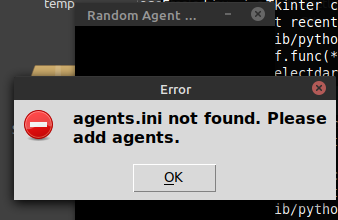
# Show window

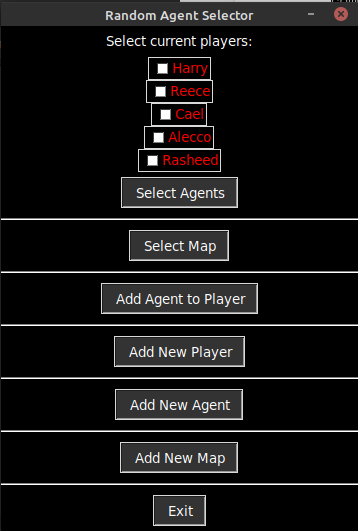
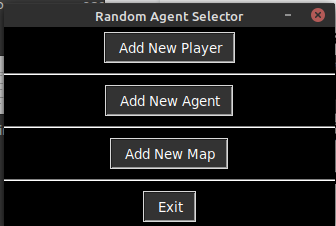
window.mainloop()

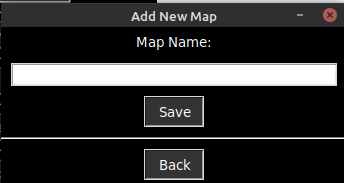
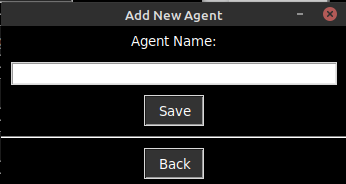
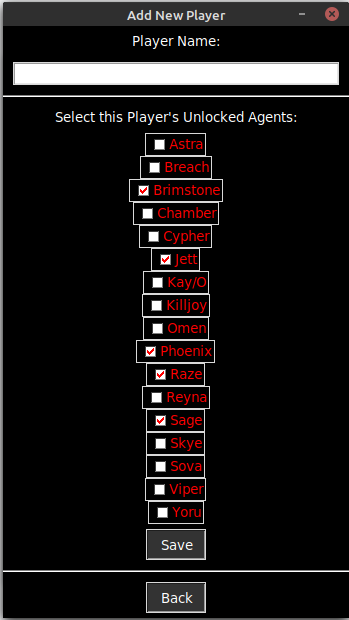
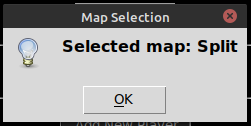
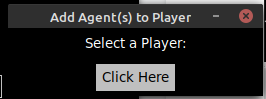
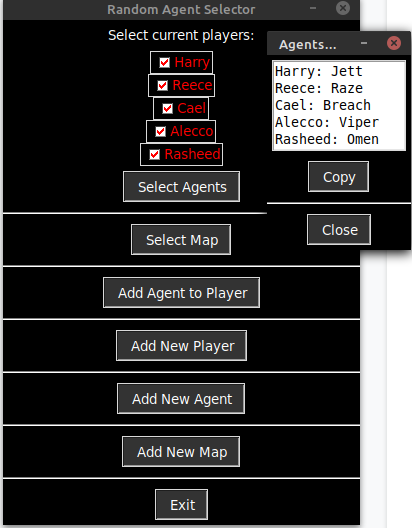
if \_\_name\_\_ == '\_\_main\_\_':

main()

# 4. Software Testing

**

**

**

# 5. Conclusions, & Future Directions

Getting to learn and practice using tkinter and configparser has been very useful. Configparser is used in an API for controlling Roombas that I have been using for senior project, and I did not know how to utilize it before I started this project.

About halfway through, it was over 300 lines long due to hard coding the checkbuttons, and I got it down to about 214 lines long until we tested it out while playing the game. We then realized that a map selection feature would also be useful and very easy to implement.

I ended up figuring out how to make a list of variables for the checkboxes and using that list to separate each checkbox, which massively reduced the line count and allowed for full flexibility with adding agents without modifying the code.

It’s currently designed to run itself again whenever it is refreshed, which works in linux perfectly but doesn’t work in Windows. Since my friend usually uses it and I don’t have Python installed on my desktop, he runs it on his Windows desktop, and whenever it refreshes it restarts the program to reload the config files. Since it doesn’t restart on Windows, I wrote him a quick batch script to loop running the command.

Future modifications would include implementing a more efficient method of refreshing the window.

# 6. References

[*https://docs.python.org/3/library/tkinter.html*](https://docs.python.org/3/library/tkinter.html)

[*https://www.geeksforgeeks.org/python-tkinter-tutorial/*](https://www.geeksforgeeks.org/python-tkinter-tutorial/)

[*https://pythonbasics.org/tkinter/*](https://pythonbasics.org/tkinter/)

[*https://docs.python.org/3/library/configparser.html*](https://docs.python.org/3/library/configparser.html)

[*https://www.tutorialspoint.com/configuration-file-parser-in-python-configparser*](https://www.tutorialspoint.com/configuration-file-parser-in-python-configparser)