ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Introduction to Information Systems and Applications

Course Unit 2: Data processing with python

M. Tzagarakis, V. Daskalou

School of Business Administration

Department of Economics

Functions and Exceptions

Functions

Use your own functions when you are writing the same snippet of code over and over again

definition -> `def function_name(parameter):`

`return` -> used to return the value

call: `function_name(parameter)`

```
#defines a functions that returns average of nums
def calc_avg(num_values):
        if len(num_values)>0:
                return sum(num_values)/len(num_values)
        else:
                return float('nan')
lista = [3,4,5,6,6,7,8,9]
print("Outfunc-1: Average: {0:.2f}".format(calc_avg(lista)))
```

Exceptions

- Robust code must handle errors and exceptions effectively, e.g. program does not throw and error with a division with zero
- Use try-exception block to catch and handle exceptions

```
#defines a functions that returns average of nums
def calc_avg(num_values):
               return sum(num_values)/len(num_values)
listb=[]
#handles division by zero exception
try:
    print("Outfunc-1: Average: {0:.2f}".format(calc_avg(listb)))
except ZeroDivisionError as detail:
    print("Outfunc-1: (Error): {}".format(float('nan')))
    print("Outfunc-1: (Error): {}".format(detail))
```

Files

Datasets in text files

We can use python to process open data in formats like:

- Text files: A text file contains human-readable characters and each line of text is terminated with a special character known as EOL (End of Line) character.
  - Text files use type .txt
- CSV (Comma Separated Value) text files: use “,” or “;” as delimiter between columns, or
  TSV (Tab Separated Value) text files: use tab as delimiter
  - CSV files use type .csv and TSV type .tsv

A user can read the contents of a text file or edit it using a text editor (such as MS notepad)

Read a text file (1)

```
with open("animals.txt", mode="r",encoding="utf-8") as my_file:
       for line in my_file:
           print(line)
```

Improvement: `print(line.rstrip("\n"))` or `print(line, end='')`

Content of file animals.txt

```
antelope
bear
cat
dog
elephant
fox
```

`open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)`

- `file`: file name
- `mode`:
  - **w** – open file for writing (deletes previous content)
  - **r** – open file for reading
  - **a** – open file for writing and append to existing content
- `encoding`: for text file by different countries we use UTF8

Read a text file (2)

- `myfile.read()  ->` reads all file lines in one string
  ```
  'antelope\nbear\ncat\ndog\nelephant\nfox‘
  ```

- `myfile.readlines()  ->` reads all lines in a list
  ```
  ['antelope', 'bear', 'cat', 'dog', 'elephant', 'fox']
  ```

- `myfile.read().splitlines()->` reads all lines in a string and separates it in lines
  ```
  ['antelope\n', 'bear\n', 'cat\n', 'dog\n', 'elephant\n',
  'fox']
  ```

- `for line in myfile` -> reads each line one-by-one in the variable `line`

Read CSV files

Centent of file party_guests.csv

```
"name";"whose";"adults";"children"
"Anne";"Mum’s";2;3
"Bod";"Dad’s";1;1
"Susan";"Mine";2;2
"Marek";"Brother’s";2;0
"Mairy";"Mum’s";2;0
```

```
import csv
with open('party_guests.csv', mode='r', newline='',encoding='utf-8') as f:
        reader=csv.reader(f,delimiter=';',quotechar='"', quoting=csv.QUOTE_NONNUMERIC)
        header=next(reader)
        print(header)
        for row in reader:
                print(row)
```

1. Use csv module to manage CSV files
2. Opens csv with `open()`
3. Use `csv.reader()` to read from file. Define `delimiter`, `quotechar` and that non-quoted will be floats
4. Use `for in` to access each line one-by-one
5. Each row is a list and the value in each column in the list is a list member

Read from CSV file: find sums of columns

```
import csv
names=[]
whose_friends=[]
adults=[]
childrens=[]
with open('party_guests.csv', mode='r', newline='',encoding='utf-8',) as f:
    reader=csv.reader(f,delimiter=';',quotechar='"', quoting=csv.QUOTE_NONNUMERIC)
    header=next(reader)
    for row in reader:
          #members of list row are assigned to variables on the left
          #name,whose,nadult,nchildren=row
          name=row[0]
          whose=row[1]
          nadult=row[2]
          nchildren=row[3]
          names.append(name)
          whose_friends.append(whose)
          adults.append(nadult)
          childrens.append(nchildren)
print(sum(adults))
print(sum(childrens))
```

Write to a CSV file

Contents of output file myOutput.csv

```
"name";"whose";"adults";"children"
"Anne";"Mum’s";2;3
"Bod";"Dad’s";1;1
"Susan";"Mine";2;2
"Marek";"Brother’s";2;0
"Mairy";"Mum’s";2;0
```

```
import csv
data = [['name','whose','adults','children'],\
['Anne','Mum’s',2.0,3.0],\
['Bod',"Dad's",1,1]]
with open('myOutput.csv',mode='w', newline='') as myWFile:
    writer = csv.writer(myWFile, delimiter=';',quotechar='"',quoting=csv.QUOTE_NONNUMERIC)
    for line in data:
        writer.writerow(line)
```

1. Open output file with `open()` but with `mode=w` (write)
2. Use `csv.writer()` to write to output file. Define `delimiter`, `quotechar` and that non-quoted chars with be numeric
3. Use `for in` to access each list in output data list (variable `data` is a list of lists)
4. Use method `writerow()` of writer object to write a line in the output file with the specified format

Read from csv to dictionary (1)

```
# module cvs has functions to manage csv files
import csv
mguests=0
with open('party_guests.csv', mode='r', newline='',encoding='utf-8') as f:
    # reads lines in dictionary with keys the header names
    reader=csv.DictReader(f,delimiter=';',quotechar='"', quoting=csv.QUOTE_NONNUMERIC)
    # handles exceptions with try-except
    try:
        for row in reader:
            if row['whose']=="Mum’s":
                print("Line: {!s}".format(row))
                mguests=mguests+row['adults']+row['children']
        print("Mum has {0} guests".format(mguests))
    except csv.Error as e:
            sys.exit('file %s, line %d: %s' % (filename, reader.line_num, e))
```

Read from csv to dictionary (2)

New issues:

- Use of `csv.DictReader:` Reads in an ordered dictionary each line:

```
OrderedDict([('name', 'Anne'), ('whose', 'Mum’s'),
('adults', 2.0), ('children', 3.0)])
```

- Use `try-except` to handle errors