**Introduction to Programming-2 (Python)**

# Yermek, Temirlan, se2218, 220063

ASSIGNMENT 6

Problem – a0  
  
my\_bot.py (were changed only this part of code)

from random import randint

#import numpy as np

def name():

return "Bot-MY\_BOT"

def main():

#board = np.full((6,7),0)

pass

def turn(board, p = 1, e = 2):

for col in range(7):

if can\_win(board, col, p):

return col

for col in range(7):

if can\_win(board, col, e):

return col

non\_full\_cols = [col for col in range(7) if isempty(board, col)]

return non\_full\_cols[randint(0, len(non\_full\_cols) - 1)]

def can\_win(board, col, player):

test\_board = board.copy()

row = 5

while row >= 0 and test\_board[row, col] != 0:

row -= 1

if row < 0:

return False

test\_board[row, col] = player

return is\_winning\_move(test\_board, row, col, player)

def is\_winning\_move(board, row, col, player):

count = 0

for i in range(max(0, col - 3), min(4, col) + 1):

if board[row, i] == player:

count += 1

if count == 4:

return True

else:

count = 0

count = 0

for i in range(max(0, row - 3), min(2, row) + 1):

if board[i, col] == player:

count += 1

if count == 4:

return True

else:

count = 0

count = 0

i = max(row - 3, 0)

j = max(col - 3, 0)

while i <= min(row + 3, 5) and j <= min(col + 3, 6):

if board[i, j] == player:

count += 1

if count == 4:

return True

else:

count = 0

i += 1

j += 1

count = 0

i = max(row - 3, 0)

j = min(col + 3, 6)

while i <= min(row + 3, 5) and j >= max(col - 3, 0):

if board[i, j] == player:

count += 1

if count == 4:

return True

else:

count = 0

i += 1

j -= 1

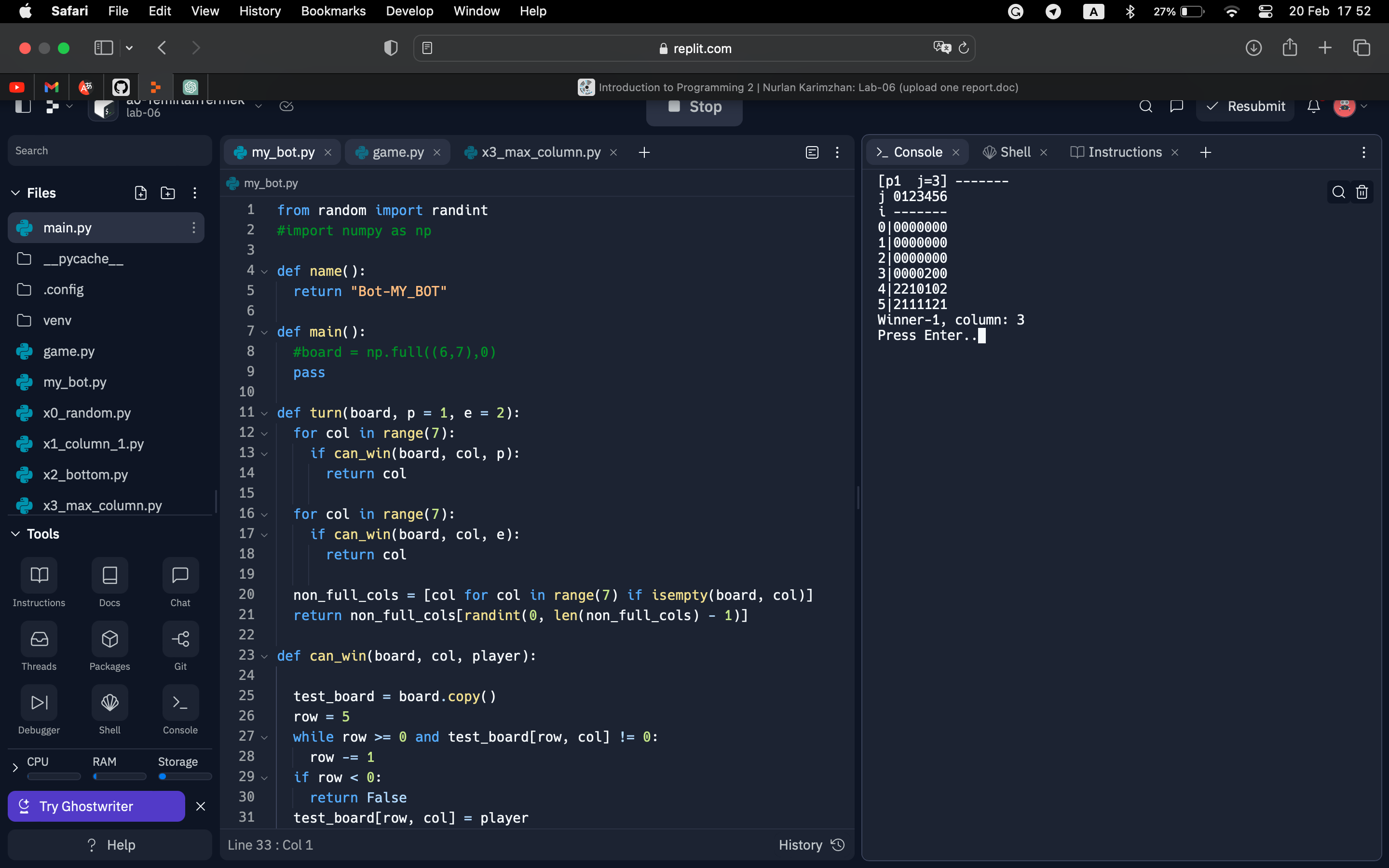
return False

def isempty(board, col\_j): # checks, is column's "col\_j" top empty?

if board[0,col\_j] != 0: return False # 1 or 2 atop

else: return True # 0 atop, we can put there

if \_\_name\_\_ == "\_\_main\_\_":

main()  
  


**Problem – a1**

import numpy as np

def read\_file(filename):

with open(filename, 'r') as file:

lines = file.readlines()

matrix = np.zeros((len(lines), len(lines[0].split())), dtype=int)

for i, line in enumerate(lines):

matrix[i] = np.array([int(x) for x in line.split()])

return matrix

def print\_matrix(matrix):

for row in matrix:

print(' '.join(str(x) for x in row))

def main\_diagonal\_mean(matrix):

diagonal = np.diag(matrix)

return diagonal.mean()

def second\_diagonal\_median(matrix):

diagonal = np.diag(np.fliplr(matrix))

return np.median(diagonal)

def sum\_mean\_median(filename):

matrix = read\_file(filename)

print\_matrix(matrix)

sum\_value = main\_diagonal\_mean(matrix) + second\_diagonal\_median(matrix)

print('{:.1f}'.format(sum\_value))

sum\_mean\_median('test-01.txt')

sum\_mean\_median('test-02.txt')  
Graphical user interface, text, application

Description automatically generated

Problem – a2

import numpy as np

def read\_file(filename):

with open(filename, 'r') as file:

lines = file.readlines()

matrix = np.zeros((len(lines), len(lines[0].split())), dtype=int)

for i, line in enumerate(lines):

matrix[i] = np.array([int(x) for x in line.split()])

return matrix

def print\_matrix(matrix):

for row in matrix:

print(' '.join(str(x) for x in row))

def replace\_diagonals\_with\_zero(matrix):

size = len(matrix)

for i in range(size):

matrix[i][i] = 0

matrix[i][size-i-1] = 0

return matrix

def save\_to\_file(matrix, filename):

with open(filename, 'w') as file:

for row in matrix:

file.write(str(row) + '\n')

def process\_matrix(filename):

matrix = read\_file(filename)

print\_matrix(matrix)

matrix = replace\_diagonals\_with\_zero(matrix)

print\_matrix(matrix)

save\_to\_file(matrix, 'result.txt')

process\_matrix('test-01.txt')

Graphical user interface, text

Description automatically generated

Problem – a3

import csv

import random

def main():

file\_name = input()

if file\_name == "":

file\_name = "gpa.csv"

generate\_csv\_file(file\_name)

print\_avg\_gpa(file\_name)

def print\_avg\_gpa(file\_name):

students = []

with open(file\_name) as file:

reader = csv.DictReader(file)

for row in reader:

id = row["ID"]

group = row["GROUP"]

gpa1 = int(row["GPA1"])

gpa2 = int(row["GPA2"])

gpa3 = int(row["GPA3"])

gpa\_avg = round((gpa1 + gpa2 + gpa3) / 3)

students.append({"ID": id, "GROUP": group, "GPA\_AVG": gpa\_avg})

for student in sorted(students, key=lambda student: student["ID"]):

print(f"{student['ID']}, {student['GROUP']}, {student['GPA\_AVG']}")

def generate\_csv\_file(file\_name):

with open(file\_name, "w", newline='') as file:

writer = csv.writer(file)

writer.writerow(["ID", "GROUP", "GPA1", "GPA2", "GPA3"])

for i in range(10):

id = f"220{random.randint(100, 999)}"

group = f"SE-22{random.randint(10, 99)}"

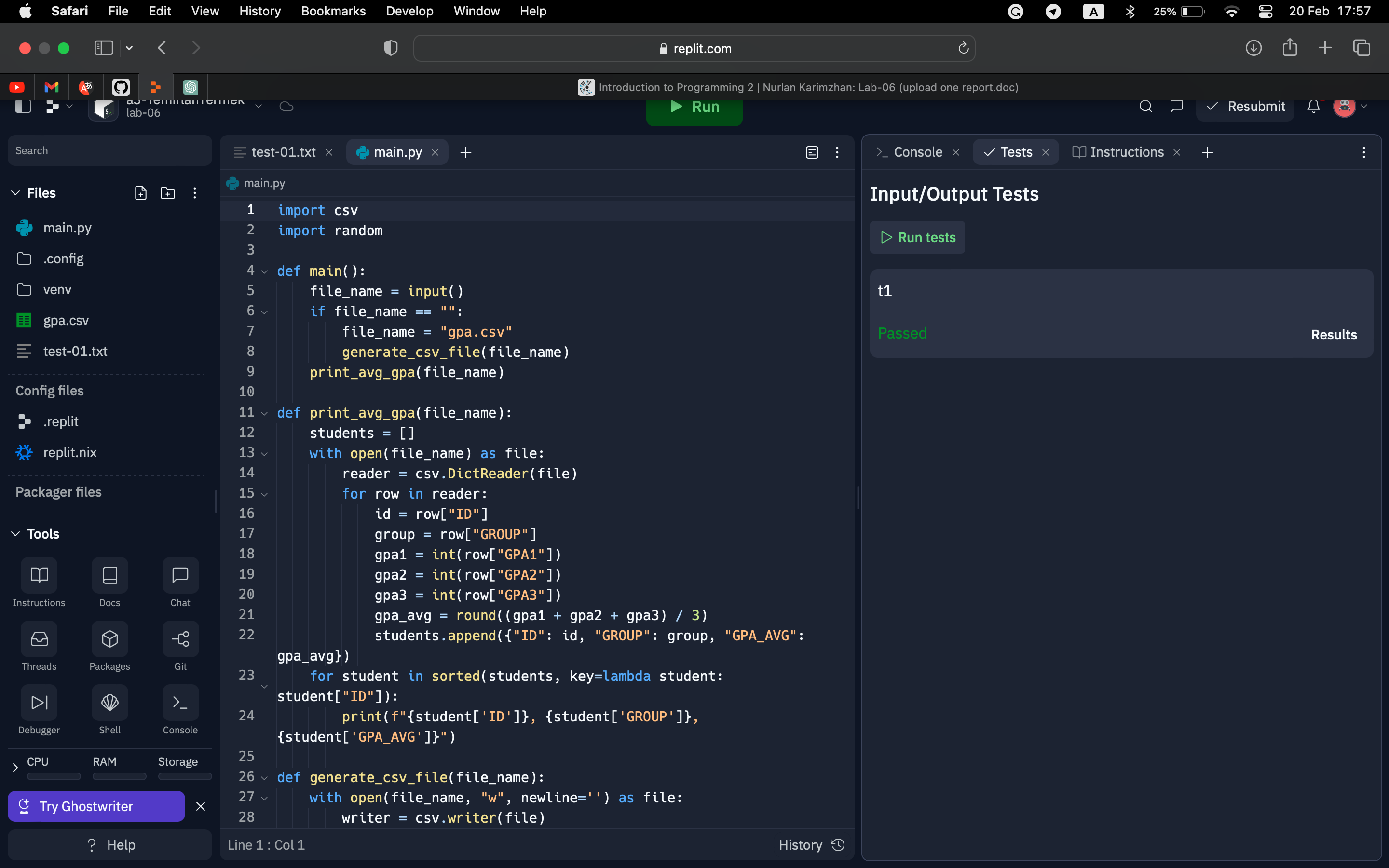
gpa1 = random.randint(0, 100)

gpa2 = random.randint(0, 100)

gpa3 = random.randint(0, 100)

writer.writerow([id, group, gpa1, gpa2, gpa3])

main()



Problem – a4

import re

def main():

phone = input("Phone number: ").strip()

print(validate(phone))

def validate(phone):

pattern = r'^\+\d\(\d{3}\)-\d{3}-\d{4}$'

match = re.search(pattern, phone)

return match is not None

main()

Graphical user interface, text

Description automatically generated

Problem – a5

import re

def main():

address = input("Address: ").strip()

print(validate(address))

def validate(address):

pattern = r'(?i)^[\w\s]+ st\. \d+, apt\. \d+, zip \d{6}$'

match = re.search(pattern, address.replace(" ", " "))

return match is not None

main()

Graphical user interface, text

Description automatically generated

Problem – a6

import re

def main():

print(recognize(input("Data Structure: ")))

def recognize(ds):

list\_pattern = r"^\[([\d.,\s]+)\]$"

tuple\_pattern = r"^\((.\*)\)$"

set\_pattern = r"^\{([\w.,\s]+)\}$"

dict\_pattern = r"^{([\w\d\s:'\",.\s]+)}$"

list\_match = re.match(list\_pattern, ds)

tuple\_match = re.match(tuple\_pattern, ds)

set\_match = re.match(set\_pattern, ds)

dict\_match = re.match(dict\_pattern, ds)

if list\_match:

values = list\_match.group(1).split(",")

if all(re.match(r"^\d+(.\d+)?$", v.strip()) for v in values):

return "List with Float"

elif tuple\_match:

values = tuple\_match.group(1).split(",")

if all(re.match(r"^[\w\s.'-]+$", v.strip()) for v in values):

return "Tuple with String"

elif set\_match:

values = set\_match.group(1).split(",")

if all(re.match(r"^\d+(\.\d+)?$", v.strip()) for v in values):

return "Set with Float"

elif dict\_match:

values = dict\_match.group(1).split(",")

if all(re.match(r"^\d+$", k.strip()) and re.match(r"^\d+$", v.strip()) for k, v in [p.split(":") for p in values]):

return "Dictionary with Integer"

return None

main()



Problem – a7

import re

def validate(ip):

pattern = r"^((\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5]).){3}(\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])$"

if re.match(pattern, ip):

return True

else:

return False

def main():

ip\_address = input("IPv4 address: ")

is\_valid = validate(ip\_address)

if is\_valid:

print(True)

else:

print(False)

main()

Graphical user interface, text

Description automatically generated

Problem – a8

def count\_lines\_of\_code(filename):

try:

with open(filename) as f:

lines = f.readlines()

except FileNotFoundError:

print(f"Error: File '{filename}' does not exist.")

return 0

lines = [line for line in lines if line.strip() and not line.startswith("#")]

return len(lines)

filename = input("Enter filename: ")

count = count\_lines\_of\_code(filename)

print(count)

Graphical user interface, text

Description automatically generated

Problem – a9

import csv

from tabulate import tabulate

def main():

filename = input("Enter the filename: ")

try:

with open(filename, newline='') as csvfile:

reader = csv.reader(csvfile)

table = []

for row in reader:

table.append(row)

print(tabulate(table, headers='firstrow',tablefmt='grid'))

except FileNotFoundError:

print("Error: File does not exist")

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

main()

