Abertay University

HCS501 – Lecture 6

Numerical Processing and Libraries

Data Files (1)

- Recall that text files offer a simple way to store data records:
  - one line of text per record.
  - data are delimited field.
  - string method **split()** is useful.
- The CSV file format is popular:
  - Comma Separated Values (may use spaces).
  - usually uniform number of fields per record.
  - “free-form” text fields often quoted.
  - often one or more header lines e.g. field labels.

tempLog.csv

```
Timestamp,Air,CPU
1634036226,21.69,31.95
1634036406,21.67,32.07
1634036586,21.75,32.16
...
```

Data Files (2)

- Examining the contents of ‘tempLog.csv’ we see:
  - a single header line of field labels.
  - uniform record length of three fields.
  - data are comma delimited, without spaces.
  - no units of measurement or detailed description, just numbers.

```
Timestamp,Air,CPU
1634036226,21.69,31.95
1634036406,21.67,32.07
1634036586,21.75,32.16
...
```

Data Files (3)

- Interpreting ‘tempLog.csv’:
  - looks like a series of temperature values (in degrees Celsius?) recorded over time.
- Timestamp?
  - a “Unix timestamp” measures seconds since 1970 Jan 1st 00:00 (UTC).
  - the Python datetime module is useful:

```
Timestamp,Air,CPU
1634036226,21.69,31.95
1634036406,21.67,32.07
1634036586,21.75,32.16
...
```

```
>>> from datetime import datetime as dt
>>> print(dt.fromtimestamp(1634036226))
2021-10-12 10:57:06
```

Data Files (4)

- Processing ‘tempLog.csv’ with Python:
  - the simple CSV structure does not present any particular problems.

```
with open('tempLog.csv') as f:
    hdr= f.readline() # skip header
    nextRecord= True
    while nextRecord:
        recLine= f.readline()
        fields= recLine.split(',')
        timestamp= int(fields[0])
        air= float(fields[1])
        cpu= float(fields[2])
        # process data here
        if '' == recLine: # EOF
            nextRecord= False
```

Data Files (5)

- The Python CSV library offers an alternative:
  - slightly shorter.
  - the **csv.reader()** method is very robust (handles complex formatting, identifies errors).
  - many extra parameters available to configure the ‘dialect’ of CSV file.

```
import csv
with open('tempLog.csv') as f:
    # create an iterable object
    lineReader= csv.reader(f)
    hdr= next(lineReader) # skip
    for rec in lineReader:
        timestamp= int(rec[0])
        air= float(rec[1])
        cpu= float(rec[2])
```

External Libraries (1)

- Many third-party libraries are available:
  - internet, graphics, data analysis, machine learning…
  - Python makes very sophisticated software easy to use.
  - library **packages** are available from **repositories** using a **package manager**.
- Linux users may be familiar with package management:
  - for stable (older) library versions try: `> apt list python3*`
- Microsoft Visual Studio has menu options for library management.

External Libraries (2)

- Python has its own package manager **pip**:
  - IDLE uses this during installation, but doesn't make it available within the RTE or from menus – you have to use the command line (e.g. Terminal or MS-Win PowerShell).
  - remember that Python2 remains the default on many systems so it is advisable to specify Python3 using the "launcher command" **py**.

```
> py -3 –m pip install <libraryname>
```

- Further guidance on pip:

https://pip.pypa.io/en/latest/installation

External Libraries (3)

- For MS-Windows users who encounter intractable problems with any of the previous methods:
  - it is possible to manually download Windows compatible Python libraries.
  - this is undesirable for security reasons and very much a last resort.
  - read instructions (disclaimers) carefully, take warnings seriously.
  - back up all your work and remember: “if in doubt, miss it out”.

https://www.lfd.uci.edu/~gohlke/pythonlibs

numpy (1)

- A numerical library, based on **array** representation:
  - somewhat similar to MatLab® and some other maths software.
  - an array is a table of data arranged in rows and columns.
  - the type **ndarray** (n-dimensional array) is provided by numpy.
  - number of **elements** must be uniform (every row same length).
  - numerical algorithms supported by numpy are the basis for image processing, machine learning and other advanced computation.

1	2	3
-2	1	2
-3	-2	1

```
> py -3 –m pip install numpy
```

- Python doesn't have its own array type:
  - but a **list of lists** can do the same job (albeit much slower).

```
[ [1,2,3],
  [-2,1,2],
  [-3,-2,1] ]
```

numpy (2)

- Only numerical data is allowed:
  - hence csv file format is not automatically supported.
  - instead the very flexible **loadtxt()** method can be used.
  - there are many parameters with default values.
  - Python can replace a default using the **formal parameter** name (i.e. in the method or function definition).
  - formal parameter name is assigned actual parameter value (expression).

```
>>> import numpy
>>> array= numpy.loadtxt('tempLog.csv',delimiter=',',skiprows=1)
```

  - default delimiter is whitespace.
  - need to skip the first line containing text labels.

numpy (3)

- Loads 2D (two dimensional) array:
  - floating point numbers (IEEE-754 binary64 “double”) are the default.
  - printed using “list of list” notation.

```
>>> import numpy
>>> array= numpy.loadtxt('tempLog.csv',delimiter=',',skiprows=1)
>>> print(array)
[[1.63403623e+09 2.16900000e+01 3.19500000e+01]
[1.63403641e+09 2.16700000e+01 3.20700000e+01]
[1.63403659e+09 2.17500000e+01 3.21600000e+01]
...
[1.65822139e+09 2.70700000e+01 3.43200000e+01]
[1.65822157e+09 2.71300000e+01 3.44500000e+01]
[1.65822175e+09 2.71700000e+01 3.44900000e+01]]
```

numpy (4)

- Attributes:

```
>>> print(array.shape, array.dtype)
(128801,3) float64
```

- Accessing data is easy:

```
>>> print(array[0])    # first row
[1.63403623e+09 2.16900000e+01 3.19500000e+01]

>>> print(array[0,0])  # first element of first row
1634036226.0
```

numpy (5)

- Powerful control of functions:

```
>>> numpy.mean(array,axis=0)  # column-wise, full array
array([1.64618988e+09, 2.10541331e+01, 3.17532126e+01])

>>> numpy.mean(array[0:2],axis=0)     # column-wise, first three rows
array([1.63403632e+09, 2.16800000e+01, 3.20100000e+01])
```

- Key advantages:
  - data representation directly compatible with hardware (FPU/CPU).
  - high computational performance and memory efficiency.

PANDAS (1)

- Python data analysis library:
  - rather like a spreadsheet controlled via programming language.
  - uses **data frames** that are tables with rows and columns.
  - data elements can be anything, not just numbers.
  - no restrictions on layout.

```
> py -3 –m pip install pandas
```

PANDAS (2)

- Working with csv files is easy:
  - an index column is included within the data frame.

```
>>> import pandas
>>> df= pandas.read_csv('tempLog.csv')
>>> print(df)
          Timestamp    Air    CPU
0        1634036226  21.69  31.95
1        1634036406  21.67  32.07
2        1634036586  21.75  32.16
...
128798   1658221386  27.07  34.32
128799   1658221566  27.13  34.45
128800   1658221746  27.17  34.49
```

PANDAS (3)

- Accessing a range of rows:

```
>>> df.loc[0:2]
     Timestamp    Air    CPU
0   1634036226  21.69  31.95
1   1634036406  21.67  32.07
2   1634036586  21.75  32.16
```

PANDAS (3)

- Data labels (from file header) can be used for columns:

```
>>> df['Air'].mean()   # full column
21.054133120084472

>>> df['CPU'].mean()
31.75321263033672

>>> # average first 240 records of two columns
>>> df.loc[0:239,['Air','CPU']].mean()
Air    22.157958
CPU    32.842250
```

- Key advantage:
  - flexibility - more ‘Pythonic’ than numpy.