Programs for linguists

Python for Linguists
Week 3

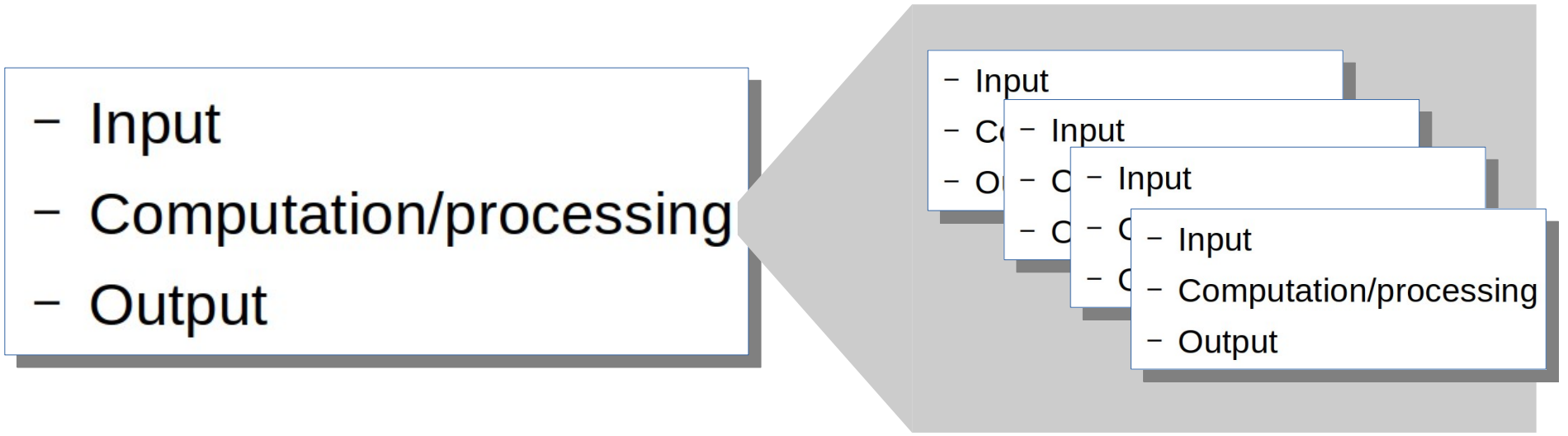
Why program? For a researcher

- Standard research tools: use them if they fit your need
  - e.g., Excel, SPSS, SketchEngine, PennController, ...
- But: these may not fit your *groundbreaking* research
- If you can program, you'll be better able to:
  - Understand these existing tools
  - Adapt existing tools to suit your specific needs
  - Make your own tools

Why Python?

- *Relatively* easy to learn (but not *easy* for most people)
- Relatively easy to read by humans
- Very popular (also in academia inc. linguistics)
- ...hence many useful *libraries.*

What is a program?



Example: experimental data

- Input:
  - CSV file with data from a psycholing. experiment.
- Computation:
  - Read in the csv file
  - Remove outliers
  - Group data by the various factors and plot the means and sd
  - Apply appropriate statistical tests
- Output:
  - Plots, effect sizes & p-values for various factors

Example: twitter trends

- Input:
  - Millions of tweets about the climate, stored in several large files.
- Computation:
  - Read each file, and for each tweet:
  - extract the tweet’s main text and classify it as ‘happy’ or ‘sad’.
  - Then aggregate these values by tweet creation month.
- Output:
  - Twitter climate sentiment per month.

Example: nonsensical sentences

- Input:
  - Schematic specification of various sentence types
  - English vocabulary with parts of speech
- Computation:
  - Read in the vocabulary file
  - Randomly instantiate the schematic with syntactically fitting vocabulary items
  - Remove results that are semantically coherent
- Output:
  - Nonsensical sentences that match the specification

Example: research website

- Input:
  - A bunch of text files.
- Computation:
  - When a user requests the research website:
  - Present a random text file, sentence by sentence
  - After every 3 sentences, let the user enter a question the text evokes.
  - Store entered questions in a database.
- Output:
  - A dataset of texts annotated with the questions they evoke.

Example: machine translation

- Input:
  - Python exercises in English.
- Computation:
  - Separate text from code examples.
  - Translate the text.
  - Add translated text to code examples.
- Output:
  - Python exercises in Dutch.

Example: Machine Translation

- Input:
  - 50 Million word English-Dutch parallel dataset from EuroParl
- Computation:
  - Read in the data
  - Set up a deep learning model with random weights
  - Feed the data to the model in batches
  - Iteratively update the model’s weights to minimize translation error
- Output:
  - A model that can translate English to Dutch(ish).

Essential skills

Programming requires:

- **Decomposing** a task into sub-tasks that other people have likely already solved.

  Hence also:

- **Searching** for those expected solutions on google/bing/duckduckgo (usually StackOverflow).

We’re already practicing the first; the second not so much (yet).

Adventure

- Input:
  - Tweets about COVID
- Computation:

  ...

- Output:
  - Plot showing twitter anti-vax trend over time

Next section: Functions

- **Chunking** a program into functions serves several purposes:
  - Improve code **usability**:
    - Defining functions = *teaching the computer a new words.*
    - This makes it easier for us to tell it what to do.
  - Improve code **quality** (e.g., SLAP, DRY):
    - readability
    - correctness/safety (e.g., encapsulation)
    - maintainability