UNIVERSITÄT BIELEFELD
Faculty of Technology

Programming

Data Management & Analysis

Luna Pianesi

Faculty of Technology, Bielefeld University

UNIVERSITÄT BIELEFELD

Recap

- Code reuse through:
  - Functions
  - Classes
  - Modules
  - Packages
- Compile-time errors
- Runtime errors

Course syllabus

Part 1

- Programming basics and terminology
- Introduction to Python

Part 2

- Scientific Programming
- Data Science with Python

Course syllabus - where are we?

Part 1

- T0: how to use ChatGPT
- T1: programming and Python basics
- T2: data types and arithmetic operations
- T3: conditions, comparisons, and loops
- T4: functions and debugging
- T5: functional programming and lazy evaluation

Part 2

- T6: object-oriented programming
- ***T7: input, file processing, and text mining***
- T8: data visualization and NumPy
- T9: Pandas
- T10: machine learning
- T11: databases and distributed computing


File I/O
File Formats
Text mining
NLTK

What is file I/O? Input and output!

Just as a string object in Python is defined by the `str` class, a file object is defined by the `file` class.
Instances of the file class have several ***methods*** available for performing typical operations, such as *reading* and *writing*.

Source: `https://cvw.cac.cornell.edu/python-intro/input-output/file-io`

Interactive reading from console

- Reading a string from console:
  ```
  my_string = input()
  ```
- Specify prompt:
  ```
  year_str = input('What␣year␣were␣you␣born?')
  ```

Reading from command line

example_input_argument.py

```
1 #!/usr/bin/env python3
2 from sys import argv
3
4 if __name__ == '__main__':
5     my_arg1 = argv[1]
6     my_arg2 = argv[2]
7     print('1st input argument:', my_arg1)
8     print('2nd input argument:', my_arg2)
```

Reading from file

... like a book: open & read!

```
1 f = open('Frankenstein.txt')
2 my_text = f.read()
```

Reading from file

alternatively, use “with” statement:

```
1 with open('Frankenstein.txt') as f:
2     my_text = f.read()
```

Reading from file

read file line-by-line:

```
1 lines = list()
2 f = open('Frankenstein.txt',
3         newline='\n')
4 for line in f:
5     lines.append(line)
```

Reading from file

Dynamic: read from file with name requested by prompt

```
1 fName = input('Input␣file:␣')
2 lines = list()
3 f = open(fName)
4 for line in f:
5     lines.append(line)
```

Writing to file

… just as simple as reading!

```
1 f = open('letter1.txt', 'w')
2 f.write('TO␣Mrs.␣Saville,␣England')
3 f.flush()
```

Writing to file

`close()` flushes, then closes the file:

```
1 f = open('letter1.txt', 'w')
2 f.write('TO␣Mrs.␣Saville,␣England')
3 f.close()
```

Writing to file

`with` automatically closes the file:

```
1 with open('letter1.txt', 'w') as f:
2     f.write('TO␣Mrs.␣Saville,␣England')
```

Writing to file

Direct printout to file::

```
1 with open('output.txt', 'w') as f:
2     print('TO␣Mrs.␣Saville,␣England',
        file = f)
```

Quiz

- Command line arguments are received through the

  `input()` function `argv` list

- Complete the code for reading a file by filling in the blanks:

```
1 __________ open('myfile.txt') as __________:
2     contents = f.__________()
```

- Which function(s) invoke(s) the writing of file buffer data to the file system?

  `clear` `close` `write` **`buffer`** `flush`

source (in part): https://realpython.com/quizzes

Quiz

- Command line arguments are received through the

  `input()` function `argv` list ✔

- Complete the code for reading a file by filling in the blanks:

```
1 with open('myfile.txt') as f:
2     contents = f.read()
```

- Which function(s) invoke(s) the writing of file buffer data to the file system?

  `clear` `close`✔ `write` `buffer` `flush`✔

source (in part): https://realpython.com/quizzes


File I/O
File Formats
Text mining
NLTK

File formats

Unstructured data

- Plain text

Structured data

- XML
- JSON
- Tables
- Matrices

Unstructured data: plain text

... like a book: open & read!

```
1 f = open('Frankenstein.txt')
2 my_text = f.read()
```

Structured data: XML

EXtensible Markup Language: *a hierarchical data structure*

```
<book category="Python">
    <title lang="en">The Quick Python Book</title>
    <isbn>1884777740</isbn>
    <pageCount>444</pageCount>
    <publishedDate>
        <date>1999-10-01T00:00:00.000-0700</date>
    </publishedDate>,
    <authors>
        <author>Daryl Harms</author>
        <author>Kenneth McDonald</author>
    </author>
</book>
```

Structured data: JSON

JavaScript Object Notation: *similar to XML, but more compact*

```
{
    "title" : "The Quick Python Book",
    "isbn" : "1884777740",
    "pageCount" : 444,
    "publishedDate" : { "date" : "1999-10-01T00:00:00.000-0700" },
    "authors" : [ "Daryl Harms", "Kenneth McDonald" ],
    "categories" : [ "Python" ]
}
```

Structured Data: tables

Extract from file “books.tsv”

title	*isbn*	*pageCount*	*publishedDate*	*authors*	*categories*
Unlocking Android	1933988673	416	2009-04-01	W. Frank Ableson, Charlie Collins, Robi Sen	Open Source, Mobile
Specification by Example	1617290084	-	2011-06-03	Gojko Adzic	Software Engineering
Flex 4 in Action	1935182420	600	2010-11-15	Tariq Ahmed, Dan Orlando, John C. Bland II, Joel Hooks	Internet
Zend Framework in Action	1933988320	432	2008-12-01	Rob Allen, Nick Lo, Steven Brown	Web Development
Flex on Java	1933988797	265	2010-10-15	Bernerd Allmon, Jeremy Anderson	Internet
Griffon in Action	1935182234	375	2012-06-04	Andres Almiray, Danno Ferrin, , James Shingler	Java
OSGi in Depth	193518217X	325	2011-12-12	Alexandre de Castro Alves	Java
Flexible Rails	1933988509	592	2008-01-01	Peter Armstrong	Web Development
Hello! Flex 4	1933988762	258	2009-11-01	Peter Armstrong	Internet
Coffeehouse	1884777384	316	1997-07-01	Levi Asher, Christian Crumlish	Miscellaneous
MongoDB in Action	1935182870	-	2011-12-12	Kyle Banker	Next Generation Databases
Taming Jaguar	1884777686	362	2000-07-01	Michael J. Barlotta, Jason R. Weiss	PowerBuilder
Hibernate in Action	193239415X	400	2004-08-01	Christian Bauer, Gavin King	Java
Java Persistence with Hibernate	1932394885	880	2006-11-01	Christian Bauer, Gavin King	Java
JSTL in Action	1930110529	480	2002-07-01	Shawn Bayern	Internet
iBATIS in Action	1932394826	384	2007-01-01	Clinton Begin, Brandon Goodin, Larry Meadors	Web Development
Designing Hard Software	133046192	350	1997-02-01	Douglas W. Bennett	Object-Oriented Programming
Hibernate Search in Action	1933988649	488	2008-12-21	Emmanuel Bernard, John Griffin	Java
…					

Structured data: tables

Reading tables using the `csv` module

```
1  import csv
2
3  f = open('books.tsv')
4  table = list()
5
6  for row in csv.reader(f, delimiter = '\t'):
7
8      # ignore rows that are empty or start with '#'
9      if not row or row[0].startswith('#'):
10         continue
11
12     table.append(row)
13
14 # print first row of table
15 print(table[0])
```

Structured data: Matrices

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Quiz

True or *false*?

- XML tags have opening and closing elements
- XML and JSON are archaic data formats
- The *delimiter* parameter of `csv` reader specifies the the character that separates rows
- Each column of a table represents a single data point

Quiz

True or *false*?

- XML tags have opening and closing elements — true
- XML and JSON are archaic data formats — false
- The *delimiter* parameter of `csv` reader specifies the the character that separates rows — false
- Each column of a table represents a single data point — false

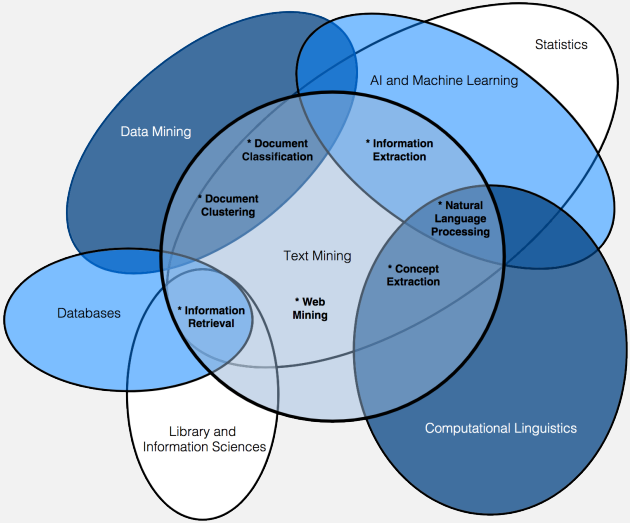

File I/O
File Formats
Text mining
NLTK

Text mining

Relies on *Natural Language Processing* (NLP)

Main (constitutive) tasks:

- Document summarization, clustering & classification
- Information extraction
- Information discovery

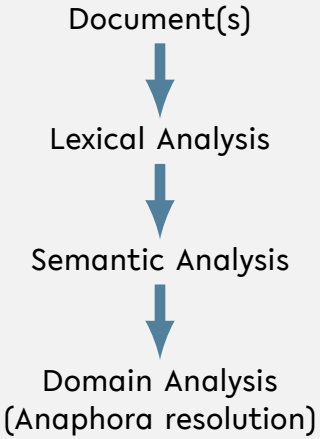


source: Miner, Gary. Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications. 1st ed. Amsterdam: Academic Press, 2012.

Document summarization, clustering & classification

- ***Document summarization***
  - Goal: Extract essence of a text
  - TextRank
    - Method for ranking sentences
    - Similar to Google's PageRank
- ***Document clustering & classification***
  - Uses classic data mining techniques
  - Popular: Supervised Learning methods
  - Applied to terms, documents or parts thereof

Information extraction



source: Miner, Gary. Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications. 1st ed. Amsterdam: Academic Press, 2012.

Lexical Analysis

- ***Tokenization***: decomposition into sentences or words
- ***Stemming***: reduction of words to their roots
- ***Lemmatization***: inflection & reduction of words to roots

Semantic & Domain Analysis

- ***Semantic Analysis***
  - Infers relationships of words
  - Often relies on *parse trees*
- ***Domain Analysis***
  - Establishes references between parts of text


File I/O
File Formats
Text mining
NLTK

Natural Language Toolkit – NLTK

A comprehensive library for natural language processing

NLTK supports

- Text corpora and lexical resources
- Tools for
  - Document summarization & classification,
  - Information extraction

Read the free book to learn more about NLTK at
`https://www.nltk.org/book/`

Stemming

- Process of reducing a word to its *root* (*stem*)
- Porter Stemmer
  - Proposed by Martin Porter in 1979
  - Idea: Each word can be represented by the form $[C](VC)^m[V]$ where
    $C :=$ consecutive consonants and
    $V :=$ consecutive vowels
    $m \geq 0$
  - Simple set of suffix reduction rules, e.g.
    - sses $\rightarrow$ s
    - ies $\rightarrow$ i
    - y $\rightarrow$ i
  - ***Outcome is not true root of the word***, but works well in practice to find words with same root of the English language

Lemmatization

- reduces the *inflected* words e.g.: runs, running, ran → run
- Requires additional information of the language
- WordNet Lemmatizer:
  - Uses WordNet database to inflect words
  - Works best if *part-of-speech* (POS) information is provided: e.g. is word a verb or noun?

```
from nltk.stem.wordnet import WordNetLemmatizer
wnl = WordNetLemmatizer()

# calling lemmatizer without POS information
wnl.lemmatize('ran')
# returns 'ran'

wnl.lemmatize('ran', 'v')  # 'v' for 'verb'
# returns 'run'
```

Quiz

True or *false*?

- Stemming is the process of decomposing text into smaller units
- Inflection is the change of a word's form
- The Porter Stemmer requires no adaptions to work well on any language
- The quality of lemmatization depends on the utilized data base
- Semantic analysis often relies on parsely trees

Quiz

True or *false*?

- Stemming is the process of decomposing text into smaller units — false
- Inflection is the change of a word's form — true
- The Porter Stemmer requires no adaptions to work well on any language — false
- The quality of lemmatization depends on the utilized data base — true
- Semantic analysis often relies on parsely trees — false

Recap

Summary

- Reading and writing files, command line arguments, prompt
- Structured and unstructured data formats
  - Text
  - JSON & XML
  - Tables & matrices
- Text mining, lexical analysis

What comes next?

- Play with NLTK
- Due date for this week's exercises is ***Wednesday, December 04, 2 pm, 2024***.

Next lecture: Numerical Data Analysis, NumPy, … …