IST664 - Homework 2

Originality assertion: All of the text and comments in this file are my original work (except for template items written by the instructor). All of the code in this file is my work, except where I give credit to another source. By adding my name below, I affirm this originality assertion.

My name: Ben Tisinger_________

Task 1: Use Beautiful Soup

```
# Import Beautiful Soup for its web scraping capabilities
import bs4 as bs
import urllib.request # For retrieving from web pages
import re # Regular expressions
import spacy
```

```
# Change this URL to a Wikipedia article of your choice
wiki_url_1 = 'https://en.wikipedia.org/wiki/Android_Oreo'

scraped_data = urllib.request.urlopen(wiki_url_1)

type(scraped_data) # A response object for a web page

```

```
http.client.HTTPResponse
def __init__(sock, debuglevel=0, method=None, url=None)

/usr/lib/python3.11/http/client.py
Base class for buffered IO objects.

The main difference with RawIOBase is that the read() method
supports omitting the size argument, and does not have a default
implementation that defers to readinto().
```

```
# Now extract the text from the article and organize into paragraphs
article = scraped_data.read() # Extract the data from the response object

parsed_article = bs.BeautifulSoup(article,'lxml') # Use lxml as the back end parser

paragraphs = parsed_article.find_all('p')

article_text = ""

for p in paragraphs:
    article_text += p.text

len(article_text)
```

```
8319
```

Task 2: Use RegEx to remove Wikipedia References and Extra Space

```
# Put your code for task 2 here

remove_wiki_ref = re.sub(r'\[\d+\]', '', article_text)
article_text = re.sub(r'\s+', ' ', remove_wiki_ref)
len(article_text)

```

```
8098
```

Task 3: Tokenize with spaCy

```
from spacy import displacy

nlp = spacy.load("en_core_web_sm")

my_article = nlp(article_text)
```

```

len(my_article) # Length in tokens
```

```
1507
```

```
# Here's one way to work with individual sentences:
my_spans = list(my_article.sents)

my_spans[1] # Let's view just the first sentence
```

```
It was initially unveiled as an alpha quality developer preview in March 2017 and later made available to the public, on August 21,
2017.
```

```
my_spans = list(my_article.sents)
```

```
Located outside the park's railroad tracks and named after a Georgia mining town in the late 19th century, Lickskillet added three new
rides — the Spindle Top (a Rotor flat ride, the Wheel Burrow (a Chance Tumbler) and the Sky Buckets, the park's second cable car ride —
along with several craft shops and a shootout show performed on the street.
```

Task 4: Use displaCy to show named entities for an early sentence

```
# Put your code for task 4 here

displacy.render(my_spans[1], style="ent", jupyter=True)

```

It was initially unveiled as an alpha quality developer preview in March 2017 DATE and later made available to the public, on August 21, 2017 DATE .

For the following tasks, create a pandas dataframe and store each disovered token in an appropriately-named column.

Task 5: Find the Root Verbs for each span

Make a sentence by sentence list of all of the root verbs.

```
my_spans[1].root # The span object has an attribute that points to the root token
```

```
unveiled
```

```
len(my_spans)
```

```
50
```

```
import pandas as pd

data_pd  = []

for i, sent in enumerate(my_spans):
    root = sent.root
    data_pd.append({
        "sentence_number": i + 1,
        "root_text": root.text,

    })

wiki_1 = pd.DataFrame(data_pd)
wiki_1

#Used Source - https://www.phind.com/
```

	sentence_number	root_text
0	1	is
1	2	unveiled
2	3	contains
3	4	introduces
4	5	ran
5	6	codenamed
6	7	released
7	8	released
8	9	released
9	10	finalized
10	11	released
11	12	released
12	13	unveiled
13	14	made
14	15	were
15	16	released
16	17	snoozed
17	18	orders
18	19	contains
19	20	features
20	21	set
21	22	supports
22	23	limited
23	24	adds
24	25	features
25	26	contains
26	27	specify
27	28	adds
28	29	supports
29	30	introduced
30	31	is
31	32	revised
32	33	made
33	34	allows
34	35	support
35	36	modified
36	37	reduces
37	38	perform
38	39	reboot
39	40	introduces
40	41	designed
41	42	intended
42	43	has
43	44	highlight
44	45	menu
45	46	modularized

46	47	made
47	48	sideloaded
48	49	implemented
49	50	includes

Next steps: Generate code with wiki_1 View recommended plots New interactive sheet

Task 6: Find the Subjects of Each Span

Put a column in your pandas dataframe and fill it with the subjects from each span.

```
# Here's one simple way to find the subject of a sentence
for tok in my_spans[1]:
  if tok.dep_ == "nsubj":
    print(tok)
```

```
subjects = []


for sent in my_spans:
    subj = "NA"
    for tok in sent:
        if tok.dep_ == "nsubj":
            subj = tok.text
            break
    subjects.append(subj)

wiki_1["subjects"] = subjects
wiki_1
```

	sentence_number	root_text	subjects
0	1	is	Oreo
1	2	unveiled	NA
2	3	contains	It
3	4	introduces	Oreo
4	5	ran	Oreo
5	6	codenamed	NA
6	7	released	Google
7	8	released	NA
8	9	released	NA
9	10	finalized	DP3
10	11	released	which
11	12	released	NA
12	13	unveiled	which
13	14	made	NA
14	15	were	Compact
15	16	released	which
16	17	snoozed	NA
17	18	orders	NA
18	19	contains	Oreo
19	20	features	app
20	21	set	NA
21	22	supports	update
22	23	limited	NA
23	24	adds	Oreo
24	25	features	Runtime
25	26	contains	Oreo
26	27	specify	Apps
27	28	adds	Oreo
28	29	supports	Oreo
29	30	introduced	which
30	31	is	functionality
31	32	revised	NA
32	33	made	NA
33	34	allows	architecture
34	35	support	devices
35	36	modified	NA
36	37	reduces	This
37	38	perform	Oreo
38	39	reboot	device
39	40	introduces	update
40	41	designed	it
41	42	intended	NA
42	43	has	mode
43	44	highlight	Store
44	45	menu	interface
45	46	modularized	NA

46	47	made	NA
47	48	sideloaded	re
48	49	implemented	NA
49	50	includes	boot

Next steps: Generate code with `wiki_1` View recommended plots New interactive sheet

Task 7: Find the Direct Objects of Each Span

Put a column in your pandas dataframe and fill it with the direct objects from each span.

```
for tok in my_spans[1]:
  if tok.dep_ == "dobj":
    print(tok)
```

```
direct_objects = []

for sent in my_spans:
    direct_obj = "NA"
    for tok in sent:
        if tok.dep_ == "dobj":
            direct_obj = tok.text
            break
    direct_objects.append(direct_obj)

wiki_1["direct_objects"] = direct_objects
wiki_1
```

	sentence_number	root_text	subjects	direct_objects
0	1	is	Oreo	O
1	2	unveiled	NA	NA
2	3	contains	It	number
3	4	introduces	Oreo	features
4	5	ran	Oreo	updates
5	6	codenamed	NA	NA
6	7	released	Google	preview
7	8	released	NA	NA
8	9	released	NA	version
9	10	finalized	DP3	API
10	11	released	which	behaviors
11	12	released	NA	NA
12	13	unveiled	which	factory
13	14	made	NA	NA
14	15	were	Compact	NA
15	16	released	which	fixes
16	17	snoozed	NA	NA
17	18	orders	NA	alerts
18	19	contains	Oreo	support
19	20	features	app	design
20	21	set	NA	NA
21	22	supports	update	display
22	23	limited	NA	NA
23	24	adds	Oreo	support
24	25	features	Runtime	improvements
25	26	contains	Oreo	limits
26	27	specify	Apps	icons
27	28	adds	Oreo	support
28	29	supports	Oreo	emoji
29	30	introduced	which	figures
30	31	is	functionality	NA
31	32	revised	NA	hardware
32	33	made	NA	NA
33	34	allows	architecture	modifications
34	35	support	devices	interface
35	36	modified	NA	files
36	37	reduces	This	requirements
37	38	perform	Oreo	system
38	39	reboot	device	reset
39	40	introduces	update	API
40	41	designed	it	mode
41	42	intended	NA	NA
42	43	has	mode	optimizations
43	44	highlight	Store	apps
44	45	menu	interface	prominence
45	46	modularized	NA	footprint

46	47	made	NA	NA
47	48	sideloaded	re	features
48	49	implemented	NA	installation
49	50	includes	boot	feature

Next steps: Generate code with wiki_1 View recommended plots New interactive sheet

Task 8: Find the first Named Entity (if any) from Each Span

Put two columns in your pandas dataframe. Find the first named entity in a span (if any) and record the ent_type_ and the corresponding token in those two columns.

```
for tok in my_spans[1]:
  if len(tok.ent_type_) > 0:
    print(tok.ent_type_, tok)
```

```
DATE March
DATE 2017
DATE August
DATE 21
DATE ,
DATE 2017
```

```
first_named_entity = []
first_named_entity_type = []

for sent in my_spans:
    first_ent = "NA"
    first_ent_type = "NA"
    for tok in sent:
        if tok.ent_type_:
            first_ent = tok.text
            first_ent_type = tok.ent_type_
            break

    first_named_entity.append(first_ent)
    first_named_entity_type.append(first_ent_type)


wiki_1["first_named_entity"] = first_named_entity
wiki_1["first_named_entity_type"] = first_named_entity_type
wiki_1

#Used Source - https://www.phind.com/
```

	sentence_number	root_text	subjects	direct_objects	first_named_entity	first_named_entity_type
0	1	is	Oreo	O	Android	ORG
1	2	unveiled	NA	NA	March	DATE
2	3	contains	It	number	5	CARDINAL
3	4	introduces	Oreo	features	Android	ORG
4	5	ran	Oreo	updates	January	DATE
5	6	codenamed	NA	NA	Android	ORG
6	7	released	Google	preview	March	DATE
7	8	released	NA	NA	second	ORDINAL
8	9	released	NA	version	third	ORDINAL
9	10	finalized	DP3	API	API	ORG
10	11	released	which	behaviors	July	DATE
11	12	released	NA	NA	Android	ORG
12	13	unveiled	which	factory	Chelsea	ORG
13	14	made	NA	NA	Pixel	PERSON
14	15	were	Compact	NA	Sony	ORG
15	16	released	which	fixes	Android	ORG
16	17	snoozed	NA	NA	NA	NA
17	18	orders	NA	alerts	NA	NA
18	19	contains	Oreo	support	Android	ORG
19	20	features	app	design	NA	NA
20	21	set	NA	NA	NA	NA
21	22	supports	update	display	Android	ORG
22	23	limited	NA	NA	one	CARDINAL
23	24	adds	Oreo	support	Android	ORG
24	25	features	Runtime	improvements	ART	ORG
25	26	contains	Oreo	limits	Android	ORG
26	27	specify	Apps	icons	Apps	PERSON
27	28	adds	Oreo	support	Android	ORG
28	29	supports	Oreo	emoji	Android	ORG
29	30	introduced	which	figures	KitKat	ORG
30	31	is	functionality	NA	Android	GPE
31	32	revised	NA	hardware	Android	ORG
32	33	made	NA	NA	Android	ORG
33	34	allows	architecture	modifications	Project	ORG
34	35	support	devices	interface	NA	NA
35	36	modified	NA	files	Android	GPE
36	37	reduces	This	requirements	NA	NA
37	38	perform	Oreo	system	Android	ORG
38	39	reboot	device	reset	NA	NA
39	40	introduces	update	API	Android	ORG
40	41	designed	it	mode	API	ORG
41	42	intended	NA	NA	Android	ORG
42	43	has	mode	optimizations	Data	FAC
43	44	highlight	Store	apps	The	PRODUCT
44	45	menu	interface	prominence	four	QUANTITY
45	46	modularized	NA	footprint	Google	ORG

46	47	made	NA	NA	Android	ORG
47	48	sideloaded	re	features	Google	ORG
48	49	implemented	NA	installation	the	ORG
49	50	includes	boot	feature	a	LAW

Next steps: Generate code with wiki_1 View recommended plots New interactive sheet

Task 9: Show the shape of your data frame

```
# Add code to show the shape of your data frame
wiki_1.shape
```

```
(50, 6)
```

Task 10: Show the unique set of elements from each column in the data frame

```
# Use set() on the data in each column to show the list of unique elements
for col in wiki_1.columns:
    unique_elements = set(wiki_1[col])
    print(f"Unique elements in column '{col}':")
    print(unique_elements)
    print()
```

```
Unique elements in column 'sentence_number':
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,

Unique elements in column 'root_text':
{'sideloaded', 'menu', 'allows', 'adds', 'is', 'were', 'codenamed', 'introduces', 'orders', 'revised', 'supports', 'contains', 'designed

Unique elements in column 'subjects':
{'Google', 'Compact', 'Apps', 'This', 'which', 'device', 're', 'It', 'app', 'Store', 'update', 'mode', 'NA', 'it', 'boot', 'functionalit

Unique elements in column 'direct_objects':
{'footprint', 'emoji', 'installation', 'feature', 'fixes', 'behaviors', 'files', 'reset', 'version', 'interface', 'features', 'design',

Unique elements in column 'first_named_entity':
{'Google', 'ART', 'Pixel', 'Chelsea', 'Apps', 'KitKat', 'second', 'July', '5', 'January', 'four', 'NA', 'the', 'one', 'Sony', 'March', '

Unique elements in column 'first_named_entity_type':
{'PERSON', 'NA', 'DATE', 'PRODUCT', 'ORG', 'FAC', 'ORDINAL', 'GPE', 'QUANTITY', 'CARDINAL', 'LAW'}
```

Task 11: Show your data frame

```
# Type the name of your data frame on a line by itself to display it
```

```
wiki_1
```

	sentence_number	root_text	subjects	direct_objects	first_named_entity	first_named_entity_type
0	1	is	Oreo	O	Android	ORG
1	2	unveiled	NA	NA	March	DATE
2	3	contains	It	number	5	CARDINAL
3	4	introduces	Oreo	features	Android	ORG
4	5	ran	Oreo	updates	January	DATE
5	6	codenamed	NA	NA	Android	ORG
6	7	released	Google	preview	March	DATE
7	8	released	NA	NA	second	ORDINAL
8	9	released	NA	version	third	ORDINAL
9	10	finalized	DP3	API	API	ORG
10	11	released	which	behaviors	July	DATE
11	12	released	NA	NA	Android	ORG
12	13	unveiled	which	factory	Chelsea	ORG
13	14	made	NA	NA	Pixel	PERSON
14	15	were	Compact	NA	Sony	ORG
15	16	released	which	fixes	Android	ORG
16	17	snoozed	NA	NA	NA	NA
17	18	orders	NA	alerts	NA	NA
18	19	contains	Oreo	support	Android	ORG
19	20	features	app	design	NA	NA
20	21	set	NA	NA	NA	NA
21	22	supports	update	display	Android	ORG
22	23	limited	NA	NA	one	CARDINAL
23	24	adds	Oreo	support	Android	ORG
24	25	features	Runtime	improvements	ART	ORG
25	26	contains	Oreo	limits	Android	ORG
26	27	specify	Apps	icons	Apps	PERSON
27	28	adds	Oreo	support	Android	ORG
28	29	supports	Oreo	emoji	Android	ORG
29	30	introduced	which	figures	KitKat	ORG
30	31	is	functionality	NA	Android	GPE
31	32	revised	NA	hardware	Android	ORG
32	33	made	NA	NA	Android	ORG
33	34	allows	architecture	modifications	Project	ORG
34	35	support	devices	interface	NA	NA
35	36	modified	NA	files	Android	GPE
36	37	reduces	This	requirements	NA	NA
37	38	perform	Oreo	system	Android	ORG
38	39	reboot	device	reset	NA	NA
39	40	introduces	update	API	Android	ORG
40	41	designed	it	mode	API	ORG
41	42	intended	NA	NA	Android	ORG
42	43	has	mode	optimizations	Data	FAC
43	44	highlight	Store	apps	The	PRODUCT
44	45	menu	interface	prominence	four	QUANTITY
45	46	modularized	NA	footprint	Google	ORG

46	47	made	NA	NA	Android	ORG
47	48	sideloaded	re	features	Google	ORG
48	49	implemented	NA	installation	the	ORG
49	50	includes	boot	feature	a	LAW

Next steps: Generate code with wiki_1 View recommended plots New interactive sheet

Don't forget to also process your second article in the same fashion as you did for the first one.

```
# Code for processing the second article starts here
```

```
##########################################################################################################################
```

```
# Change this URL to a Wikipedia article of your choice
wiki_url_2 = 'https://en.wikipedia.org/wiki/Six_Flags_Over_Georgia'

scraped_data = urllib.request.urlopen(wiki_url_2)

type(scraped_data) # A response object for a web page

```

```
http.client.HTTPResponse
def __init__(sock, debuglevel=0, method=None, url=None)

/usr/lib/python3.11/http/client.py
Base class for buffered IO objects.

The main difference with RawIOBase is that the read() method
supports omitting the size argument, and does not have a default
implementation that defers to readinto().
```

```
# Now extract the text from the article and organize into paragraphs
article = scraped_data.read() # Extract the data from the response object

parsed_article = bs.BeautifulSoup(article,'lxml') # Use lxml as the back end parser

paragraphs = parsed_article.find_all('p')

article_text = ""

for p in paragraphs:
    article_text += p.text

len(article_text)
```

```
11011
```

```
# Put your code for task 2 here

remove_wiki_ref = re.sub(r'\[\d+\]', '', article_text)
article_text = re.sub(r'\s+', ' ', remove_wiki_ref)
len(article_text)

```

```
10896
```

```
from spacy import displacy

nlp = spacy.load("en_core_web_sm")

my_article = nlp(article_text)

len(my_article) # Length in tokens
```

```
2100
```

```
# Here's one way to work with individual sentences:
my_spans = list(my_article.sents)

```

```
my_spans[1] # Let's view just the first sentence
```

```
Opened in 1967, it is the second park in the Six Flags chain following the original Six Flags Over Texas, which opened in 1961.
```

```
# Put your code for task 4 here

displacy.render(my_spans[1], style="ent", jupyter=True)
```

Opened in 1967 **DATE** , it is the second **ORDINAL** park in the Six **CARDINAL** Flags chain following the original Six **CARDINAL** Flags Over Texas **GPE**

```
my_spans[1].root # The span object has an attribute that points to the root token
```

```
is
```

```
len(my_spans)
```

```
72
```

```
import pandas as pd

data_pd  = []

for i, sent in enumerate(my_spans):
    root = sent.root
    data_pd.append({
        "sentence_number": i + 1,
        "root_text": root.text,

    })

wiki_2 = pd.DataFrame(data_pd)
wiki_2

#Used Source - https://www.phind.com/
```

	sentence_number	root_text
0	1	is
1	2	is
2	3	is
3	4	features
4	5	began
...	...	...
67	68	occurred
68	69	managed
69	70	said
70	71	began
71	72	fired

72 rows × 2 columns

Next steps: Generate code with wiki_2 View recommended plots New interactive sheet

```
# Here's one simple way to find the subject of a sentence
for tok in my_spans[1]:
  if tok.dep_ == "nsubj":
    print(tok)
```

```
it
which
```

```
subjects = []


for sent in my_spans:
    subj = "NA"
    for tok in sent:
        if tok.dep_ == "nsubj":
            subj = tok.text
            break
    subjects.append(subj)

wiki_2["subjects"] = subjects
wiki_2
```

	sentence_number	root_text	subjects
0	1	is	Flags
1	2	is	it
2	3	is	Flags
3	4	features	it
4	5	began	Wynne
...	...	...	...
67	68	occurred	bulk
68	69	managed	park
69	70	said	police
70	71	began	people
71	72	fired	officer

72 rows × 3 columns

Next steps: Generate code with wiki_2 View recommended plots New interactive sheet

```
for tok in my_spans[1]:
  if tok.dep_ == "dobj":
    print(tok)
```

```
direct_objects = []

for sent in my_spans:
    direct_obj = "NA"
    for tok in sent:
        if tok.dep_ == "dobj":
            direct_obj = tok.text
            break
    direct_objects.append(direct_obj)

wiki_2["direct_objects"] = direct_objects
wiki_2
```

	sentence_number	root_text	subjects	direct_objects
0	1	is	Flags	NA
1	2	is	it	NA
2	3	is	Flags	NA
3	4	features	it	themes
4	5	began	Wynne	NA
...	...	...	...	...
67	68	occurred	bulk	NA
68	69	managed	park	damage
69	70	said	police	day
70	71	began	people	CCPD
71	72	fired	officer	weapon

72 rows × 4 columns

Next steps: Generate code with wiki_2 View recommended plots New interactive sheet

```
for tok in my_spans[1]:
  if len(tok.ent_type_) > 0:
    print(tok.ent_type_, tok)
```

```
DATE 1967
ORDINAL second
CARDINAL Six
CARDINAL Six
GPE Texas
DATE 1961
```

```
first_named_entity = []
first_named_entity_type = []

for sent in my_spans:
    first_ent = "NA"
    first_ent_type = "NA"
    for tok in sent:
        if tok.ent_type_:
            first_ent = tok.text
            first_ent_type = tok.ent_type_
            break

    first_named_entity.append(first_ent)
    first_named_entity_type.append(first_ent_type)


wiki_2["first_named_entity"] = first_named_entity
wiki_2["first_named_entity_type"] = first_named_entity_type
wiki_2

#Used Source - https://www.phind.com/
```

	sentence_number	root_text	subjects	direct_objects	first_named_entity	first_named_entity_type
0	1	is	Flags	NA	Six	CARDINAL
1	2	is	it	NA	1967	DATE
2	3	is	Flags	NA	Six	CARDINAL
3	4	features	it	themes	Six	CARDINAL
4	5	began	Wynne	NA	Six	CARDINAL
...	...	...	...	...	...	...
67	68	occurred	bulk	NA	the	DATE
68	69	managed	park	damage	NA	NA
69	70	said	police	day	opening	DATE
70	71	began	people	CCPD	the	ORG
71	72	fired	officer	weapon	one	CARDINAL

72 rows × 6 columns

Next steps: Generate code with wiki_2 View recommended plots New interactive sheet

```
# Add code to show the shape of your data frame
wiki_2.shape
```

```
(72, 6)
```

```
# Use set() on the data in each column to show the list of unique elements
for col in wiki_2.columns:
    unique_elements = set(wiki_2[col])
    print(f"Unique elements in column '{col}':")
    print(unique_elements)
    print()
```

```
Unique elements in column 'sentence_number':
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,

Unique elements in column 'root_text':
{'based', 'prides', 'terminated', 'owned', 'began', 'caused', 'is', 'left', 'proposed', 'were', 'rank', 'was', 'fired', 'upgraded', 'app
```