emails.py

#!/usr/bin/env python3

import csv

import sys

def populate\_dictionary(filename):

"""Populate a dictionary with name/email pairs for easy lookup."""

email\_dict = {}

with open(filename) as csvfile:

lines = csv.reader(csvfile, delimiter = ',')

for row in lines:

name = str(row[0].lower())

email\_dict[name] = row[1]

return email\_dict

def find\_email(argv):

""" Return an email address based on the username given."""

# Create the username based on the command line input.

try:

fullname = str(argv[1] + " " + argv[2])

# Preprocess the data

email\_dict = populate\_dictionary('/home/student-00-10f26c770d24/data/user\_emails.csv')

# If email exists, print it

if email\_dict.get(fullname.lower()):

return email\_dict.get(fullname.lower())

else:

return "No email address found"

except IndexError:

return "Missing parameters"

def main():

print(find\_email(sys.argv))

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

main()

emails\_test.py

#!/usr/bin/env python3

import unittest

from emails import find\_email

class EmailsTest(unittest.TestCase):

def test\_basic(self):

testcase = [None, "Bree", "Campbell"]

expected = "breee@abc.edu"

self.assertEqual(find\_email(testcase), expected)

def test\_one\_name(self):

testcase = [None, "John"]

expected = "Missing parameters"

self.assertEqual(find\_email(testcase), expected)

def test\_two\_name(self):

testcase = [None, "Roy","Cooper"]

expected = "No email address found"

self.assertEqual(find\_email(testcase), expected)

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

unittest.main()