Computer Science 2020-21
Exercises III:
Files, sets, tuples and dictionaries

Teaching Assistant: ing. Gian Enrico Conti
10047232@polimi.it
18th Nov, 2020

Ex3.1(warm-up): read file

Prepare on you PC in the SAME directory of our python program a file composed by multiple integer values, comma separated like that:

120,230,100,400,500 called **samples.csv**

Write a program that reads these values and outputs the sum of all values, max, and min found,

or False if no data in file.

Ex3.1: read file: Solution


```
FNAME = "samples.csv"
f=open(FNAME)=
lines = f.read().splitlines() # prevent be user has typed return.
lineCount = len(lines)
if lineCount > 0:
    line = lines[0]     #take first only
    elementsList = line.split(',')
    elementsCount = len(elementsList)
    sum = 0
    if elementsCount>0:
        max = int(elementsList[0])
        min = int(elementsList[0])
        for i in range(1, elementsCount):
            value = int(elementsList[i])
            sum += value
            if value >max:
                max = value
            if value < min:
                min = value
        print(sum, max, min)
    else:
        print(False)

else:
    print(False)
```

Ex3.2: read pressure

Get from "Beep" of course (ex3 -> materials)
The file **patient_samples.csv** and copy in directory of our python program.

The file has structure:

Smith	**80**	**90**	**90**	**95**	**100**
Cooper	80	90	90	95	120

Where for every patients, we have surname and 5 values of blood pressure.

Write a program that reads data and prints patient with highest average pressure.

Ex3.2: read pressure: Solution


```
FNAME = "patient_samples.csv"
f=open(FNAME)
lines = f.read().splitlines()

numOfPatients = len(lines)
if numOfPatients > 0:
    maxAveragePressure = 0
    indexOfPatient = 0
    for i in range(len(lines)):
        line = lines[i]
        elements = line.split(',')
        elementsCount = len(elements)
        if elementsCount > 0:
            pressures =  list(map(int, elements[1:]))  #take from 2nd AND convert ('map')
from string to int
            #print(pressures)
            rowAvg = sum(pressures) / (elementsCount-1)
            if rowAvg>maxAveragePressure:
                maxAveragePressure = rowAvg
                indexOfPatient = i


    surname = lines[indexOfPatient].split(',')[0]
    print(surname)
else:
    print(False)
```

Ex3.3: intro

On WHO site:

- files (https://www.who.int/tb/country/data/download/en/)

- Terminology used (https://extranet.who.int/tme/generateCSV.asp?ds=dictionary)

Download: "WHO TB incidence estimates disaggregated by...."

https://extranet.who.int/tme/generateCSV.asp?ds=estimates_age_sex

(File is available on Beep, too)

Ex3.3: intro

File contains:

TB_burden_age_sex_2020-11-05

country	iso2	iso3	iso_numeric	year	measure	unit	age_group	sex	risk_factor	best	lo	hi
Afghanistan	AF	AFG	4	2019	inc	num	0-14	a	all	15000	7900	22000
Afghanistan	AF	AFG	4	2019	inc	num	0-14	f	all	7300	2400	12000
Afghanistan	AF	AFG	4	2019	inc	num	0-14	m	all	7800	2600	13000

Ex3.3 Sets

After copying "TB_burden_age_sex_2020-11-05.csv" in your working directory,

write a python program that asks user 2 iso codes, reads that file and outputs them sum of all values in "**best**" column for female sex,

or False if no data in file.

Ex3.3


```
iso1 = str(input('1st ISO code '))
iso2 = str(input('2nd ISO code '))

isoSet = {iso1, iso2}  # build a set.
FNAME = "TB_burden_age_sex_2020-11-05.csv"
#FNAME = "TB_burden_reduced.csv"
f=open(FNAME)
lines = f.read().splitlines()
nOfLines = len(lines)
if nOfLines > 0:
    sum = 0
    titleRow = lines[0].replace('"', '')
    columnsTitles = titleRow.split(',')
    columnOfBest =  columnsTitles.index('best')
    columnOfISOCode = columnsTitles.index('iso2')
    columnOfSex = columnsTitles.index('sex')
    for i in range(1, len(lines)):
        line = lines[i].replace('"', '')
        columnsValues = line.split(',')
        a = columnsValues[columnOfISOCode]
        b = columnsValues[columnOfSex]

        if columnsValues[columnOfISOCode] in isoSet and columnsValues[columnOfSex] == 'f':
            sum = sum + int(columnsValues[columnOfBest])
    print(sum)
else:
    print(False)
```

Ex3.4 Tuples

Get from "Beep" of course (ex3 -> materials)
The file **visits.csv** and copy in directory of your python program.

The file has structure:
(TAB delmited)

visits

surname	date
Smith	2020-11-03
Johnson	2020-11-03
Williams	2020-11-04
Brown	2020-11-04

Write a program that reads data and prints patients with no duplicates, with their first visit.

Ex3.4 Tuples


```
FNAME = "visits.txt"  # TAB delimited

surnames = ()       # empty tuple (dont write:  = {} ...
firstVisits = []    # will be array of tuples

# protect against file is missing using "try":
try:
    #go line by line to save memory
    f = open(FNAME)
    next(f) #skip first
    for line in f:
        cleanedLline = line.rstrip('\r\n') # strip out all tailing whitespace
        elems = cleanedLline.split('\t')
        #print(elems)
        surname = elems[0]
        date = elems[1]
        if surname not in surnames:
            surnames = surnames + (surname, ) # tuples are immutable, create new ONE.
            # note sintax-pyton 3
            #surnames = (*surnames, surname)
            t = (surname, date)
            firstVisits.append(tuple(t))


    for v in sorted(firstVisits):
        print(v)
except:
    print(False)
```

Ex3.5 Dict

After copying "patients.csv"

```
"name","surname", "Hgmm","ward"
"John","Doe",670, "Cardiology"
"Bill","Black",930, "Cardiology"
"Mark","Smith", 870,"Neurology"
"Sarah","Johnes",920, "Nephrology"
```

in your working directory,

write a python program that reads that file and outputs the patients with "Hgmm" > 680 of wards "Cardiology" and "Neurology"

or False if no data in file.

Ex3.5 Dict


```
import csv

FNAME = "patients.csv"
#FNAME = "sample.csv"

SELECTED_WARDS = {"Cardiology", "Neurology"}

# protect against file is missing using "try":
try:
    f = open(FNAME)
    csv_reader = csv.DictReader(f, skipinitialspace = True)

    list_of_patients = list(csv_reader) # we got a list of dict

    for patient in list_of_patients:
        Hgmm = int(patient["Hgmm"])
        ward = patient["ward"]
        if Hgmm>680 and ward in SELECTED_WARDS:
            print(patient["name"], patient["surname"])
except:
    print(False)
```

Ex3.6 Dict

After copying "TB_burden_age_sex_2020-11-05.csv" in your working directory,

write a python program that reads that file and outputs the countries with "**lo**" value >100 for risk factors read from another file "risks.txt" containing "hiv" and "alc" on 2 lines.

Countries must be printed ONCE and alphabetically.

Ex3.6 Dict


```
import csv

FNAME = "TB_burden_age_sex_2020-11-05.csv"
RISK_FNAME = "risks.txt"

with open(RISK_FNAME) as f_risk:
    lines = f_risk.read().splitlines()
    riskfactors = set(lines)  # build a set.

# protect against file is missing using "with":
with open(FNAME) as f:
    countries = set()
    csv_reader = csv.DictReader(f)
    list_of_rows = list(csv_reader)
    for row in list_of_rows:
        #print(row)
        lo = row["lo"]
        if lo.isdigit() and int(row["lo"])>100 : #some columns have NO data, so digit
            countries.add(row["country"])

    for c in sorted(countries):
       print(c)
```

Ex3.7 Dict

After downloading (https://ftp.ncbi.nih.gov/refseq/H_sapiens/RefSeqGene/refseqgene.5.genomic.fna.gz), expanding and copying "refseqgene.5.genomic.fna" in your working directory,

(structure is:
>NG_023443.2 Homo sapiens eyes shut homolog (EYS), RefSeqGene on chromosome 6
TGAAAATGCCTGTAGTCCAGTGTTCTAAATATCTGTCTGCAGTATATGGCACAGATTATCATTCCCTTCTTGAAATGTTT
..)

write a python program that
- the reads that file,
- find every "RefSeqGene",
- and outputs for every block, how many DNA Elems are present, in ascending order on name.

Ex3.7 Dict


```
ADENINE = 'A'
CYTOSINE = 'C'
GUANINE = 'G'
THYMINE = 'T'
DNAELEM = ADENINE + CYTOSINE + GUANINE + THYMINE

FNAME = "refseqgene.5.genomic.fna"
#FNAME = "refseqgene.5.genomic_small.fna"

blocks = []
DNACount = 0
blockName = ""

f = open(FNAME)
# go line by line to save memory
for line in f:
    cleanedLline = line.rstrip('\r\n')  # strip out all tailing whitespace
    if cleanedLline.startswith('>NG_'):
        #close previous block if present:
        if len(blockName)>0:
            block = {'name': blockName, 'DNACount': DNACount}
            blocks.append(block)
            #and clear: (we saved)
            DNACount = 0
        #now the new block name:
        blockName = cleanedLline[13:]
    else: # line with DNA
        inRow = + cleanedLline.count(DNAELEM)
       DNACount += inRow

#if here with name and count, add:
if len(blockName)>0:
   block = {'name': blockName, 'DNACount': DNACount}
   blocks.append(block)
for b in  sorted(blocks, key=lambda k: k['name']) :
    print( '{:0>4}'.format( b["DNACount"] ), b["name"])
```