Lecture 1:
Course Introduction, Setup

Ling 1330/2330 Intro to Computational Linguistics

Na-Rae Han, 8/29/2023

Objectives

- **Computational linguistics**: what is, what we will learn

- Course Introduction
  - Syllabus: https://sites.pitt.edu/~naraehan/ling1330/

- Introduction to **Python 3**
  - Using **Python Interactive Shell** (ex. IDLE, IPython)
  - A warm-up: processing a string
  - Setup, housekeeping

Language engineering applications

- **Language engineering** is at the center of many essential tools and applications:
  - Internet search engines (Google, Bing, DuckDuckGo...)
  - Automatic (machine) translation systems
  - Voice and speech recognition, speech-to-text, auto caption
  - Smart assistants (Alexa, Siri, Cortana, Google Assistant)
  - Automatic essay scoring systems (*e*-rater), computerized language tests (Duolingo English test)
  - Spelling and grammatical error correction
  - ... and many more!

Language engineering & AI

- **Computational linguistics**: an interdisciplinary field dealing with modeling of natural language from a computational perspective.
- A "**natural language**"?
  - Any real-world human language.
  - cf. constructed, artificial, or computer languages such as: formal (logic) language, computer-programming languages, Klingon, etc.
- Areas: **natural language processing** (NLP), natural language understanding (NLU), human language technology (HLT), speech processing and synthesis, dialogue systems…

  ← part of **AI** (Artificial Intelligence)

Role of linguists

- What is the role of linguists in all of these?

- Linguistic expertise intersecting with engineering is in high demand. Tasks include:
  - Build linguistic infrastructures: grammars, lexicons, ontologies (knowledge bank)
  - Manage language data: design and run linguistic annotation projects
  - Design dialogue systems for voice assistants
  - Consult for quality enhancement (machine translation systems, search results, etc.)
  - Build NLP systems, train models
  - Set standards and guidelines for localization

← Working knowledge of computational linguistics is essential

Computational linguistics

- What is the goal of computational linguistics?
  - Represent human language in terms of information processing.
- ***Theoretical* computational linguistics**
  - Investigates the *computability* of human language and linguistic theories.
  - Formal language theory and automata theory.
- ***Applied* computational linguistics**
  - The focus is on building (working) models of human language.
  - Natural language processing (NLP) and engineering (NLE), human language technology (HLT), artificial intelligence (AI)
  - Often centers around building practical applications: spell checkers, machine translation systems, automatic grader, etc.
  - Two distinct flavors: **symbolic** vs. **statistical** approaches

Computational linguistics course

- Old LING 1330 course description:

> In both linguistics and computer science, we need to study languages and their **grammar** from a **mathematical** point of view. This course is an introduction to **the mathematical theory of language and its applications**. The first half will deal mainly with elements of the theory of **automata** and its relation to grammars. The second half will survey ways in which this theory can be applied to English grammar and to the **design of programming languages**. We will concentrate on syntax, but will also pay some attention to theories of meaning.

  - The class was about *theoretical* and *symbolic* computational linguistics.

What we learn in *this* class

- Theoretical computational linguistics
  - Formal language theory
    - Automata, the Chomsky hierarchy

      ← We will cover only the basic ideas.

- Applied computational linguistics
  - Text and corpus processing
  - Using NLTK as an NLP library
  - Natural language processing applications:
    - Spelling correction, POS tagger, morphological analyzer/synthesizer, machine translation, text classification

Our objectives:

Linguistics-focused *foundations* of CL ✓

Not: cutting-edge NLP solutions ✗

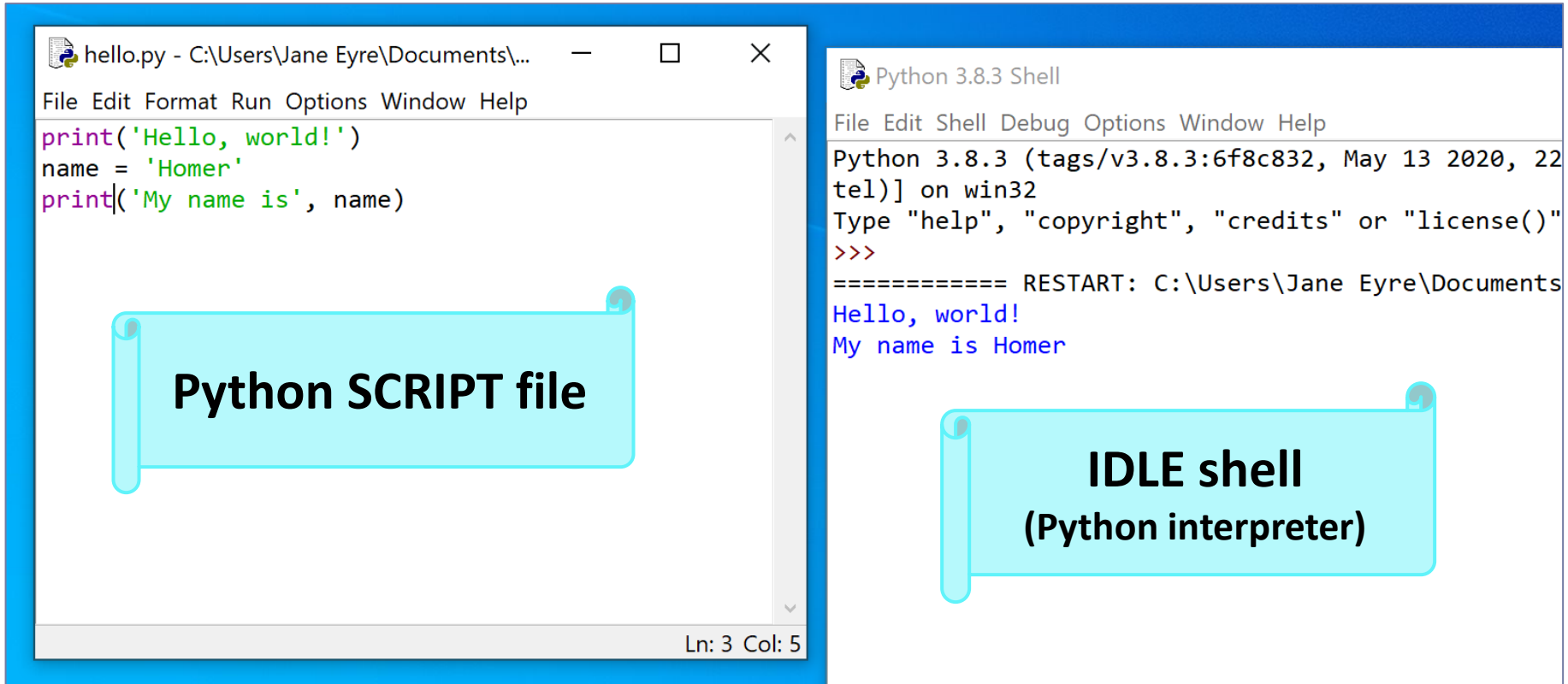
Syllabus time!

- Go over the course home page
  - https://sites.pitt.edu/~naraehan/ling1330/
  - Schedule, requirements, policies

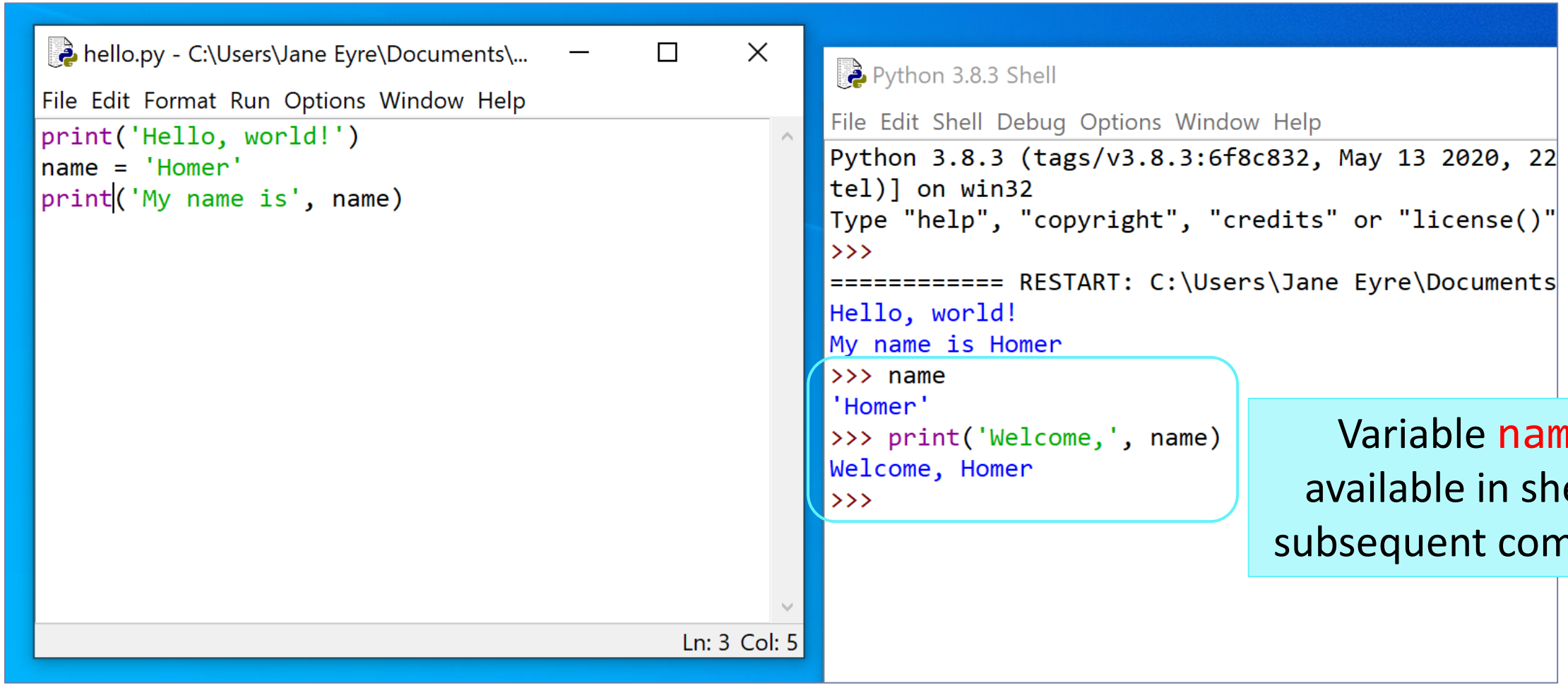
Python: script vs. shell

1. **Python script** is a plain-text file with .py extension.
2. **Python shell** is an interactive environment where Python will respond to each individual Python command.

← Great tool for learning!



Following up a script in shell



Warm-up: fox in sox

5 minutes

Let's practice string processing.

- Download this template script:
  - https://sites.pitt.edu/~naraehan/ling1330/fox_in_sox.TEMPLATE.py

  ← Will paste link on MS Teams

- Complete [1] – [5].

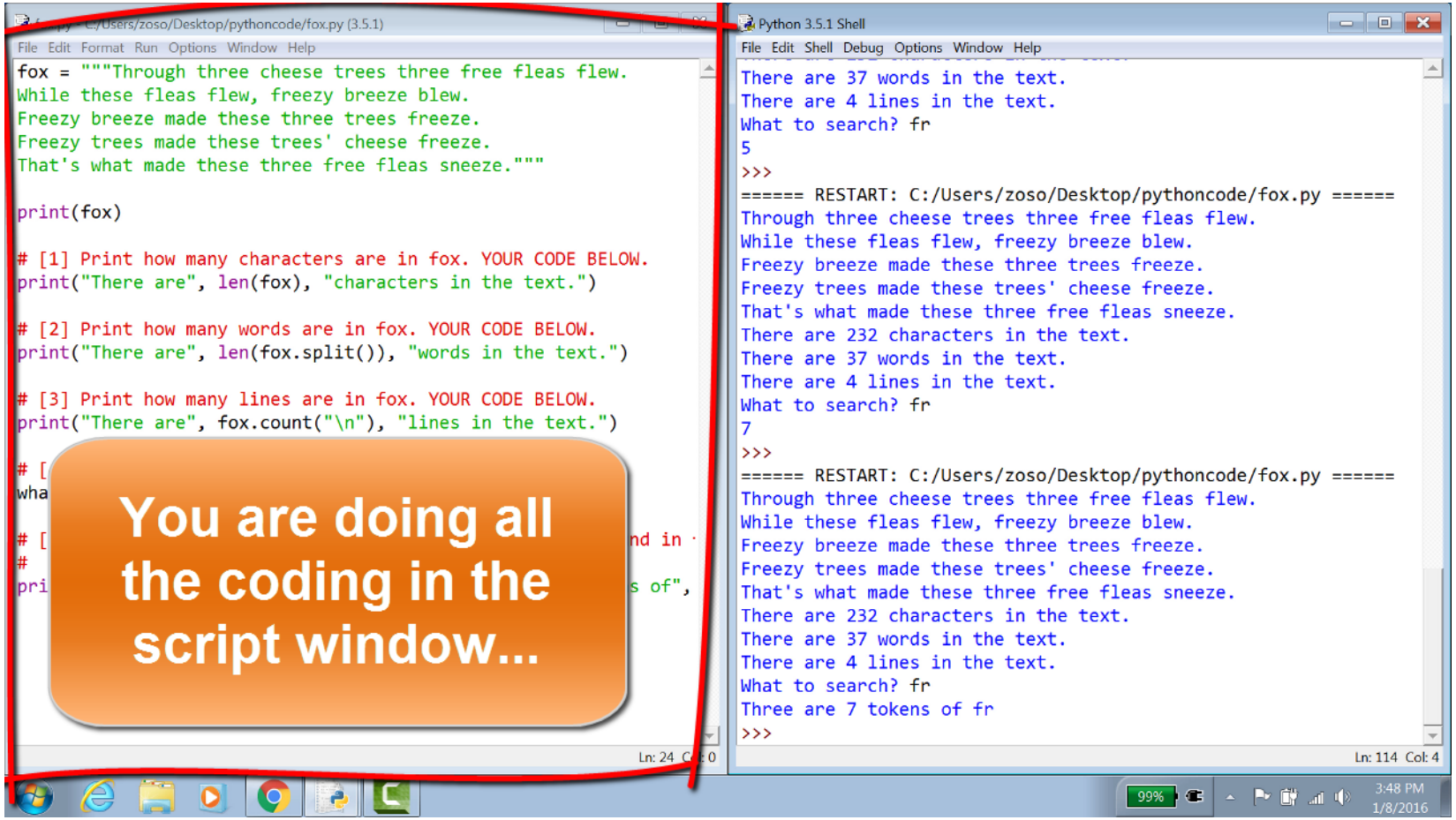
Speed-coding `fox_in_sox.py`

- Video of me completing the script, in 5x speed!
  - https://sites.pitt.edu/~naraehan/ling1330/code_in_shell.mp4

- Watch closely! This is an IMPORTANT video -- **It will change your life.**

How to program efficiently

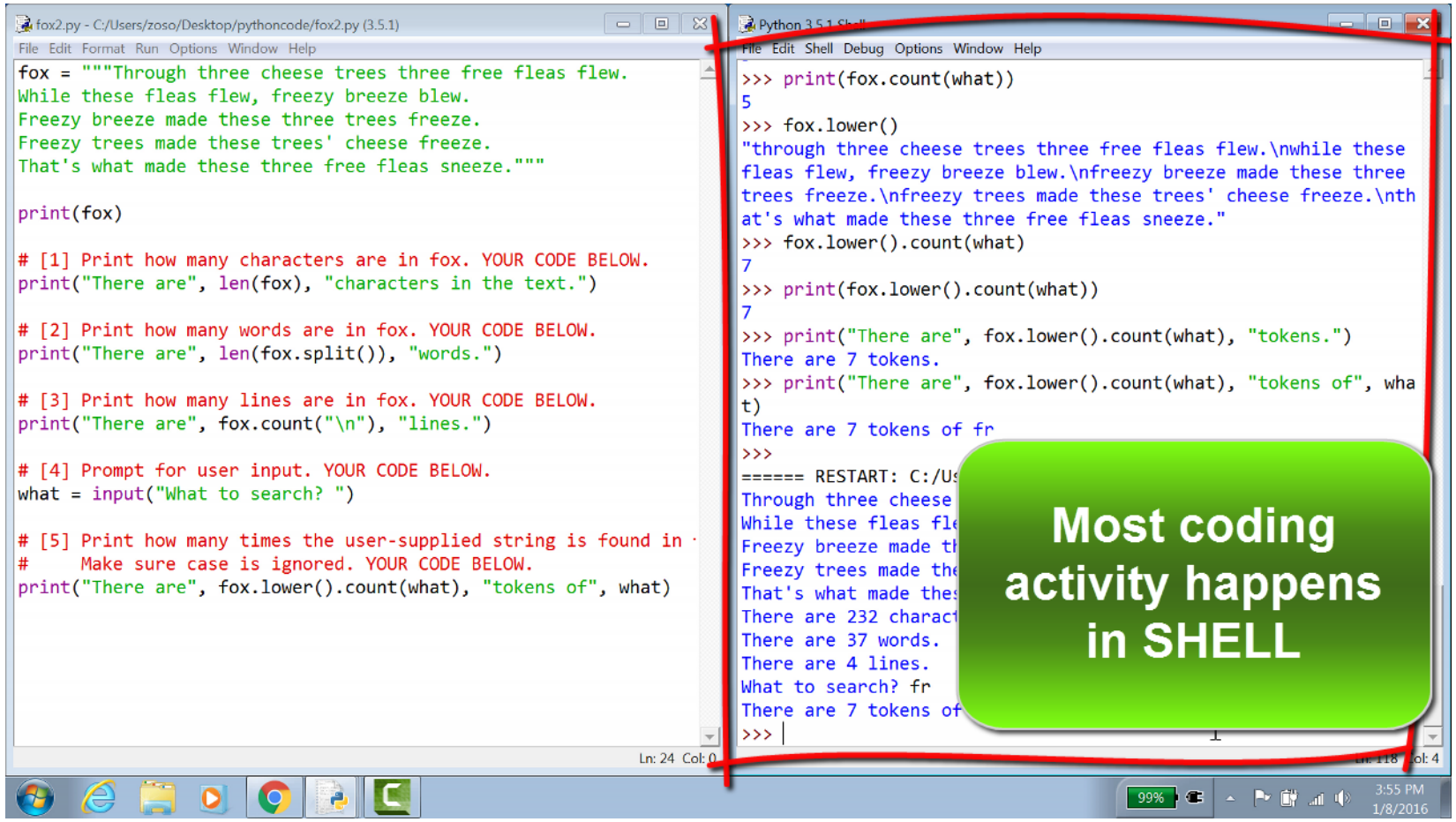
- **The rookie way**:
  - You are doing all your coding in the script window, blindly.
  - You only use the IDLE shell to verify the output of your script.



How to program efficiently

The seasoned Python programmer way:

- After a script is run, all the variables and the objects in it are still available in **IDLE shell** for you to **tinker with**.
- Experimenting in SHELL is much quicker – **instant feedback**!
- You should **go back and forth between SCRIPT and SHELL** windows, *successively building up* your script.



Command history

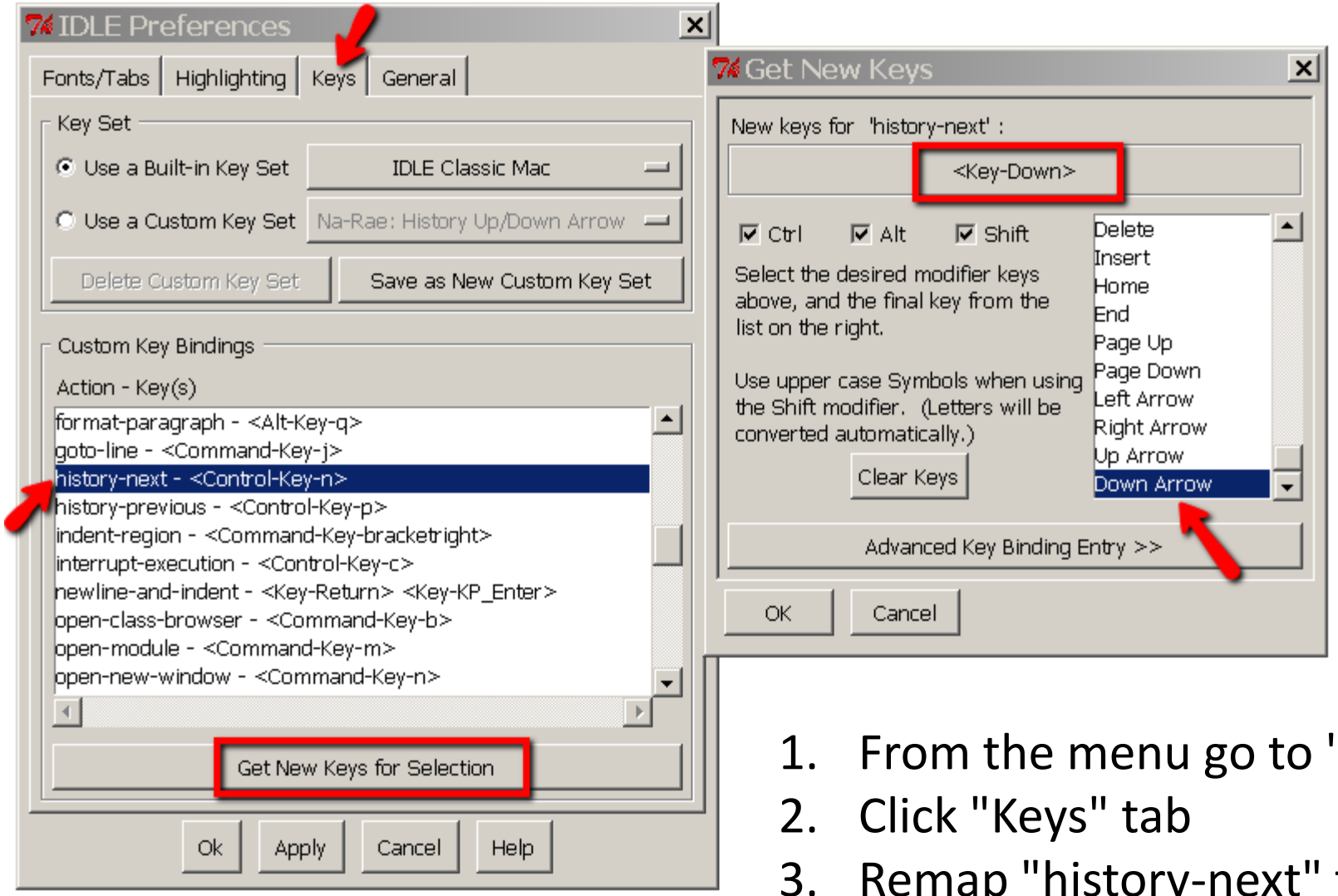
- Every command you type in is stored as command history.
- You can quickly bring up and edit previously entered commands using these shortcuts (IDLE):
  - Windows: **Alt+p** / **Alt+n**
  - Mac: **Ctrl+p** / **Ctrl+n** (previous/next)

```
>>> print('Hello, world!')
    Hello, world!
>>> print('Hello, worl
```

You can edit command line

- But these key combos are cumbersome! Let's remap to:
  - ↑ (Up arrow: **previous** command)
  - ↓ (Down arrow: **next** command)

Arrows are commonly used in shell



1. From the menu go to "Options → Configure IDLE"
2. Click "Keys" tab
3. Remap "history-next" to down arrow, and "history-previous" to up arrow

Housekeeping (1)

- All installation & setup instructions are on "Checklists" page:
  https://sites.pitt.edu/~naraehan/ling1330/checklists.html#setup

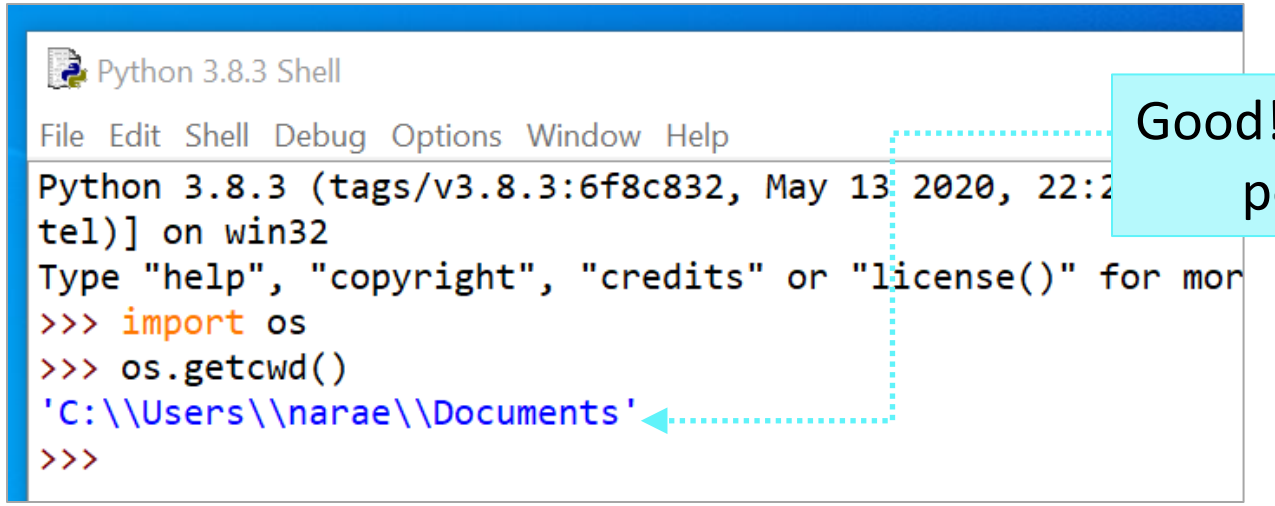
- Install **Python**, version 3.10 or 3.11.
  - Anaconda Python recommended (details on checklists page!)
  - Also good: official python.org distribution

- Launch **IDLE shell**.
  - Anaconda users: learn how to launch it
  - *Windows* Anaconda users: need some config

Housekeeping (2)

- **Designated folder for coding work**:
  - Create a folder within **Documents** called "**ling1330**" or "**ling2330**", and store all your coding work there.
  - So this folder will be something like:
    - `C:\Users\yourname\Documents\ling1330` (Windows)
    - `/Users/yourname/Documents/ling1330` (Mac)

- Verify your default "current working directory":



Good! sensible default CWD, points to user folder

NOT good: system directories like C:\Program Files\Python311

Wrap-up

- Exercise #1 out: Python refresher quiz
  - Due 15 minutes before next class (10:45am Thursday), on Canvas
- Take Day 1 Survey
  - On Canvas.
- Help with your Python settings?
  - Come see us. Na-Rae: Wed 1-3pm, Tianyi: Wed 1-4pm (temporary office hours for this week)
- Next class (Thu):
  - Encoding systems, writing structured programs