Contents

1 Computational Approaches to Studies in Human Language & Technology Using Python and NLTK: Part 1

Facilitators: Richard Medina and Aitor Alvarez

In this hands-on workshop participants will learn to apply techniques and best practices for utilizing the Natural Language Toolkit (NLTK) Python package to support studies in Human Language & Technology. The workshop will begin with an introduction to common patterns and idioms for working with data structures in Python. The second portion of the workshop will introduce NLTK through examples. The remainder of the session will focus on 1 or 2 illustrative use cases for data driven language scholarship utilizing existing corpora and the NLTK. The goal of the workshop is to provide participants experience and guidance with integrating computational approaches in their work. Prior exposure to Python will be useful but not necessary.

1.1 Structure of a Computer Program/Script

(1) instructions are executed sequentially

(2) program is interpreted by a compiler

(3) Compiler translates instructions written in a programming language (E.g. Python) to instructions that are executable by the machine

1.2 Representations, Expressions and Translations

(a) expressed in a human language

"Print the message, 'Symphony No. 25 in G Minor is a fabulous way to start the day!' to my screen."

(b) expressed in Python as

```
print('Symphony No. 25 in G Minor is fabulous way to start the day!')
```

(c) expressed in the machine as

```
0x53 0x79 0x6d 0x70 0x68 0x6f 0x6e 0x79 0x20 0x4e ...
```

1.3 Computation

The expression of an algorithm (a set of steps that solve a problem) in an executable computer program.

Some of our problems are mundane such as:

'open a file and read the first word of each sentence.'

Other problems are more complex such as:

'load the text of 1000 journal articles to determine the dominant topics that exist within the corpus.'

Some computational problems are mathematical, some involve (semantic) inference and network centrality, some rely on statistical learning models and others on trained neural networks. There does not appear to be a shortage of problems.

1.4 Computer Programs/Scripts

(a) a script typically refers to a single file containing a sequence of instructions for computing a very specific task.

(b) a program or software typically refers to 1 or more separate files each containing instructions for addressing a specific part of a larger more complex problem.

(c) a framework is collection many files that are used to construct other programs

for this workshop, we are interested in the **scripting** level of computation

1.5 The Components of Script Writing

(a) a text editor for entering our instructions in a given programming language (E.g., Python)
(b) a compiler (also called an interpreter in the scripting context) that translates the script in the file and executes each instruction on the machine
(c) an input and output component for reading in data to a script and displaying information from the script

1.6 Components of a Python Script

(a) like any program, a script will contain variables, expressions, and commands

(b) each instruction is placed on a single line

(c) example of two different instructions. Each assigns the value on the right to the variable on the left. RIGHT-TO-LEFT evaluation.

```
x = 89
y = 42
```

these instructions are executed in the order that they appear in the script (x then y)

1.7 Variables

(a) like in algebra, variables provide a way to "name" and identify values that are needed in the script
(b) Python distinguishes between numbers (integers, floats), characters, and strings (collections of characters)

Example: Assign the value 11 to a variable x, then store the value of x doubled.

```
x = 11
y = x * 2
```

x and y are both storing an integer type

1.8 Variables: character and strings

```
x = 'a'
x = 'discovery'
```

when executed, the second line overwrites the value in x - 'a' is replaced with 'discovery'

1.9 Variables: floats (decimal numbers)

```
x = .7
y = 5/10 (stored as .5)
```

1.10 Expressions with operators

```
= is assignment not equality

a = (1 + 3 + 5) / 7
print(a)
```

common operators: *, /, +, -, /

1.11 Getting input from the keyboard (standard input)

using input() function in Python, you can enter data from the keyboard into your script

```
user_info = input('Enter your age then press <enter>:')
print(user_info)
```

all data input from the keyboard is treated like a string. You can convert that string into an integer using a *cast*.

```
case = input('Enter number of cases:')
case = int(case)
case = pow(case, 2)
print('Number of expected cases:', case)
```

1.12 Summation

summation refers to the idea of aggregation in which we continually add to a variable while also updating its value.

```
a = 0
a = a + 5
a = a + 10
```

What is the value of a after all statements have executed?

1.13 Exercise Set 1

Write a statement that prints 'hello world' to the screen.

Write a set of statements that sums the following values then prints (or displays) the sum. 5, 2, 2, 6, 11

Write a set of statements that prompts for two successive numbers (a and b) then prints the quotient of a divided by b.

1.14 Strings

a string is a sequence of individual characters. Python treats a string as if it were a "list" of characters in a specific order.

```
s = 'Centrality is a very important concept'
```

Using the type function to determine the type of a variable.

```
print(type(s))
```

Using various string functions:

```
len(s)
s.split() # splits a string into characters
```

1.15 Using Functions

input() is an example of a built-in function

methods are functions that are only callable on related types. For example the string method split() is only useful on string types. Example,

```
s = 'Now is the time.'
words = s.split()
```

Here's a list of String methods in Python

https://docs.python.org/3/library/stdtypes.html#textseq

Example functions:

```
# sort the items in the word list generated from the statement above

words.sort()
```

What is the problem with the sort method above? Here's another method for sorting that helps us address our problem.

```
sorted_words = sorted(words, key=lower)
```

See: https://docs.python.org/3/howto/sorting.html for more information on sorting lists in Python.

1.16 Lists

Lists are containers (or types) that allow collection of variables using a single reference. Please note the syntax in the following.

```
# A list of books titles there are currently two in the collection
titles = ['Graph Theory', 'Network Science']


# Add a new title:
titles.append('Trump Pandemics')
```

```
# Look at the first one:
print(titles[0])


# Look at the second one:
print(titles[1])
```

Strings are lists too! Here, you can access the first letter in a string.

```
s = 'algorithmic complexity'
print(s[0]) # prints the letter a
```

1.17 Lists and Iteration

Iterative structures go hand in hand with lists because they give us a way to handle items in a list of variable lengths. A loop for iterating over each title in the titles list will look like this:

```
titles = ['Graph Theory', 'Network Science']
for title in titles:
    print('List item =>', title)
```

If we wanted to process each item a particular way we could:

```
for title in titles:
    title = title.lower()
    print('List item =>', title)
```

1.18 Exercise Set 3: Summation Pattern With a List

Let's review the summation pattern again. In this case we will start with a list:

```
cases = [5, 2, 2, 6, 11]
```

Create a variable to store our sum.

```
sum = 0
```

Next, we will iterate over the list, adding each value to our sum.

```
for value in cases:
    sum = sum + value
```

Now print the sum to the display.

```
print(sum)
```

1.19 NLTK ==> Natural Language Toolkit

(a) a Python package containing useful functions and libraries for text processing. Essentially many operations for working with lists of words.

(b) provides access to sample corpora

(c) Reference: https://www.nltk.org/book/ch01.html

1.20 Setting up NLTK for First Explorations

(a) Install the nltk package
(b) Install the nltk sample corpora for the book

(c) Access one of the sample copora using 'text1', or 'text2', etc.

1.21 Example NLTK Library Functions

`text1.concordance()` locate context for a given word in a text

`similar() and common_context()` locate similar words in context and find where two or more words are collocated

`generate()` output random text based on a text's vocabulary

1.22 Frequencies

- length
- set
- 'lexical richness'
- word frequency using `count()/len()`

1.23 Exercise Set 3

- determine the following metrics from text4 (inaugural address corpus):

lexical diversity and the frequency of the word 'freedom'

1.24 Frequency Distribution Library

- E.g., `fd = FreqDist(text4)`
- `fd.most_common(n)` n highest frequency samples from text
- `fd['whale']` reports the number of occurrences of the word whale in the samples

1.25 Exercise Set 4

use the frequency distribution functions to determine the frequency (number of samples) of the word 'citizen' from text4