COMP 204: more File IOs (Assignment3)
A bus simulation project

Yue Li
based on material from Mathieu Blanchette

Outline

Midterm thoughts

More File IO

Assignment 3

A bus simulation program

Outline

Midterm thoughts

More File IO

Assignment 3

A bus simulation program

Midterm

- 20% midterm (besides 35% assignment, 10% quiz, 35% final)
- - Multiple choices (5 questions, 20 points) (4% of total grade)
  - Short answer (10 questions, 40 points) (8% of total grade)
  - **Programming questions (4 questions, 40 points) (8% of total grade)**
- The main issue: not enough time to complete the programming questions (0.4 * 20% = 8%)

Alternative grading options (completely optional):

1. max(20% midterm + 35% final, 55% final)
2. 12% midterm (4% MC + 8% short answer), make the 8% programming questions as **another assignment posted on February 26 12:00 AM (i.e., today at midnight)** and **due on this Friday March 1 23:59**
   For option 2, students who do not submit the midterm programming questions by Friday March 1 23:59 will be marked based on what they wrote in the midterm (i.e., 8% completely marked based on your written answers to the 4 programming questions). *No late submission accepted*.

Outline

Midterm thoughts

More File IO

Assignment 3

A bus simulation program

JSON module

Strings can easily be written to and read from a file

Numbers take a bit more effort to read/write

- the `read()` method only returns strings, so we need to convert them to integers using `int()`
- the `write()` method accepts strings as arguments, so we need to covert numbers to strings before writing them.

Also: What if you want to save more complex data types like nested lists or dictionaries?

- parsing and serializing by hand becomes complicated
- serializing: converting an object to a string that allows the object and state to be more easily recreated

Serializing objects with JSON

Rather than having users constantly write code to read/write complex data, Python allows you to use the popular data interchange format called **JSON (JavaScript Object Notation)**

`json.dump()` serializes an object to a text file

`json.load()` loads serialized object from text file

```
1  import json
2
3  outfile = "/Users/yueli/Lectures/20/my_file.json"
4  some_data = [1, 'simple', {'Yue':2.0,'Maria':3.0}]
5  f = open(outfile,"w")
6  json.dump(some_data,f) # write object into json file
7  f.close()
8
9  f = open(outfile,"r") # load object from json file
10 my_data = json.load(f)  # some_data is a list
11 print(my_data) # [1, 'simple', {'Yue': 2.0, 'Maria':
   ↪  3.0}]
12 f.close()
```

Reading/writing gzip compressed files

gzip.open() provides an interface to read/write compressed files

- gzip files save *a lot of* disk space (e.g., DIAGNOSES_ICD.csv (18M) vs DIAGNOSES_ICD.csv.gz (4.5M)) (651,048 rows)
- files typically end with the '.gz' extension
- available modes: r, a, and w along with binary options

```
1  import gzip
2
3  # a comma separated value (csv) file
4  gzfile = "/Users/yueli/Lectures/20/DIAGNOSES_ICD.csv.gz"
5  f = gzip.open(gzfile, "r")
6
7  #  .decode() converts bytes to string
8  line = f.readline().decode("utf-8")
9  print(line.rstrip()) #
   ↪  "ROW_ID","SUBJECT_ID","HADM_ID","SEQ_NUM","ICD9_CODE"
10 line = f.readline().decode("utf-8")
11 print(line.rstrip()) # 243,34,115799,8,"E8790"
12
13 f.close()
```

Reading a csv file using `pandas.read_csv()` function

`pandas.read_csv` provides an easy way to read comma-separated value (csv) file as a `DataFrame` object

```
1  import pandas as pd
2
3  filename = "/Users/yueli/Lectures/20/DIAGNOSES_ICD.csv.gz"
4
5  patient_data = pd.read_csv(filename, compression="gzip")
6
7  patient_records = {} # save patient ICD-9 code into dictionary
8  for index,row in patient_data.iterrows(): # iterate row by row
9
10     patId = row['SUBJECT_ID'] # access column "SUBJECT_ID"
11     icd9_code = row['ICD9_CODE'] # access column "ICD9_CODE"
12
13     patient_records.setdefault(patId, []).append(icd9_code)
14
15     if index > 100: # iterate only the first 100 rows
16         break
17
18 for k,x in patient_records.items():
19     print(k, x ,sep='\t')
```

Click `read_csv()` and DataFrame for more info.

Outline

Midterm thoughts

More File IO

Assignment 3

A bus simulation program

International Classification of Diseases 9^{th} Ed (ICD-9)

ICD-9 definition was downloaded from here. There are 157 ICD-9 groups and 1234 ICD-9 codes.

Intestinal infectious diseases (001-009)
001 Cholera
002 Typhoid and paratyphoid fevers
003 Other salmonella infections
004 Shigellosis
005 Other food poisoning (bacterial)
006 Amebiasis
007 Other protozoal intestinal diseases
008 Intestinal infections due to other organisms
009 Ill-defined intestinal infections
Tuberculosis (010-018)
010 Primary tuberculous infection
011 Pulmonary tuberculosis
012 Other respiratory tuberculosis

Q1: Store ICD-9 in a dictionary with key as group names and values as a `set` of ICD-9 codes

group code Intestinal infectious diseases (001-009)
001 Cholera
002 Typhoid and paratyphoid fevers
003 Other salmonella infections
004 Shigellosis
⋮

⋮

group name Genetics (V83-V84)
V83 Genetic carrier status
V84 Genetic susceptibility to malignant neoplasm
group name Body mass index (V85)
V85 Body mass index
group name Estrogen receptor status (V86)
V86 Estrogen receptor status
group name Other specified personal exposures and history presenting hazards to health (V87)
V87 Other specified personal exposures and history presenting hazards to health

MIMIC-III (Medical Information Mart for Intensive Care)

"ROW_ID","**"SUBJECT_ID"**","HADM_ID","SEQ_NUM","**"ICD9_CODE"**
243,34,115799,8,"E8790"
244,34,144319,1,"42789"
245,34,144319,2,"42822"
246,34,144319,3,"4263"
247,34,144319,4,"41401"
248,34,144319,5,"V5861"
249,34,144319,6,"4280"
250,34,144319,7,"2449"
251,34,144319,8,"3659"
252,35,166707,1,"3962"
253,35,166707,2,"4260"
254,35,166707,3,"2875"

Q2: building a patient dictionary with key as patientID and values as ICD-9 codes

```
34 {'426', '244', '414', 'E879', '427', '997', '410', '425', '365', '428', 'V58'}
35 {'426', '250', '244', '414', '427', '997', '287', '396', '715', '401'}
36 {'453', '411', 'V10', '401', '996', '415', '486', '997', '596', '496', '414', '998', 'V45',
'300', '553', '518', '305', '530', '600'}
37 {'285', '496', '250', '414', '427', '486', '410', '535', '428'}
38 {'995', '584', '414', '427', '997', '998', 'E870', '608', '560', 'V45', '428', '038'}
39 {'V05', 'V29', '765', 'V30', '770', '769', '774', '276'}
41 {'348', '518', '285', '507', '482', '513', '496', '512', '191', '401', '305', '709', '707',
'V10', '112', '788'}
42 {'414', '427', '438', '412', '428', '401'}
43 {'E821', '518', '805'}
44 {'780', '414', '427', '272', '596', '443', '396', '413', '401', 'V17'}
45 {'911', '305', 'E812', '873', '913', '998', '824'}
:
```

Q4-Q7: Making diagnosis by predicting ICD-9 code

1. For a test patient, find k closely matched patients based on the similarity of their ICD-9 code groups
2. Calculate the frequency of the specific ICD-9 codes among the closely matched patients. The unrecorded ICD-9 codes with high frequencies for the test patient are considered as the highly plausible codes that might have been missed during the diagnosis process.

patId	symptoms_status	icd9 group	ICD9 code	ICD9 name	Frequency
71	observed	285		Other and unspecified anemias	1
71	observed	266		Deficiency of B-complex components	1
71	observed	E950		Suicide and self-inflicted poisoning by solid or liquid sub.	1
71	observed	295		Schizophrenic psychoses	1
71	observed	969		Poisoning by psychotropic agents	1
71	observed	276		Disorders of fluid, electrolyte, and acid-base balance	1
71	predicted	296		Affective psychoses	0.55
71	predicted	965		Poisoning by analgesics, antipyretics, and antirheumatics	0.4
71	predicted	305		Nondependent abuse of drugs	0.35

Outline

Midterm thoughts

More File IO

Assignment 3

A bus simulation program

Computer Simulations

A computer simulation attempts to recreate virtually a system of interest and its evolution.

- We can simulate the progression of a flu virus in a population
- Evolution of an ecosystem subject to climate change
- Weather systems
- etc.

Purposes: Study how the system evolves over time; evaluate the impact of changes in conditions, etc.

A bus line simulation

Bus stations:	0: St-Denis	1: Hotel-de-ville	2: St-laurent	3: St-Urbain	4: Parc	5: University
Initial state Buses:						
Customers waiting:	[1,3,4,3,5,5,5,4,2,5,5]	[2,4,4,3,4,4,4,2,4,4,3,4,4,4,2]	[4,5,5,3]	[5,5,4,5,5,5,4,5,4]	[5,5,5,5,5,5]	[]
Time: 0 Buses:	bus_time0:[1,3,4,3,5]					
Customers waiting:	[5,5,4,2,5,5]	[2,4,4,3,4,4,4,2,4,4,3,4,4,4,2]	[4,5,5,3]	[5,5,4,5,5,5,4,5,4]	[5,5,5,5,5,5]	[]
Time: 1 Buses:		bus_time0:[3,4,3,5,2]				
Customers waiting:	[5,5,5,4,2,5,5]	[4,4,3,4,4,4,2,4,4,3,4,4,4,2]	[4,5,5,3]	[5,5,4,5,5,5,4,5,4]	[5,5,5,5,5,5]	[]
Time: 2 Buses:	bus_time2:[1,3,4,3,5]		bus_time0:[3,4,3,5,4]			
Customers waiting:	[5,5,4,2,5,5]	[4,4,3,4,4,4,2,4,4,3,4,4,4,2]	[5,5,3]	[5,5,4,5,5,5,4,5,4]	[5,5,5,5,5,5]	[]
Time: 3 Buses:		bus_time2:[3,4,3,5,4]		bus_time0:[4,5,4,5,5]		
Customers waiting:	[5,5,4,2,5,5]	[4,3,4,4,4,2,4,4,3,4,4,4,2]	[5,5,3]	[4,5,5,5,4,5,4]	[5,5,5,5,5,5]	[]
Time: 4 Buses:	bus_time4:[5,5,4,2,5]		bus_time2:[3,4,3,5,4]		bus_time0:[5,5,5,5,5]	
Customers waiting:	[5]	[4,3,4,4,4,2,4,4,3,4,4,4,2]	[5,5,3]	[4,5,5,5,4,5,4]	[5,5,5,5]	[]
Time: 4 Buses:		bus_time4:[5,5,4,2,5]		bus_time2:[4,5,4,4,5]		bus_time0:[]
Customers waiting:	[5]	[4,3,4,4,4,2,4,4,3,4,4,4,2]	[5,5,3]	[5,5,4,5,4]	[5,5,5,5]	[]

A bus line simulation

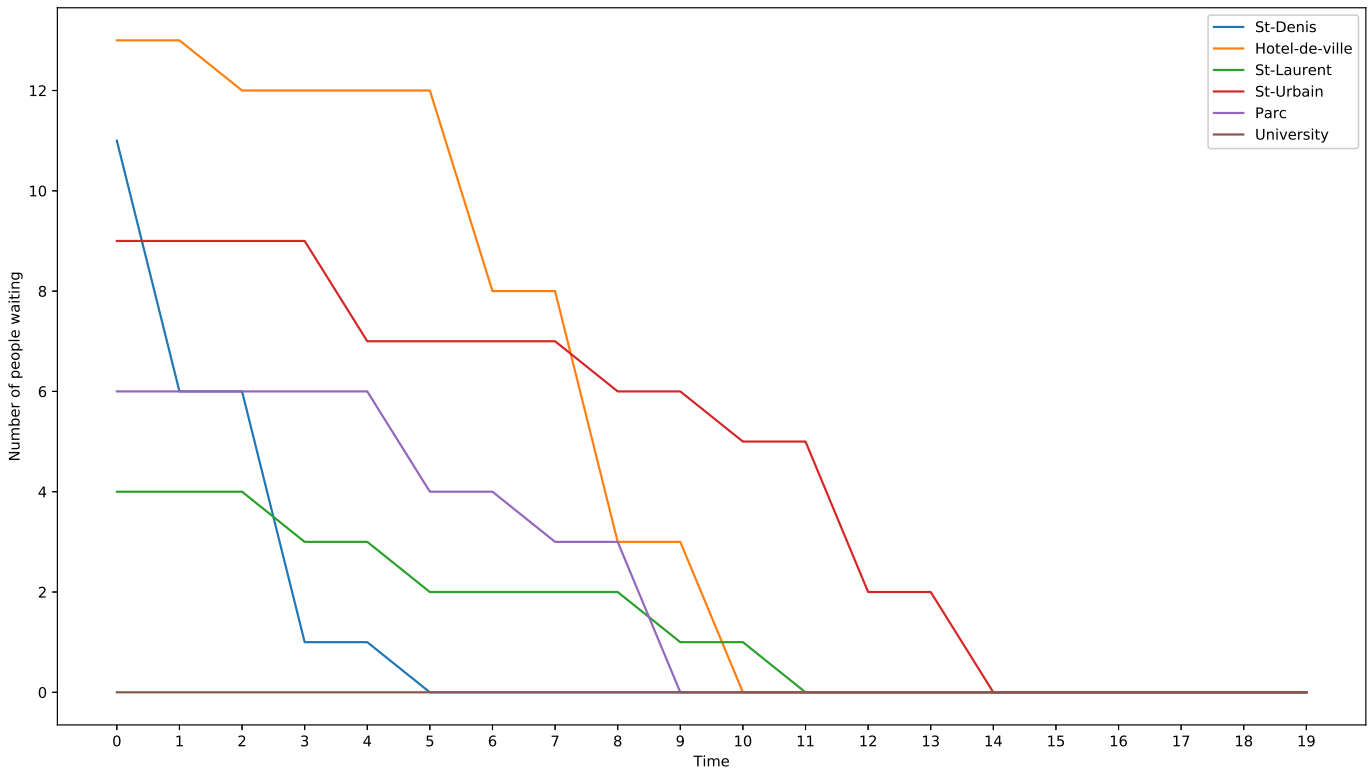
Goal: Simulate a bus line, with buses running along it, people waiting at bus stops, and

- with buses running along it,
- people waiting at bus stops,
- people boarding
- disembarking the bus

Component of a simulation system

1. A description of the state of the system:
   - A list an ordered of bus stations:
     `names_of_stations={stationID:stationName}`
   - Position of each bus:
     `base_positions={busID:stationID}`
   - Capacity of each bus: `bus_capacity=5`
   - List of people on board of each bus, with their intended destination:
     `bus_content = {busID:[customer_destiations]}`
   - List of
     people waiting at each bus stop, with their intended destination:
     `waiting_at_stop = {stationID:[customer_destiations]}`
   - Time: `range(0,simulation_duration)`
2. A set of rules describing how the system evolves
   - New bus shows up at first station every 2 minutes
   - Buses move from one station to the next in one minute
   - At each station, people who want to get off discharge
   - The empty spots on the bus get filled by the first people in the line, up to capacity

Goal 1 - track queues at each station



Goal 2 - track arrivals at each station