Q1: start\_date\_report.py

def get\_start\_date():

"""Interactively get the start date to query for."""

print()

print('Getting the first start date to query for.')

print()

print('The date must be greater than Jan 1st, 2018')

year =int( input('Enter a value for the year: '))

month = int(input('Enter a value for the month: '))

day = int(input('Enter a value for the day: '))

print()

return datetime.datetime(year, month, day)

def get\_file\_lines(url):

"""Returns the lines contained in the file at the given URL"""

# Download the file over the internet

response = requests.get(url, stream=True)

lines = []

for line in response.iter\_lines():

lines.append(line.decode("UTF-8"))

return lines

def get\_same\_or\_newer(start\_date):

"""Returns the employees that started on the given date, or the closest one."""

data = get\_file\_lines(FILE\_URL)

reader = csv.reader(data[1:])

# We want all employees that started at the same date or the closest newer

# date. To calculate that, we go through all the data and find the

# employees that started on the smallest date that's equal or bigger than

# the given start date.

min\_date = datetime.datetime.today()

min\_date\_employees = []

for row in reader:

row\_date = datetime.datetime.strptime(row[3], '%Y-%m-%d')

# If this date is smaller than the one we're looking for,

# we skip this row

if row\_date < start\_date:

continue

# If this date is smaller than the current minimum,

# we pick it as the new minimum, resetting the list of

# employees at the minimal date.

if row\_date < min\_date:

min\_date = row\_date

min\_date\_employees = []

# If this date is the same as the current minimum,

# we add the employee in this row to the list of

# employees at the minimal date.

if row\_date == min\_date:

min\_date\_employees.append("{} {}".format(row[0], row[1]))

return min\_date, min\_date\_employees

def list\_newer(start\_date):

while start\_date < datetime.datetime.today():

start\_date, employees = get\_same\_or\_newer(start\_date)

print("Started on {}: {}".format(start\_date.strftime("%b %d, %Y"), employees))

# Now move the date to the next one

start\_date = start\_date + datetime.timedelta(days=1)

def main():

start\_date = get\_start\_date()

list\_newer(start\_date)

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

main()

Q2 : start\_date\_report.py

#!/usr/bin/env python3

import csv

import datetime

import requests

FILE\_URL="http://marga.com.ar/employees-with-date.csv"

def get\_start\_date():

    """Interactively get the start date to query for."""

    print()

    print('Getting the first start date to query for.')

    print()

    print('The date must be greater than Jan 1st, 2018')

    year = int(input('Enter a value for the year: '))

    month = int(input('Enter a value for the month: '))

    day = int(input('Enter a value for the day: '))

    print()

    return datetime.datetime(year, month, day)

def get\_file\_lines(url):

    """Returns the lines contained in the file at the given URL"""

    # Download the file over the internet

    response = requests.get(url, stream=True)

    # Decode all lines into strings

    lines = []

    for line in response.iter\_lines():

        lines.append(line.decode("UTF-8"))

    return lines

def get\_same\_or\_newer(start\_date, data):

    reader = csv.reader(data[1:])

    min\_date = datetime.datetime.today()

    min\_date\_employees = []

    for row in reader:

        row\_date = datetime.datetime.strptime(row[3], '%Y-%m-%d')

        if row\_date < start\_date:

            continue

        if row\_date < min\_date:

            min\_date = row\_date

            min\_date\_employees = []

        if row\_date == min\_date:

            min\_date\_employees.append("{} {}".format(row[0], row[1]))

    return min\_date, min\_date\_employees

def list\_newer(start\_date, data):

    while start\_date < datetime.datetime.today():

        start\_date, employees = get\_same\_or\_newer(start\_date, data)

        print("Started on {}: {}".format(start\_date.strftime("%b %d, %Y"), employees))

        # Now move the date to the next one

        start\_date = start\_date + datetime.timedelta(days=1)

def main():

    data = get\_file\_lines(FILE\_URL) #Download only once at start

    start\_date = get\_start\_date() #get the date

    list\_newer(start\_date, data) #send the date and file

if \_\_name\_\_ == "\_\_main\_\_":

    main()