# CS453: Lab 1: Introduction to

# Natural Language Processing with Python

# Introduction

# All of us are familiar with text, since we read and write it every day. In NLP course, we will treat text as raw data for the programs we write, programs that manipulate and analyze it in a variety of interesting ways. But before we can do this, we have to get started with the Python interpreter. The objectives of this lab is to learn how can we automatically extract words from text and count its occurrence.

# Getting Started with Python

# Local python interpreter: you can install anaconda and use any IDE for python language such as Visual studio code, spider, Pycharm,…

# You can use online python interpreter to write jypyter notebook code such as google colab

# You can use online python interpreter to write python codes such as: replit, onlinegdp, online-python, ...

## 

## Getting Started with NLTK

## NLTK (Natural Language Toolkit) is the most popular API for NLP (Natural Language Processing) with Python. It is a powerful tool to preprocess text data for further analysis like with ML models. Before going further in this lab, you should install NLTK 3.0, downloadable for free from http://nltk.org/.

import nltk

## Counting Vocabulary

## Text data can be stored in a string in Python. To split string into words, you can use *split* method. The following code can be used to count the words in a text

text = 'After all is said and more is said than done'

tokens=text.split()

voc = set(tokens)

voc = sorted(voc)

print(len(voc))

## Output =

## 8

## Frequency Distributions

# It is useful to count the number of occurrence of each vocabulary in a text

fdist1 = nltk.FreqDist(tokens)

fdist1.plot()

fdist1.most\_common(2)

fdist1.tabulate()

## Output =

## 

## A graph with a line Description automatically generated

## Collocations and n-grams

# A collocation is a sequence of words that occur together unusually often.

list(nltk.ngrams(voc,2))

## Output =

## [('After', 'all'), ('all', 'and'), ('and', 'done'), ('done', 'is'), ('is', 'more'), ('more', 'said'), ('said', 'than')]

## Word Comparison Operators

## There is a common pattern for word comparison: [w for w in text if condition ], where condition is a Python "test" that yields either true or false

text1='this is a smaple text for detecting words ending with ing'

text1=text1.split()

sorted(w for w in set(text1) if w.endswith('ing'))

## Output =

## ['detecting', 'ending', 'ing']