Computational Methods for Linguists

Ling 471

Olga Zamaraeva (Instructor)
Yuanhe Tian (TA)

04/20/21

Reminders

- Blog 2 due tonight
  - Responses due by April 22
  - So, is coding the new literacy? :)

- Midterm evaluations
  - Please fill out today at the end of class
  - We will try to leave time

Plan for today

- Useful modules
  - built-in functions
  - module setup
    - Python project structure
- Complete preparation for Assignment 2:
  - Regular expressions
  - Tokenization
- Unicode (time-permitting)
- Clone https://github.com/olzama/April20-demo.git if want to follow along with coding

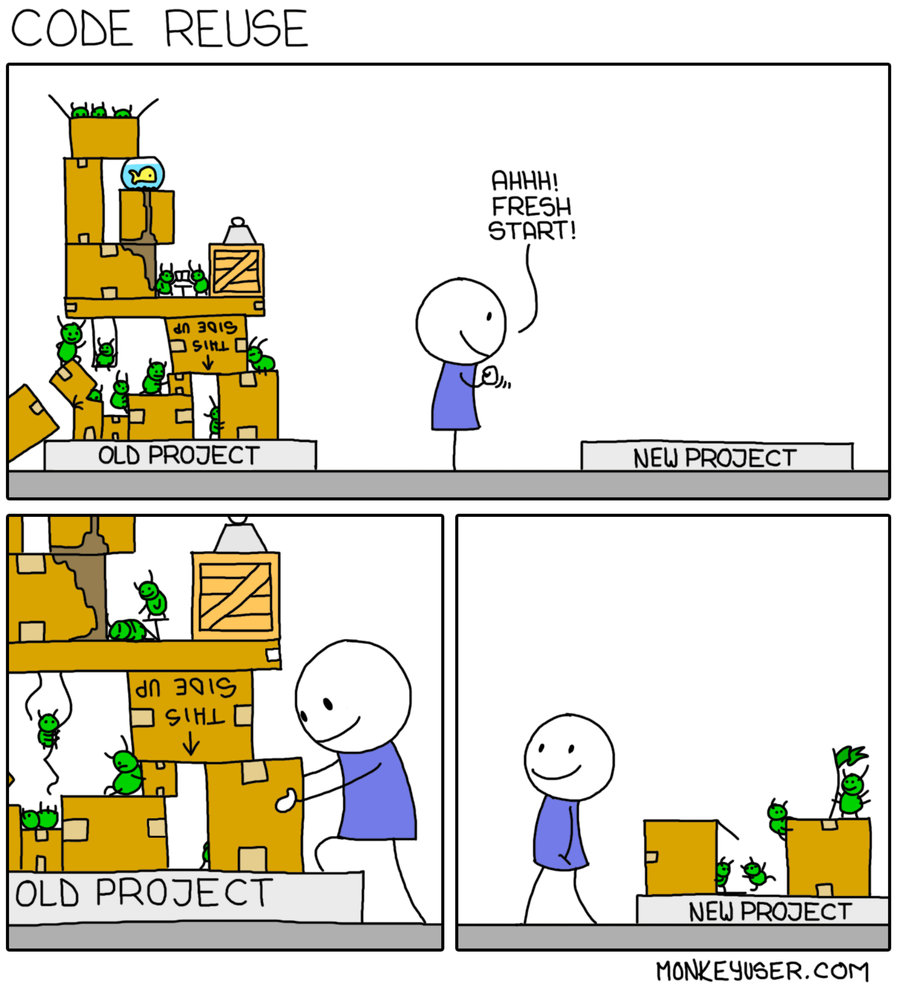
Python modules and built-in functions

Try to never write code yourself*

*unless it’s for an assignment :)

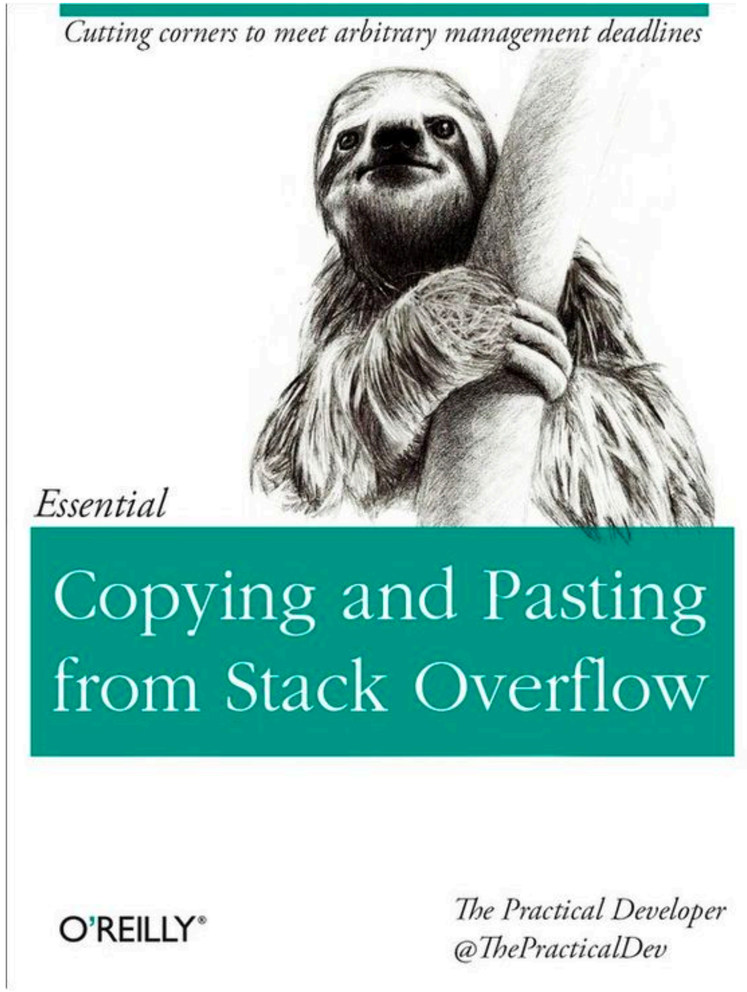
You, your code, and others' code

- It is important to **learn** programming
  - through **practice**, as in writing your own code
- **However**:
  - Code has **bugs** in it
    - **everybody's** code has bugs in it
    - unless the code had **already** been debugged
      - for a **while**!
  - Code **reuse** is usually **best** practice
    - unless it constitutes plagiarism, of course
    - P.S.: I realize this cartoon is making fun **of** code reuse, but it illustrates the problem of **actually** starting from scratch, just as well



Styles of code reuse

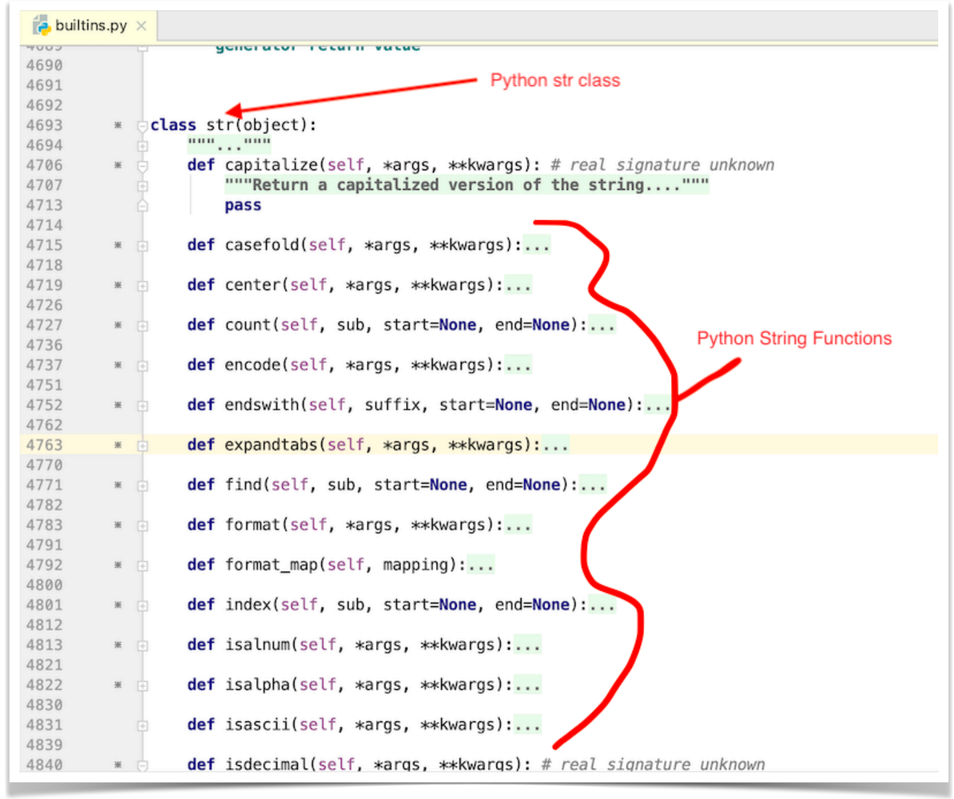
- Python **built-in functions** and **modules**
- Posted solutions/answers
  - e.g. Stackoverflow
  - careful **not to plagiarize** in HW setting!
    - if using a solution from a website, **credit** the author and give **link**!
- Finally, **without** learning programming, you **won't** be able to use solutions and in-built functions **anyway**.



https://www.pinterest.com/pin/220676450469670851/

Built-in functions

- In python in particular, many things are **already** implemented
  - **Searching** and **sorting** in data structures
  - **Cleaning** up data
  - **Splitting** text by a delimiter
  - ...and many, many **other** things
    - NB: each function is a function of a **class** or an **instance** thereof
    - Meaining, you call e.g. “hello world!”.split()
    - Or, import string -> string.punctuation

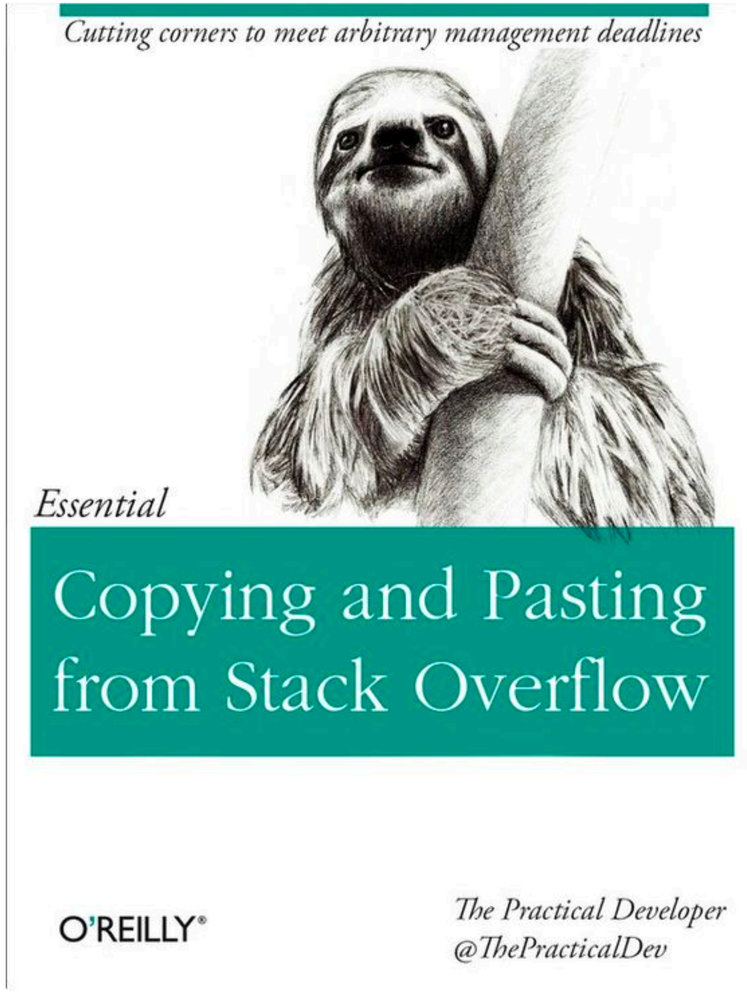


https://www.journaldev.com/24588/python-string-functions

Stackoverflow

and other forums

- A Q&A forum for programming
  - “What’s the best way to clean out punctuation in python3?”
- Unless assigned as a HW, don’t implement things yourself until convinced there is no **module** for it



Demo: Assignment 2 skeleton

Python modules

- **Modules** are simply **python files**
- They can be **imported** in other python files
- Then, the **functions** etc., of the imported file become **available** in the file where it was imported
- In order for a module to be imported:
  - the module needs to **exist** on **your** computer
    - many modules can be installed using the **pip** tool
  - Python needs to be able to **find** the module
    - same directory
    - pythonpath

https://www.hellocodeclub.com/how-to-fix-modulenotfounderror-no-module-named-pygame/

Finding modules

- **Easiest:** put the python file you want to use as a module in the same directory with your program
  - But: you **don't** want to multiplies files unnecessarily!
  - Keeping copies puts things out of sync!
    - which can be **very frustrating** to debug
- Useful reading: https://towardsdatascience.com/how-to-fix-modulenotfounderror-and-importerror-248ce5b69b1c

```
└── myproject
    ├── mypackage
    │   ├── a.py
    │   └── anotherpackage
    │       ├── b.py
    │       ├── c.py
    │       └── mysubpackage
    │           └── d.py
```

Sample directory structure
https://towardsdatascience.com/how-to-fix-modulenotfounderror-and-importerror-248ce5b69b1c

```
# in module a.py
import anotherpackage.mysubpackage.d


# in module b
import anotherpackage.c
```

Examples of valid imports

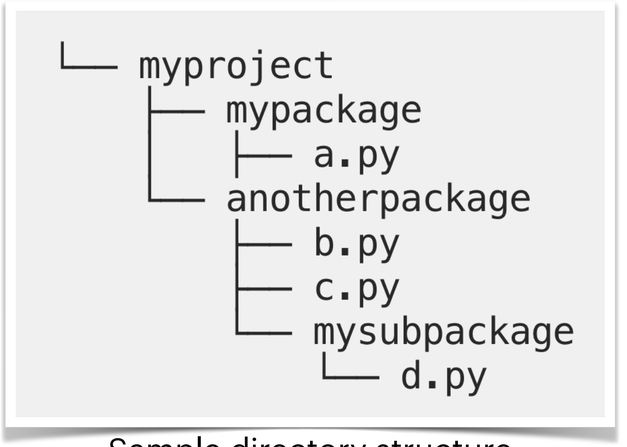
Finding modules

```
export PYTHONPATH="${PYTHONPATH}:/path/to/your/project/"

# * For Windows
set PYTHONPATH=%PYTHONPATH%;C:\path\to\your\project\
```

Commands to add a project to Pyhonpath, in bach/batch
This is to be executed in command line, or to be added to e.g. bash_profile

- Keep modules **neatly** separately
- Add the path to the **folder** from which you want to be able to import a module to PYTHONPATH
  - **PYTHONPATH**: a list of paths for python interpreter to look for modules in
- Automatic installers will often add the path when installing
  - e.g. **pip**
  - ...but not always. Then, need to **locate** the installed package **folder** and add its path to PYTHONPATH



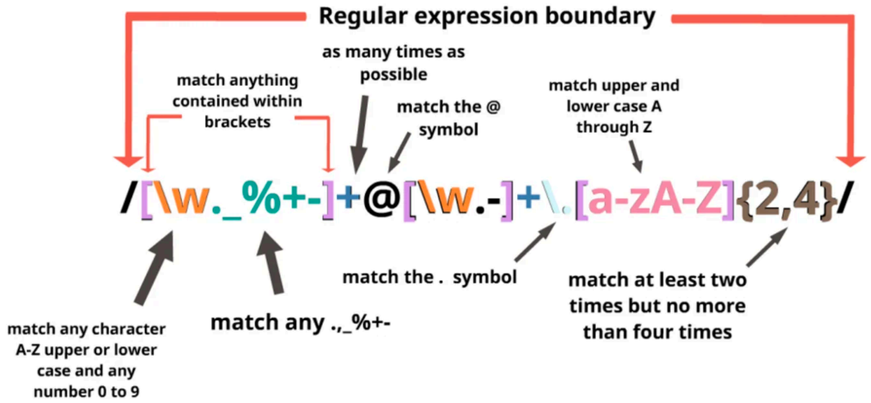
Sample directory structure
https://towardsdatascience.com/how-to-fix-modulenotfounderror-and-importerror-248ce5b69b1c

Adding to PYTHONPATH in VS Code is confusing. Only do it if really needed. There are alternatives; VS Code is good for debugging, not generally running projects!

Regular expressions

Regular expressions

aka regex

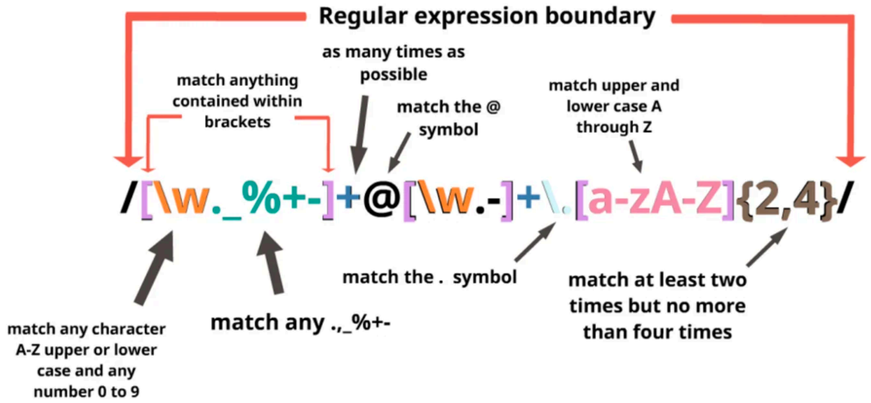


https://dev.to/mconner89/regular-expressions-grouping-and-string-methods-3ijn

- Special language for pattern matching in text
  - Built into python **re** module
- **Compactly** describe patterns using special characters
  - similar in different programming languages, but there may be slight differences
  - Formal dimension: “regular languages”
    - out of scope for this class
    - but: a regular expression describes a set of strings
      - a “language”!
- Your best friend: https://regex101.com/
  - though note only python 2.7 is available there
  - Also: https://docs.python.org/3/library/re.html

Regular expressions

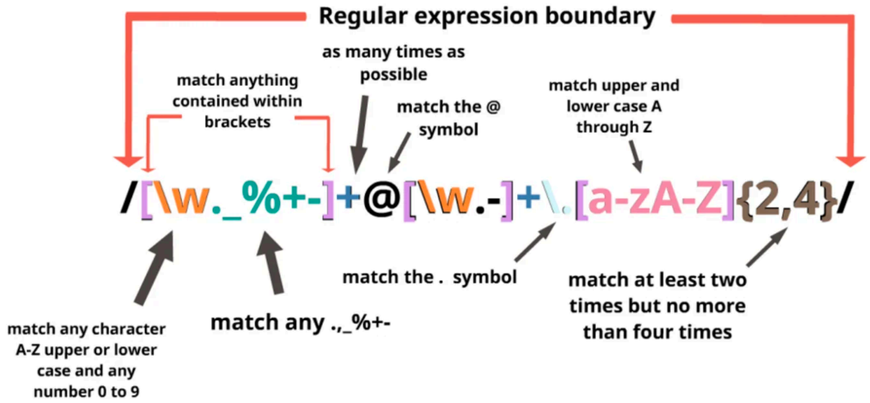
Some common examples



https://dev.to/mconner89/regular-expressions-grouping-and-string-methods-3ijn

- trivial/literal: family
  - matches just “family”
- disjunction: famil(y|ies)
- character classes: [Tt]he, bec[oa]me, [A-z], [0-9]
- More than one time: [0-9]+
- Zero or more times: [0-9]*
- Zero or one time: colou?r
- Any character: . (the dot)
- Escape special characters and treat them literally: *
- Special characters: \w (word), \s (whitespace)

Regular expressions
in python



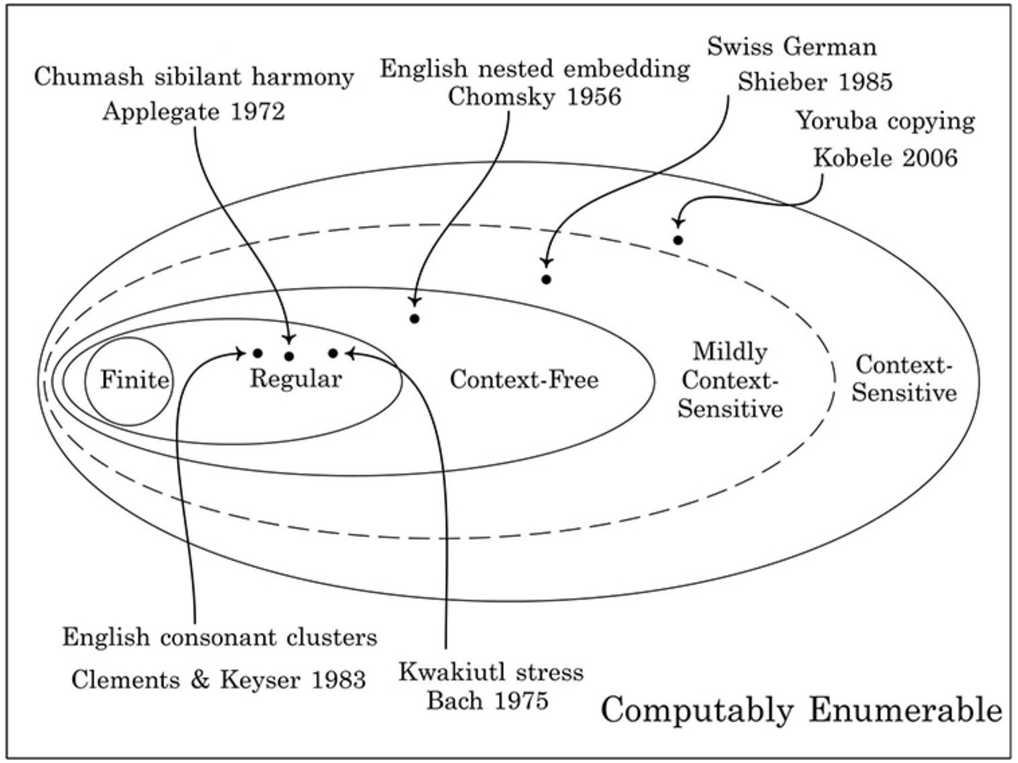
https://dev.to/mconner89/regular-expressions-grouping-and-string-methods-3ijn

- **Import** the **re** module
- **Write** a regular expression **pattern**
- Call an appropriate re **function** and **pass** the pattern as an **argument**
  - re.sub(pattern, substitution, string)
  - re.search(...)
  - re.finditer(...)
- **Save** the **resulting matches** in a variable
- **Do** what you need with the matches

Regex demo

Regex
limitations

- Regex are sometimes used for modeling human language
  - e.g. a grammar written in regex
    - famil(y|ies)
  - can go pretty far
  - but cannot model e.g. recursion
    - phonology can be modeled in regex
    - syntax generally can't be



picture from Rawski & Heinz. Language, vol. 95 no. 1, 2019, pp. e125-e135.

Regex
are slow

- Beware of using regex too much
  - they will slow your program down!
- Example: Cleaning out punctuation
  - Assignment 2
    - uses a special python function
    - could also use regex
      - but that would be slower
      - https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string

```
sets      : 19.8566138744
regex     : 6.86155414581
translate : 2.12455511093
replace   : 28.4436721802
```

https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string

Tokenization

Tokenization

- Splitting data into smaller parts (**tokens**)
  - e.g. words
- Different styles/level of tokenization
  - simply split by space
    - What if text has extra spaces?
      - cleanup required: replace multiple spaces/tabs with a single space
  - lemmatization/normalization
    - count e.g. inflected forms as identical tokens
    - capitalization