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

# Yermek, Temirlan, se2218, 220063

ASSIGNMENT 5 (Parts: A & B)

Problem – A1

import pyfiglet

def print\_asci(font\_name, text):

try:

f = pyfiglet.Figlet(font=font\_name)

print(f.renderText(text))

except ValueError:

print('Invalid usage')

return

font\_name = input("Enter font name: ")

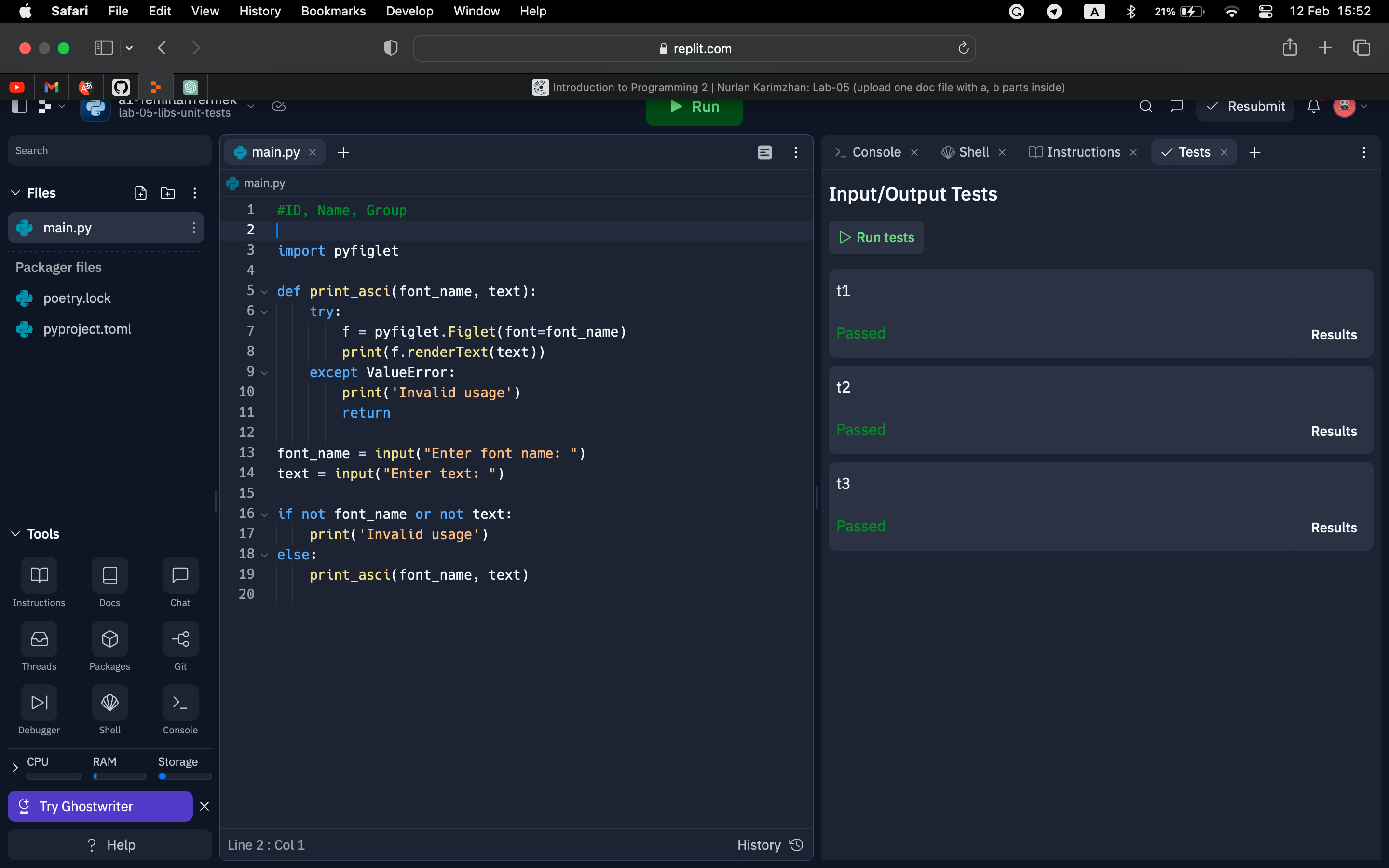
text = input("Enter text: ")

if not font\_name or not text:

print('Invalid usage')

else:

print\_asci(font\_name, text)



Problem – A2

import random

def play\_game(n):

target = random.randint(1, n)

while True:

try:

guess = int(input("Guess: "))

if guess <= 0:

print("Guess: ")

continue

if guess == target:

print("Just right!")

break

elif guess < target:

print("Too small!")

else:

print("Too large!")

except ValueError:

print("Guess: ")

while True:

try:

level = int(input("Level: "))

if level <= 0:

print("Level: ")

continue

break

except ValueError:

print("Level: ")

play\_game(level)

Graphical user interface, text

Description automatically generated

Problem – A3

def main():

A, B = get\_input()

op = int(input("Enter operation number (1-4): "))

sol(A, B, op)

def sol(A, B, op):

if op == 1:

result = A.union(B)

print("Union:", [str(x) for x in sorted(list(result))])

elif op == 2:

result = A.intersection(B)

print("Intersection:", [str(x) for x in sorted(list(result))])

elif op == 3:

result = A.difference(B)

print("Difference:", [str(x) for x in sorted(list(result))])

elif op == 4:

result = A.symmetric\_difference(B)

print("Symmetric Dif:", [str(x) for x in sorted(list(result))])

else:

print("Invalid operation number")

return

def get\_input():

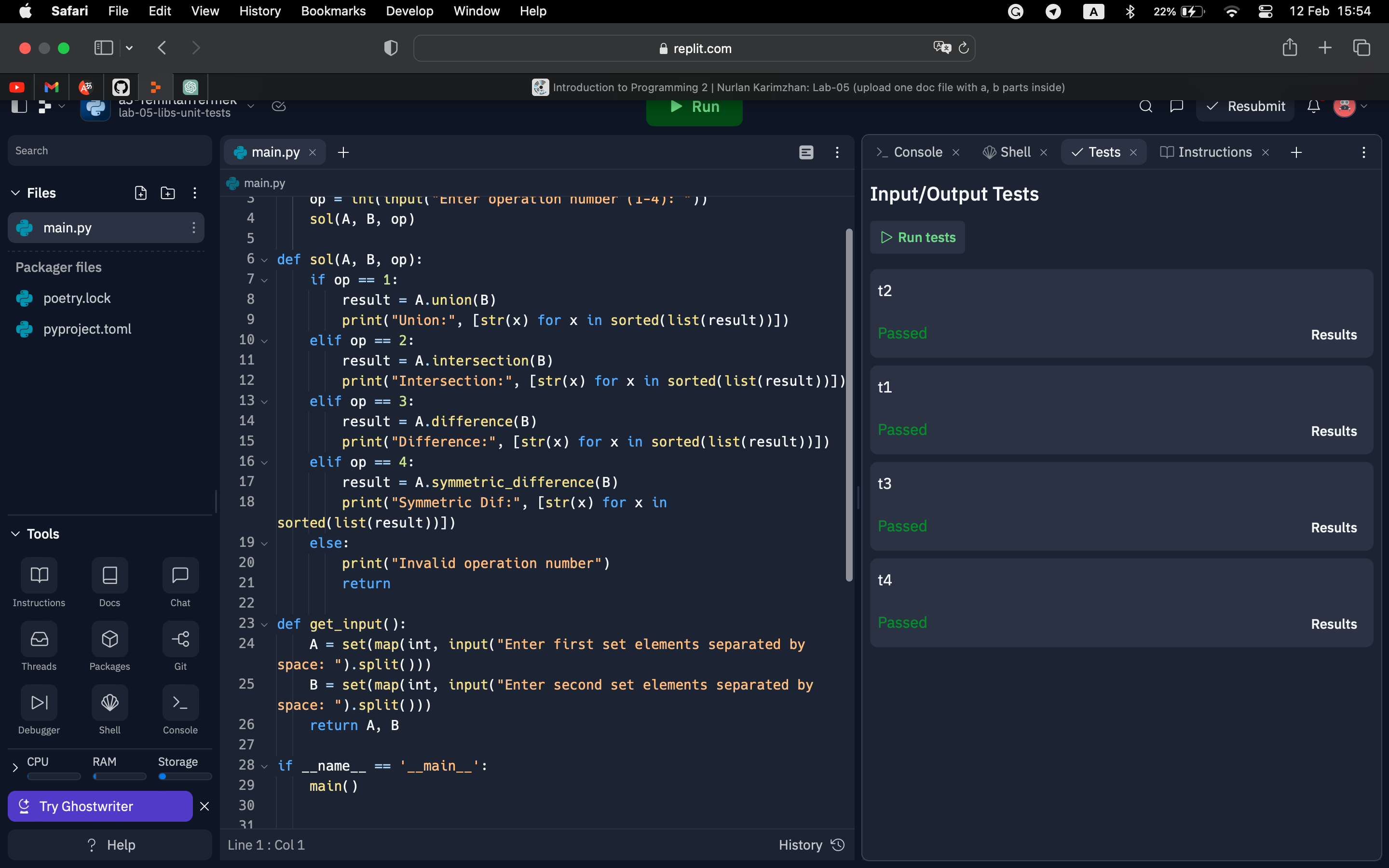
A = set(map(int, input("Enter first set elements separated by space: ").split()))

B = set(map(int, input("Enter second set elements separated by space: ").split()))

return A, B

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

main()



Problem – A4

def main():

t = get\_input()

print(sol(t))

def sol(t):

return tuple(sorted(t, key=lambda x: x[1]))

def get\_input():

t1 = input().split()

t2 = input().split()

t = tuple(zip(t1, t2))

return t

main()

Graphical user interface, text, application

Description automatically generated

Problem – B1

import emoji

import random

def main():

events = [":1st\_place\_medal:",":airplane:",":alien:",":bank:",

":beach\_with\_umbrella:",":house\_with\_garden:",

":couple\_with\_heart:",":tennis:",

":baby:",":sheep:",":graduation\_cap:",":key:",":laptop:",

":money\_bag:",":money\_with\_wings:",":open\_book:",

":package:",":wine\_glass:",":world\_map:",":gem:",":high\_heel:"]

future\_events = random.sample(events, 3)

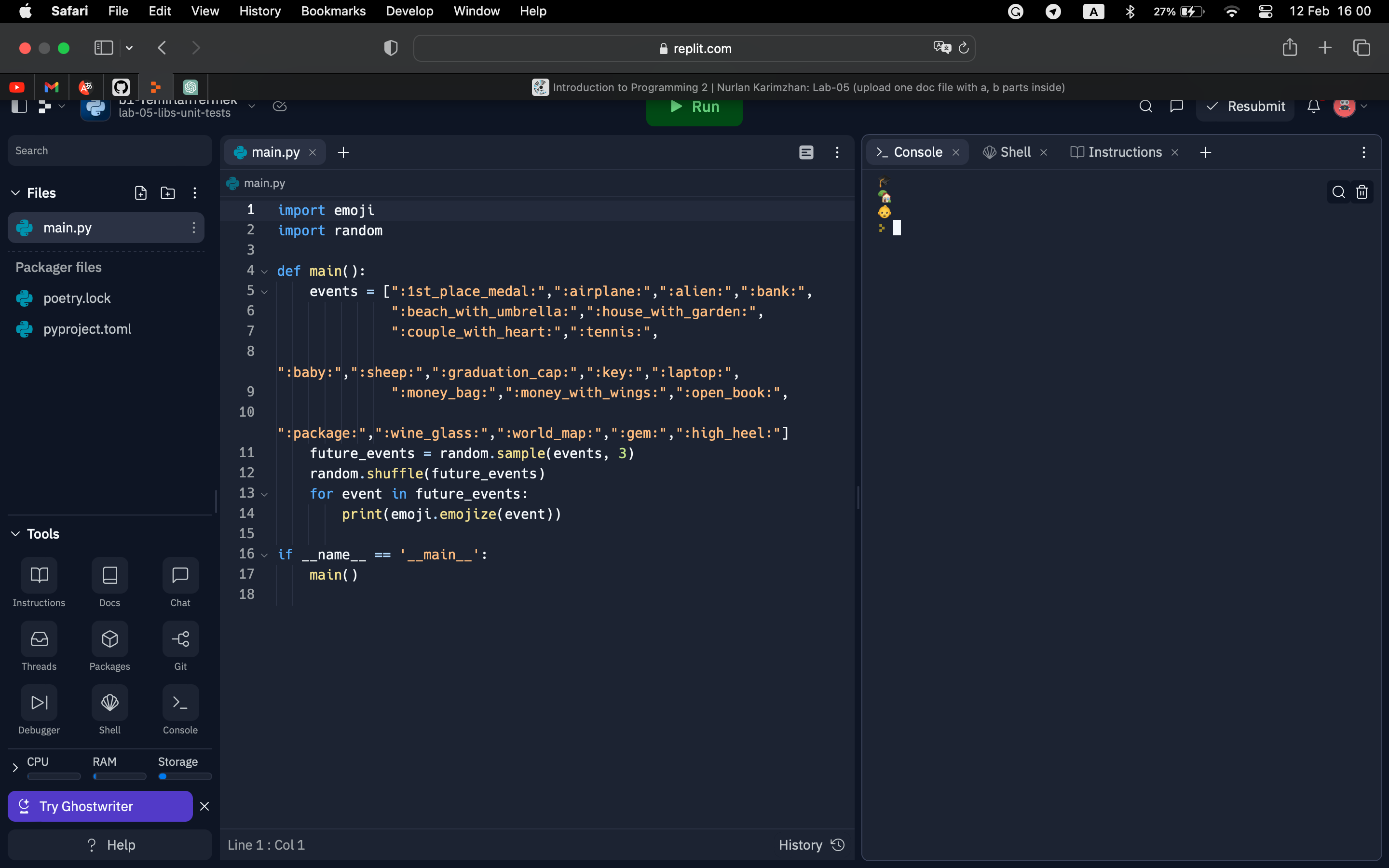
random.shuffle(future\_events)

for event in future\_events:

print(emoji.emojize(event))

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

main()



Problem – B2

import random

def main():

level = get\_level()

score = 0

for i in range(10):

x = generate\_integer(level)

y = generate\_integer(level)

correct\_answer = x + y

print("{} + {} = ".format(x, y), end="")

for j in range(3):

user\_answer = input().strip()

try:

user\_answer = int(user\_answer)

if user\_answer == correct\_answer:

score += 1

break

else:

print("EEE")

except:

print("EEE")

else:

print(correct\_answer)

print("Your score:", score)

def get\_level():

while True:

level = input("Level: ").strip()

try:

level = int(level)

if level in [1, 2, 3]:

return level

else:

print("Level should be 1, 2 or 3")

except:

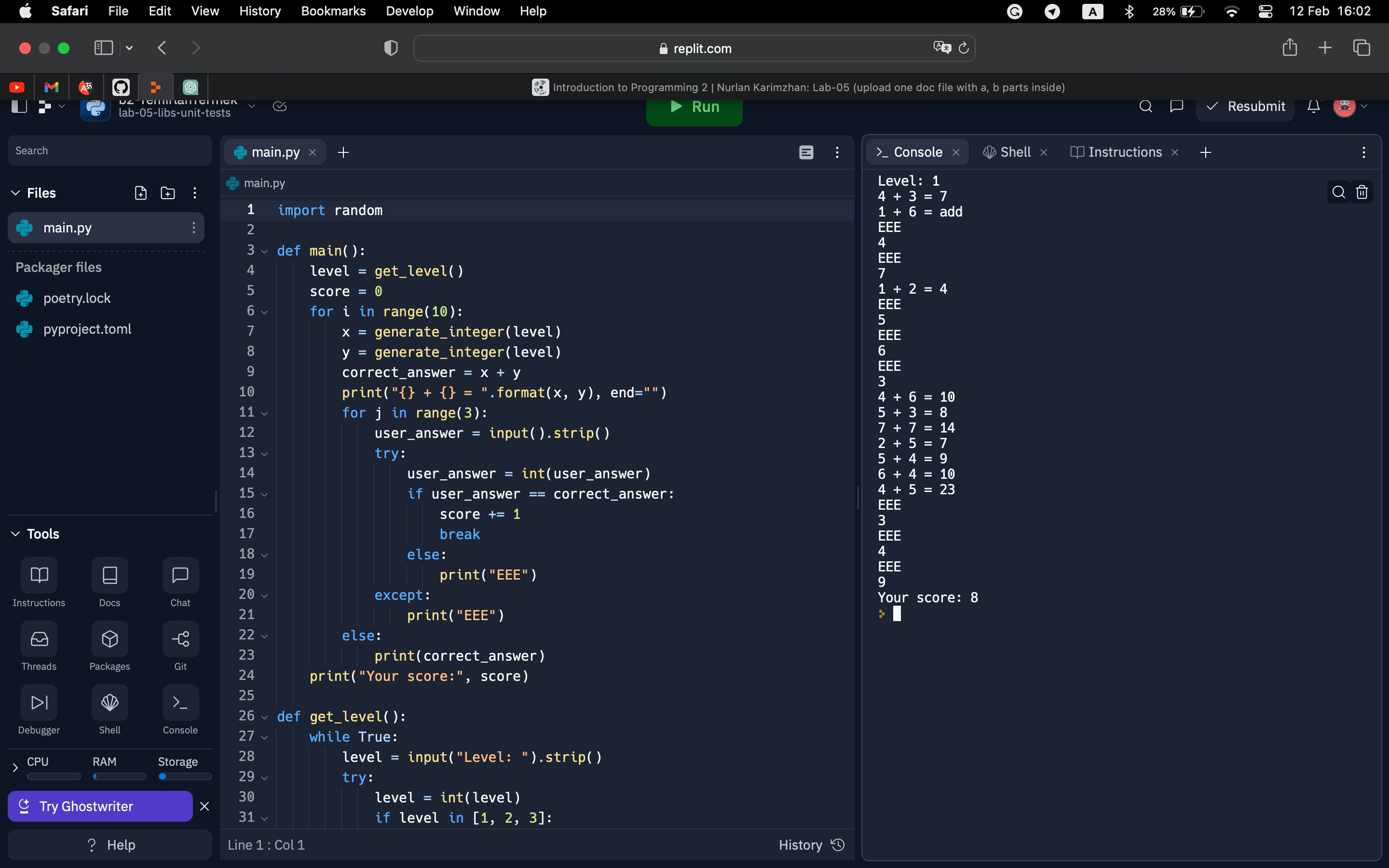
print("Level should be a number")

def generate\_integer(level):

return random.randint(10\*\*(level-1), 10\*\*level-1)

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

main()



Problem – B3

Main code:

def test\_shorten\_empty\_string():

assert twttr.shorten("") == ""

def test\_shorten\_vowel\_only\_string():

assert twttr.shorten("AEIOUaeiou") == ""

def test\_shorten\_consonant\_only\_string():

assert twttr.shorten("BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz") == "BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz"

def test\_shorten\_mixed\_string():

assert twttr.shorten("Hello World") == "Hll Wld"

def shorten(word):

vowels = "AEIOUaeiou"

result = ""

for char in word:

if char not in vowels:

result += char

return result

def main():

word = input("Enter a word: ")

short\_word = shorten(word)

print(short\_word)

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

main()  
  
  
test:

from main import shorten

def test\_empty():

assert shorten("") == ""

def test\_lowercase():

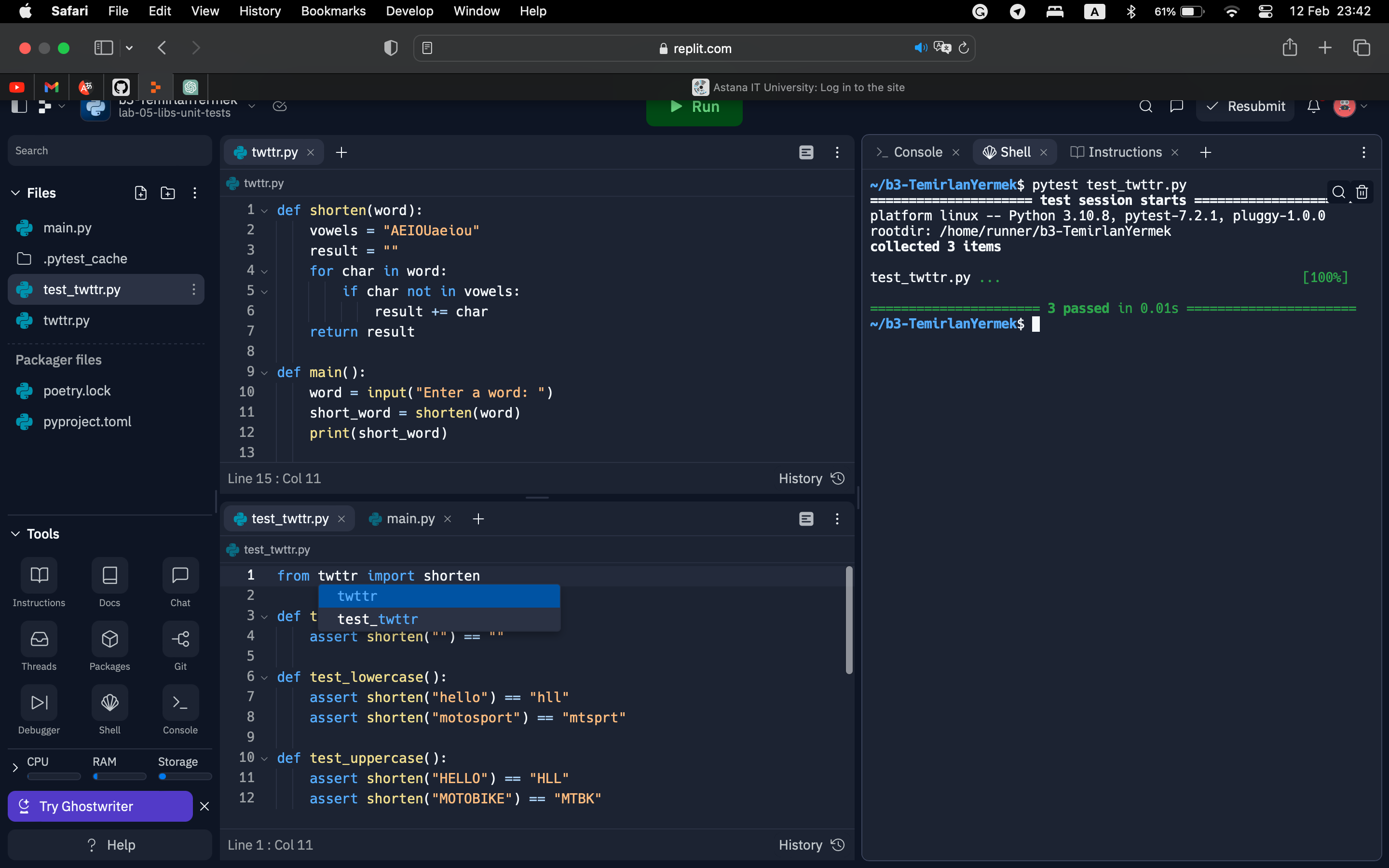
assert shorten("hello") == "hll"

assert shorten("motosport") == "mtsprt"

def test\_uppercase():

assert shorten("HELLO") == "HLL"

assert shorten("MOTOBIKE") == "MTBK"



Problem – B4

main code:

def main():

greeting = input("Enter greeting: ")

result = value(greeting)

print("Result: ", result)

def value(greeting):

greeting = greeting.lower()

if greeting.startswith("hello"):

return 0

elif greeting.startswith("h"):

return 20

else:

return 100

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

main()

( )

test:

import bank

def test\_value\_hello():

assert bank.value("Hello") == 0

assert bank.value("hello") == 0

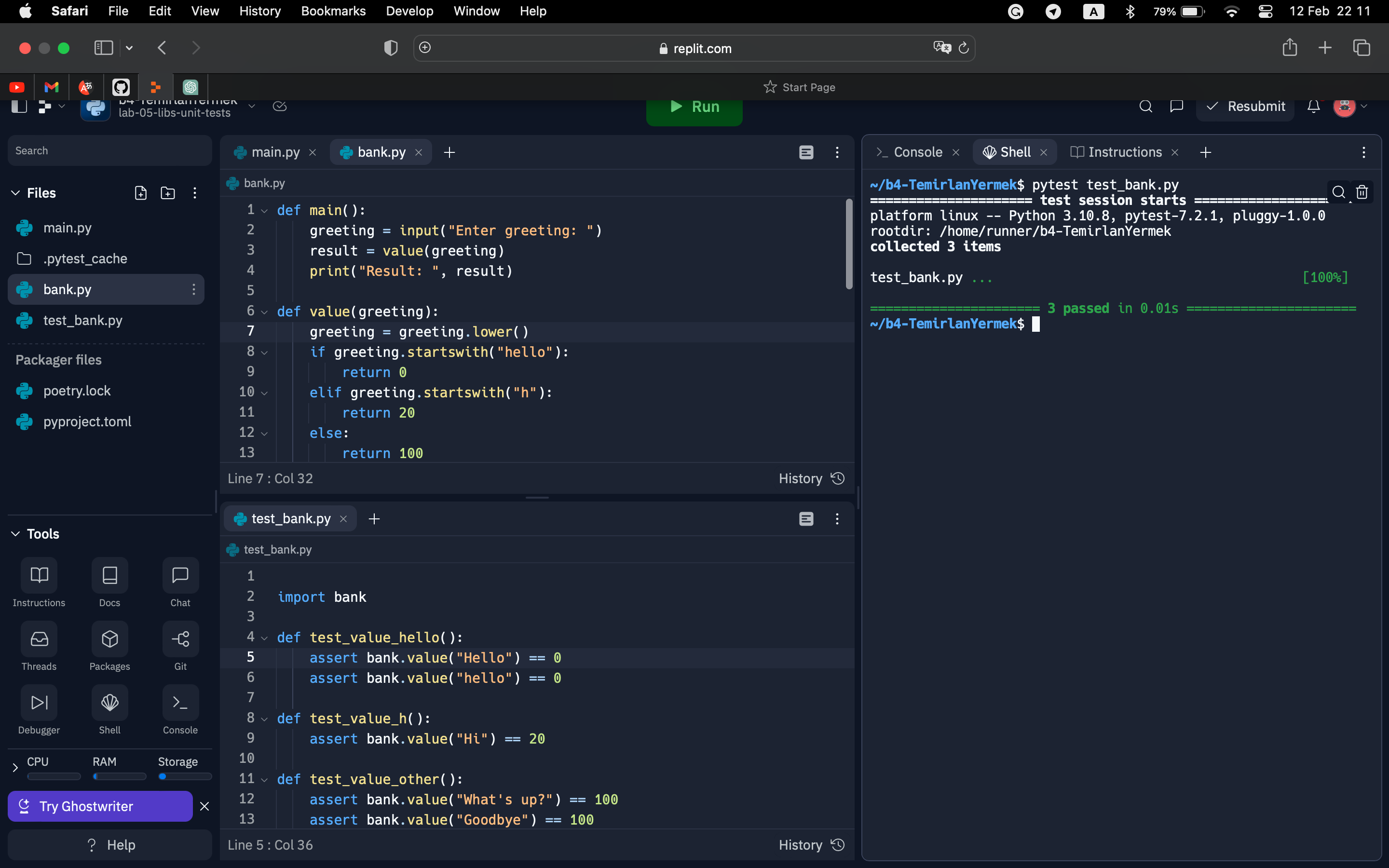
def test\_value\_h():

assert bank.value("Hi") == 20

def test\_value\_other():

assert bank.value("What's up?") == 100

assert bank.value("Goodbye") == 100



Problem – B5

Main:

def main():

plate = input("Enter a vanity plate: ")

if is\_valid(plate):

print("Valid vanity plate")

else:

print("Invalid vanity plate")

def is\_valid(s):

if len(s) > 8:

return False

for char in s:

if not char.isalnum():

return False

return True

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

main()

Test:

import pytest

from plates import is\_valid

def test\_is\_valid\_with\_valid\_input():

assert is\_valid("ABC123") == True

def test\_is\_valid\_with\_invalid\_length():

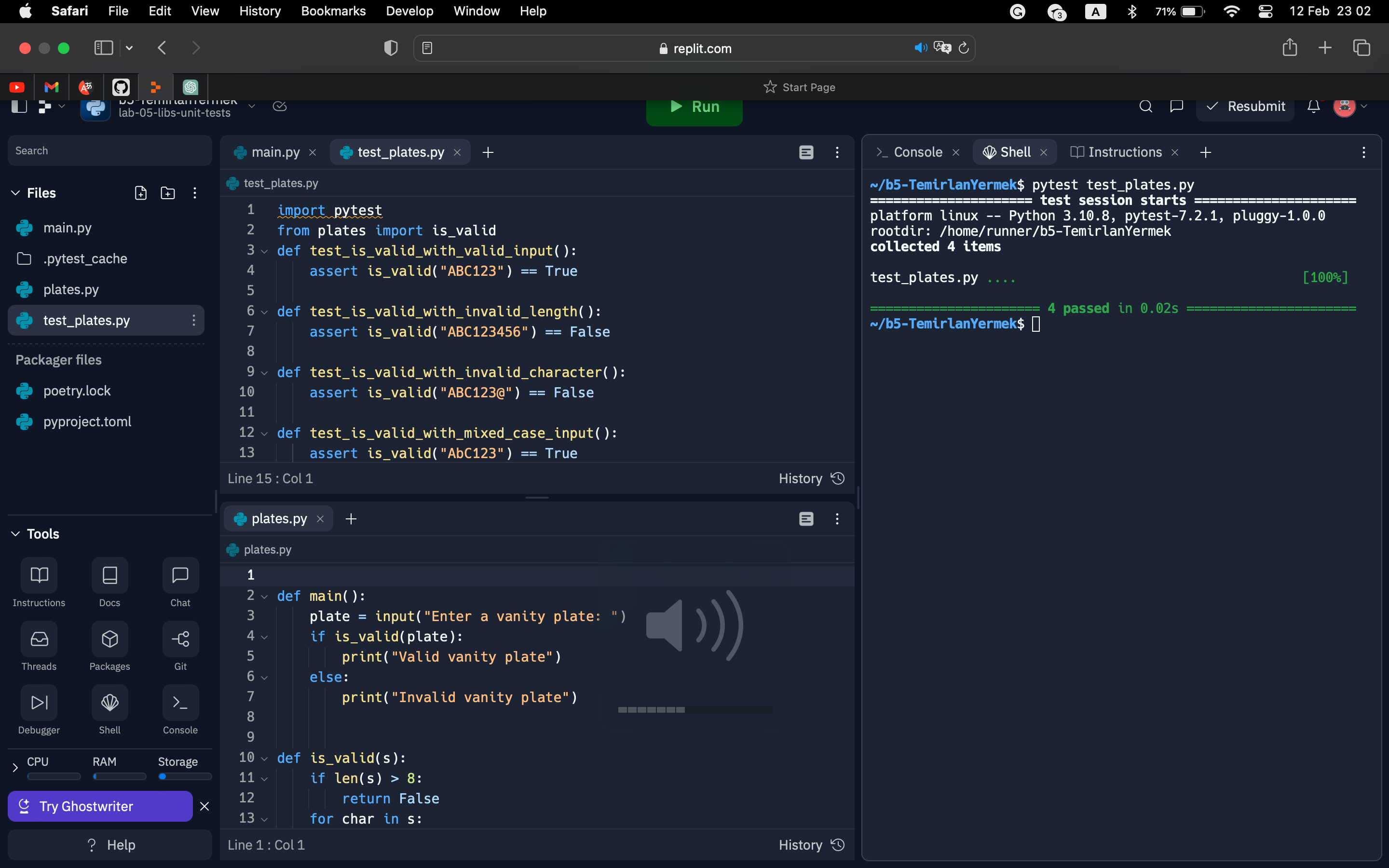
assert is\_valid("ABC123456") == False

def test\_is\_valid\_with\_invalid\_character():

assert is\_valid("ABC123@") == False

def test\_is\_valid\_with\_mixed\_case\_input():

assert is\_valid("AbC123") == True



Problem – B6

Main:

def convert(fraction):

try:

x, y = map(int, fraction.split("/"))

if x > y:

raise ValueError("X cannot be greater than Y")

if y == 0:

raise ZeroDivisionError("Y cannot be zero")

return round((x/y) \* 100)

except (ValueError, ZeroDivisionError) as e:

raise e

except:

raise ValueError("Input must be in the format X/Y, where X and Y are integers")

def gauge(percentage):

if percentage <= 1:

return "E"

elif percentage >= 99:

return "F"

else:

return str(percentage) + "%"

def main():

fraction = input("Enter a fraction in the format X/Y: ")

try:

percentage = convert(fraction)

print("Percentage:", gauge(percentage))

except (ValueError, ZeroDivisionError) as e:

print("Error:", e)

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

main()

Test:

import pytest

from fuel import convert, gauge

def test\_convert\_valid\_input():

assert convert("10/20") == 50

assert convert("50/100") == 50

def test\_convert\_invalid\_input():

with pytest.raises(ValueError, match="X cannot be greater than Y"):

convert("50/10")

def test\_gauge\_valid\_input():

assert gauge(0) == "E"

assert gauge(50) == "50%"

assert gauge(100) == "F"

