Discussion 1

Homework 1 Part A

Welcome! Information Retrieval Discussion

Itinerary September 29

Cover Discussion Syllabus

Icebreaker

Homework

- Tools Setup
- Part A

Questions (10 min)

Upcoming Deadlines

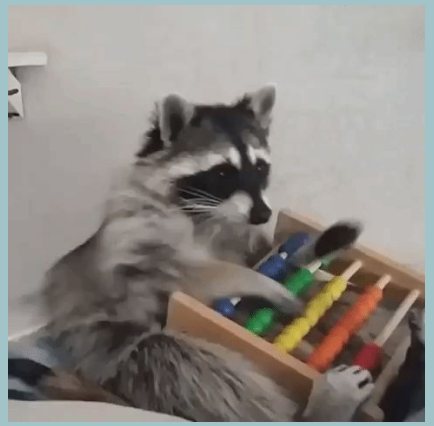
October 4, 8am: Register for STEM career fair!

Brooke Ryan

About Me

- Second year Master’s student in Computer Science
- Worked as a Software Engineer for about 3 years
  - Blizzard Entertainment
  - Intuit
- UCSD ‘17 Math-CS major
- Grew up here in OC!

Discussion
Logistics and Expectations



(15 min)

Discussion
Logistics

Location

- 3pm , **in person**
- 4pm, **virtual**

Attendance

- Not mandatory but encouraged

Masks

- Required for in-person sessions
- Cannot fill room capacity beyond 75

Discussion

- Discussion session will be a guided, **interactive** review of course material
  - Topic determined by most pressing course deadline (ex: homework, quizzes)
- Will **NOT** be a lecture
  - Active learning, participation, discussion with classmates
  - Also means I need your **participation** to continue on in the discussion!
- There are **no** “dumb” questions!!
  - Respectful, inclusive environment

VS.

Office Hours

- **Individually-focused help**
  - Debugging
  - Code errors
  - Tool setup

Ed

- Any kind of question!
  - (Extra credit earned for answering classmates’ questions!)
- Can make private post to staff for personal questions

Additional Resources

(my) Office Hours

- 10 - 10:50am Mondays and Wednesdays via Zoom

Recordings

- Will record 4pm Zoom section, please remind me at the beginnings though!
- Posted on Canvas later

Questions?

Icebreaker (3 min)

Introduce yourself to a neighbor:

- Major
- Year
- What's the best part about being back on campus?

Homework 1

Setup and Tools

How to get started?

In-Person:

ICS Computing Labs on campus

Or, on your **own machine**...

Personal Machine Setup

Install the following tools:

1. Anaconda or Miniconda
    a. Follow instructions under “regular installation”
2. PyCharm Pro
    a. Free pro version for students!

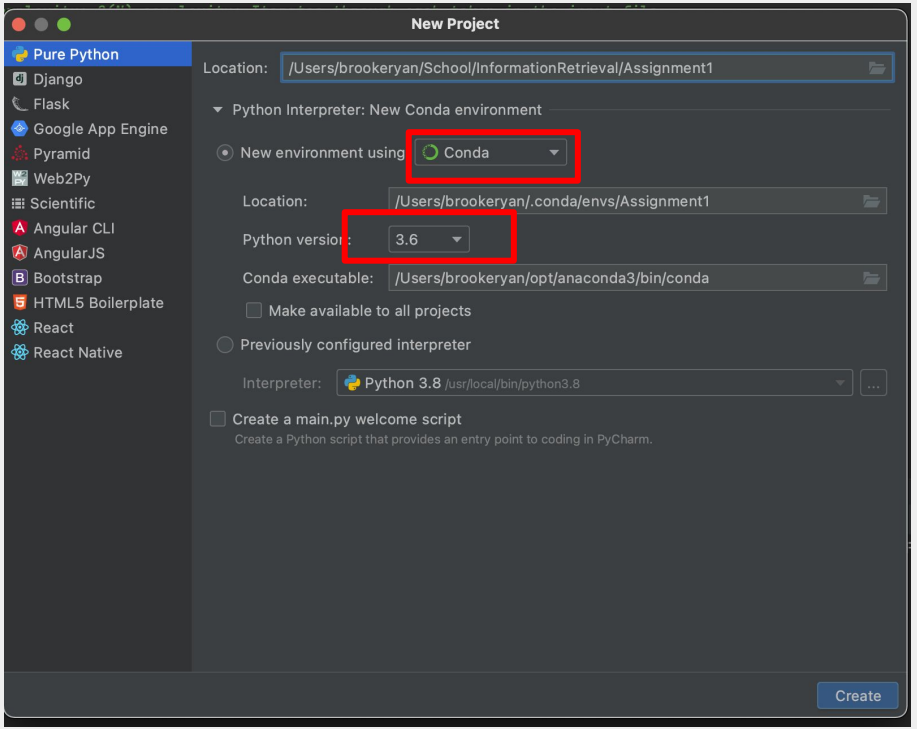
Recommended tools:

Not required, but recommended for ease of development!

- GitKraken (free Pro version for students!)
- GitHub
- iTerm2 (MacOS) or Windows Terminal

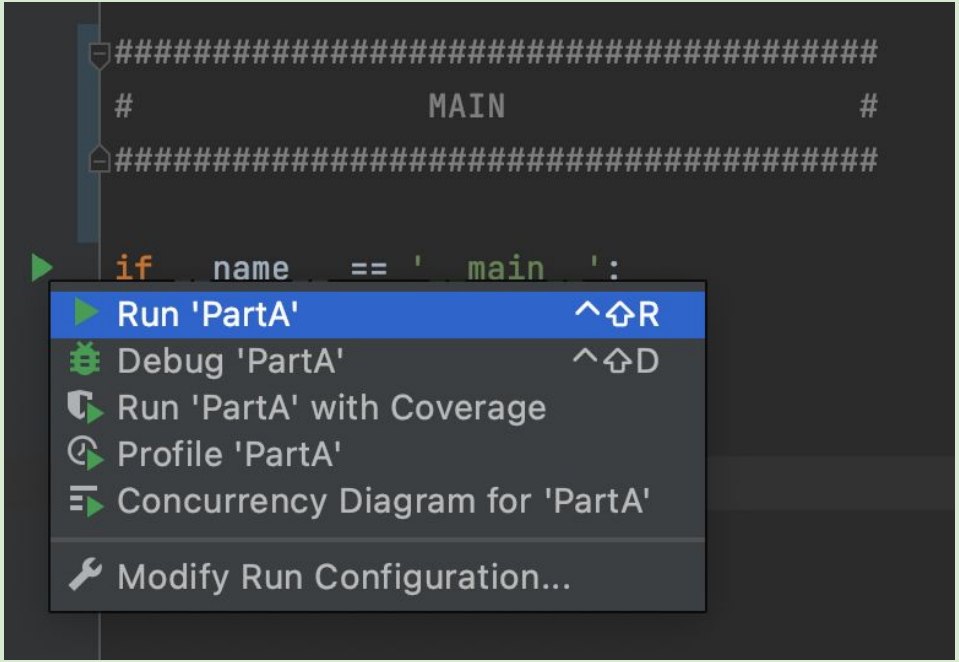
PyCharm

- Create a new PyCharm project



PyCharm

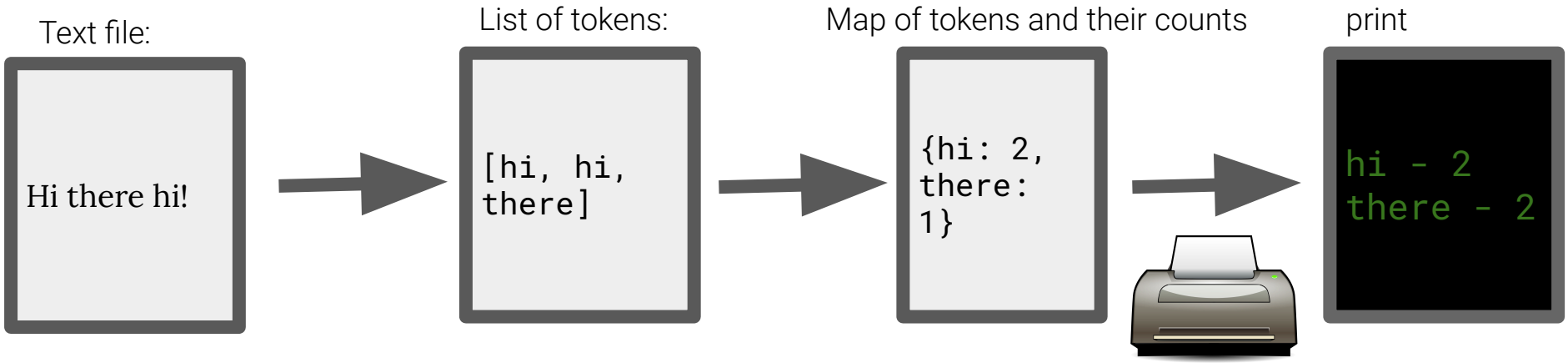
- Recommend using PyCharm because it is really easy to run and test your code!



Homework 1

Part A - Word Frequencies Library

Part A

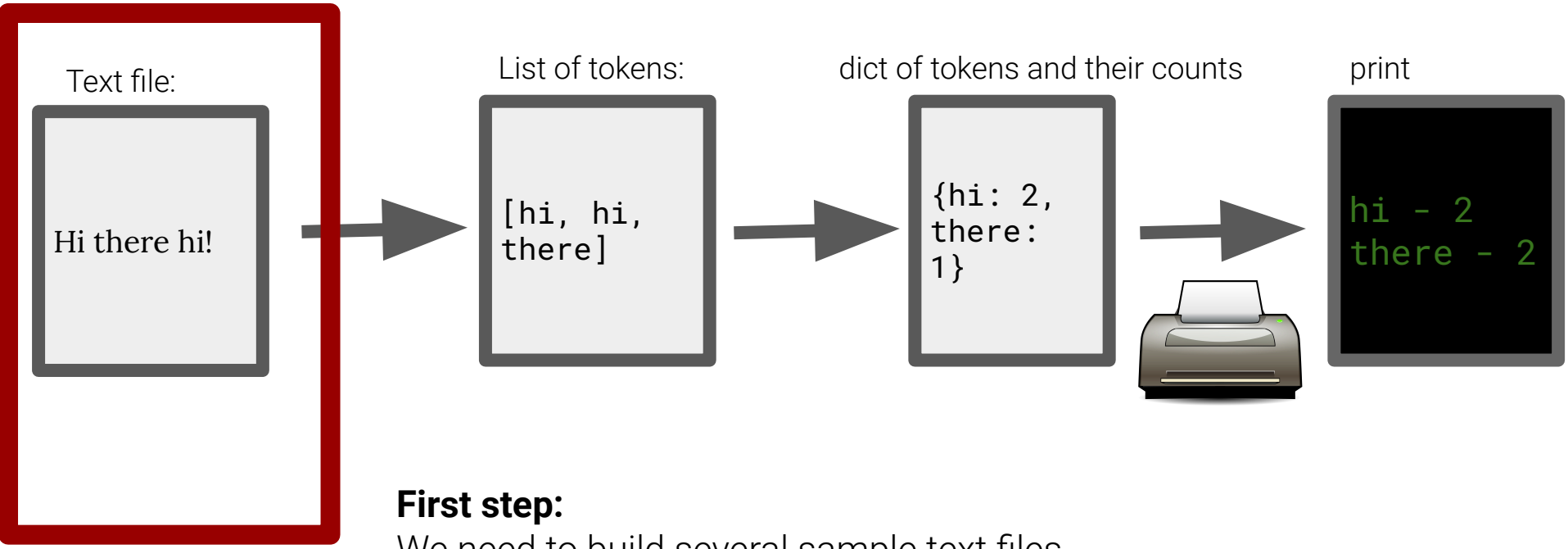


Keep in mind these functions will be re-used in Part B, which will use **larger** text files to test. Keep efficiency in mind!

Key Concepts for Part A

- **Test cases**
  - Look through the homework instructions carefully, and translate requirements into test cases
- **Tokenization**
  - Based on the requirements, what kind of tokenization do we need?
- **Efficiency**
  - How can we make sure large test files run efficiently?
- **Python Built-In Libraries**
  - re
  - list
  - dict
    - `sorted()` function

Part A



First step:
We need to build several sample text files that correspond to the assignment requirements

Building the Test Cases (Part A) - Example

1. Read through the instructions on Assignment 1 **carefully** -- some of the most important requirements are found scattered throughout the instructions!
2. Highlight parts of the instructions that correspond to **requirements**
3.

Part A: Word Frequencies (40 points)

- **Method/Function:** List<Token> tokenize(TextFilePath)
  Write a method/function that reads in a text file and returns a list of the tokens in that file. For the purposes of this project, a token is a sequence of alphanumeric characters, independent of capitalization (so *Apple*, *apple, aPpLe* are the same token). You are allowed to use regular expressions if you wish to (and you can use some regexp engine, no need to write it from scratch), but you are not allowed to import a tokenizer (e.g. from NLTK), since *you are being asked to write a tokenizer*.

lowercase.txt

apple apple apple

multicase.txt

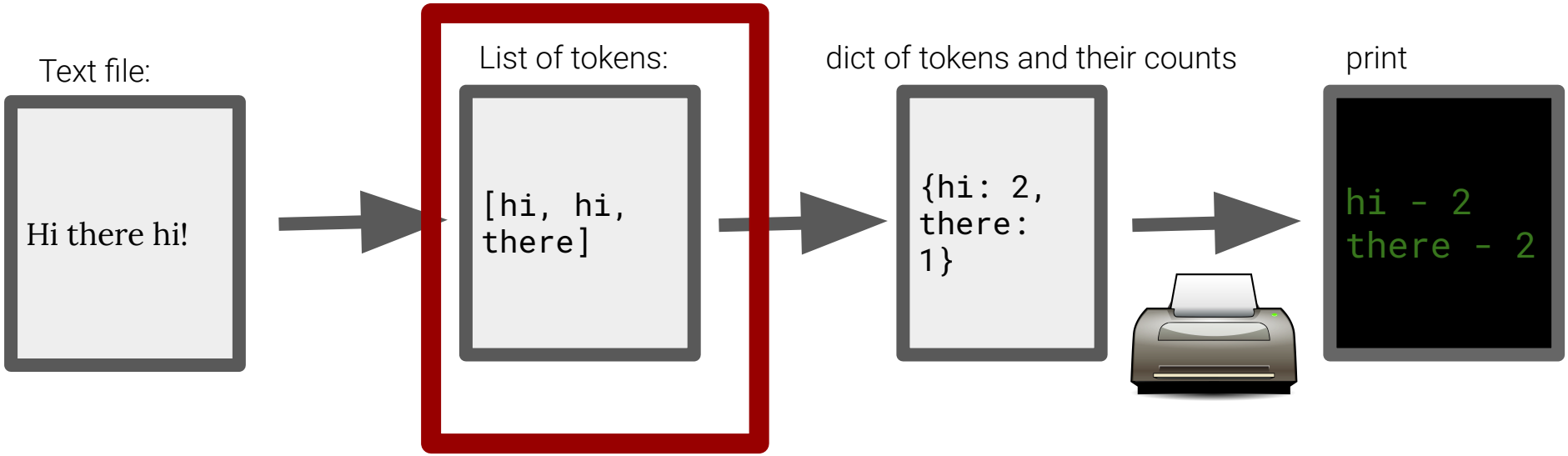
Apple apple aPpLe

Important assignment requirements/ test cases:

- Non-English characters should be skipped
  - Cannot have code crashing on “bad” input
- “Whitespace” (newlines and spaces) should not be counted as a token
- Blank files should have 0 tokens

If you have any hesitations or questions about requirements, be sure to post on Ed Discussion to clarify!

Part A



Tokenization:

How can we build a `tokenize` function that matches our supporting test cases & requirements?

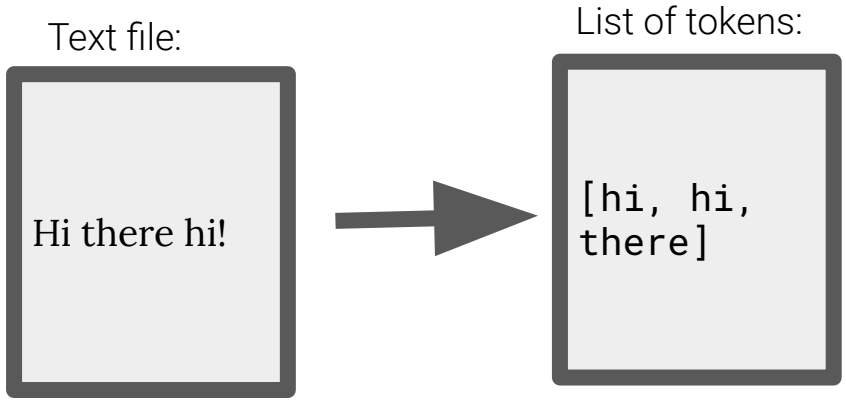
Also, efficient file handling for large text files

Tokenization

Tokenization is the process of breaking text into smaller pieces called tokens.

Traditional methods of tokenization include whitespace, punctuation, or regex tokenization.

Assignment says: “a **token** is a sequence of **alphanumeric (English) characters**, independent of capitalization”



Python's Regular Expression Operations (regEx)

- Reference the documentation
- Looking for tools we can use to filter out unwanted characters based on the test cases/ requirements we built

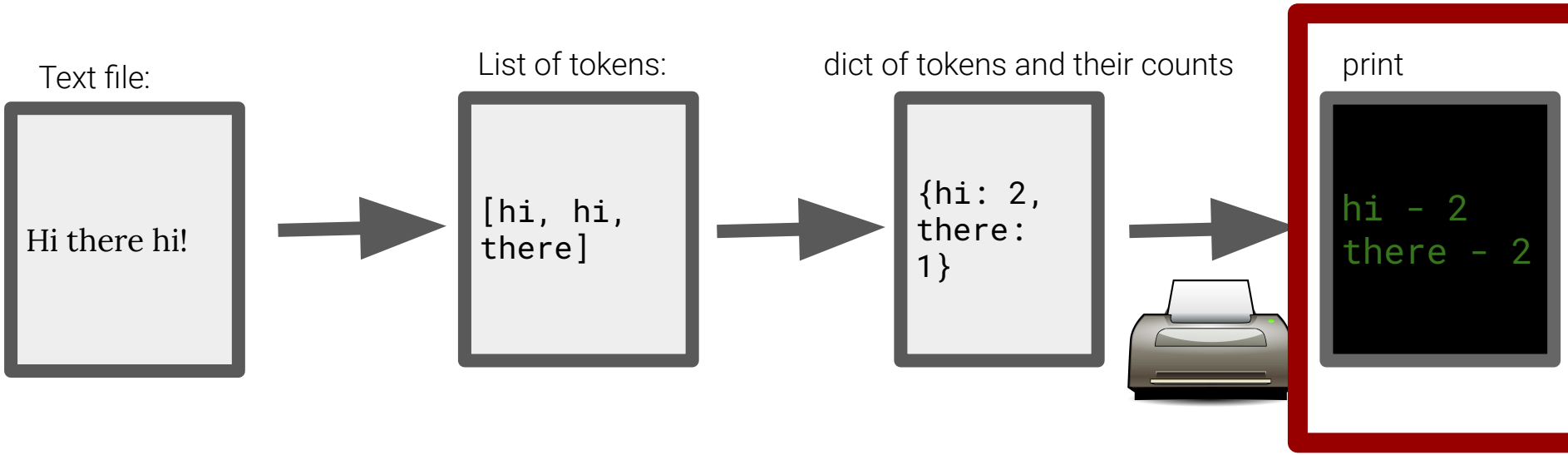
"a **token** is a sequence of **alphanumeric (English) characters**, independent of capitalization"

Hints:

Look at special sequences. Are there any that match our requirements?

Look at **flags**. These change the options in special sequences. What is the difference between "ASCII" and "Unicode?"

Part A



Print:

Sorting is going to be an important part of the homework, especially for efficiency in Part B. Is there a python built-in function we can use to sort in decreasing order of counts?

Next Week's Session

- Homework (meaty coverage!)
  - Part B
  - Optimization
- Come prepared with questions and ready to participate!

Recommended Homework

- Set up PyCharm and Conda
- Get through `tokenize` function in Part A
- (Optional) Set an alarm for 8:00am on Monday, October 4 to register for the STEM career fair!!