Cheat Sheet 3

1. Reading and Writing Files

1.1 **Reading** in files

- we can use the context manager to read in files
- what we need:
    - path to file to be read
    - reading mode (see: geeksforgeeks for more information)
    - encoding (normally `utf-8`)
    - variable to operate on the file content

```
with open (path, "r", encoding="utf-8") as file_in:
    text = file_in.read() # read  whole file as one string
    text_list = file_in.readlines() # get a list of lines from file
    text_list_no_breaks = file_in.read().splitlines() # get list of lines from file without line br
```

We then can go on with processing the text, e.g. tokenize, get word frequencies etc.

For large files it is advisable to use an iterator to read a file line by line (see DigitalOcean for more information):

```
lines = []
with open (path, "r", encoding="utf-8") as file_in:
    for line in file_in:
    lines.append(line)
```

1.2. **Writing** output to a file

- we can use the context manager to write objects to a file, e.g. a dictionary
- what we need:
    - path (location) for file to be saved
    - writing mode (see: geeksforgeeks for more information)
    - encoding (normally `utf-8`)
    - variable to operate on the file content

```
my_dict ={"apple": "green", "banana":"yellow", "mango":"yellow", "orange": "orange"}

with open ("fruitcolors.txt", "w", encoding="utf-8") as file_out:
    for key, value in my_dict.items():
      file_out.write(f"{key} - {value}\n")
```

file content:

apple - green
banana - yellow
mango - yellow
orange - orange

2. Dictionaries

Syntax:
dictionary = {"key1" : "value1", "key2" : "values2" }
key + value = item

```
my_dict = {
  "name": "Chomsky",
  "firstname": "Noam",
  "year of birth": 1928
}
print(my_dict)
```

> {'name': 'Chomsky', 'firstname': 'Noam', 'year of birth': 1928}

Access a dictionary value:

```
print(my_dict["name"])
```

> Chomsky

3. Counting with Counter

- we can use the library `Counter` to count elements of a list within a dictionary
- what we need:
  - library import
  - list of objects (preferably strings)
  - variable for saving dictionary and its content

```
from collections import Counter

sample = "I go up up up without looking down down down."

counted= Counter(sample.split(" ")) # split sentence to list at whitespace
print(counted)
```

> Counter({'up': 3, 'down': 2, 'I': 1, 'go': 1, 'without': 1, 'looking': 1, 'down.': 1})

4. Working with spacy

- we can use the library `spacy` as a powerful tool to tokenize and annotate our data
- what we need:
  - library import
  - loading the language model
  - variable for saving the annotated tokens

```
import spacy

text = "Hello, Jim, I am Lynn! It is nice to meet you. I like you!"
nlp = spacy.load('en_core_web_sm') # load language model
annotated_text = nlp(text) # tokenization and annotation
```

3.1 Using spacy's annotation for **POS-Tags**

- get frequency of each personal pronoun in our variable `text`
- write each pronoun and its frequency to a file

```
PRONOUN_TAG = "PRON"
pronouns = []
# get list of all pronouns
for elem in annotated_text:
    if elem.pos_ == PRONOUN_TAG:
        pronouns.append(elem.lemma_)

# count frequencies
pronoun_counts = Counter(pronouns)

# write pronouns and their frequencies to file
with open ("word_freq_nouns.txt", "w", encoding="utf-8") as file_out:
  for key, value in pronoun_counts.items(): # access dictionary keys and values
    file_out.writelines(f"{key}: {value}\n")
```

file content:

```
I: 2
it: 1
you: 2
```

3.2 Using spacy's annotation for **lemmata**

- get number of types and tokens in variable sample
- calculate type-token ratio

```
import spacy

sample = "I go up up up without looking down down down."
nlp = spacy.load('en_core_web_sm') # load language model
annotated_text = nlp(sample) # tokenization and annotation

num_tokens = len(annotated_text) # get number of tokens
tokens_to_types = [] # temporary list

# get lemma of every token
for elem in annotated_text:
   tokens_to_types.append(elem.lemma_)

types = set(tokens_to_types) # get unique lemmata
num_types = len(types) # get number of types

ratio = num_types/num_tokens *100 # calculate ratio
print(round(ratio, 3))
```

```
63.636
```

5. Some additional hacks

Remove punctuation marks from a list of tokens:

```
# for loop with list comprehension and condition
tokenized_clean = [word for word in tokenized if word.isalpha()]
```

Lower all words in a list of tokens:

```
# lower words to remove duplicates (but be careful with proper nouns)
tokenized_clean_lower = [word.lower() for word in tokenized_clean]
```

Count words using a dictionary:

```
# count word frequencies
word_freq = {}
for word in tokenized_clean:
    if word in word_freq:
        word_freq[word] += 1
    else:
        word_freq[word] = 1
```

Get the most frequent word in a dictionary with word counts:

```
# count word frequencies
for key, value in word_freq.items():
    if value == max(word_freq.values()):
        print(f"{key}: {value}")
```

Get the least frequent word in a dictionary with word counts:

```
# count word frequencies
for key, value in word_freq.items():
    if value == min(word_freq.values()):
        print(f"{key}: {value}")
```