CS110 Lab Assignment: A day in the life of a Minervan

```
# A Tweet object represents one tweet in this assignment.
class Tweet:
    '''
    Tweet: Tweet object represents one tweet.
    Attributes:
    tweet_id (str): Tweet's unique identifier.
    text (str): Tweet's text.
    hashtags (list): Hashtags in the tweet.
    urls (list): URLs in the tweet.
    user_id (str): Tweet author's unique identifier.
    is_retweet (bool): True if this tweet is a retweet.
    retweeted_tweet_id (str): If this is a retweet, the id of the original tweet.
    retweeted_user_id (str): If this is a retweet, the id of the original tweet's author.
    mentioned_ids (list): Ids of users mentioned in the tweet.
    reply_to_id (str): If this is a reply, the id of the tweet being replied to.
    reply_to_user_id (str): If this is a reply, the id of the user being replied to.
    '''
    def __init__(self, tweet_id, text, hashtags, urls, user_id,
                 is_retweet, retweeted_tweet_id,
                 retweeted_user_id, mentioned_ids,
                 reply_to_id, reply_to_user_id):
        '''
        Initialize a Tweet object.
        '''
        self.tweet_id = tweet_id
        self.text = text
        self.hashtags = hashtags
        self.urls = urls
        self.user_id = user_id
        self.is_retweet = is_retweet
        self.retweeted_tweet_id = retweeted_tweet_id
        self.retweeted_user_id = retweeted_user_id
        self.mentioned_ids = mentioned_ids
        self.reply_to_id = reply_to_id
        self.reply_to_user_id = reply_to_user_id
```

01 Setting up

Part 01

```
# A Tweet object represents one tweet.
class Tweet:
    '''
    Tweet: Tweet object represents one tweet.
    Attributes:
    tweet_id (str): Tweet's unique identifier.
    text (str): Tweet's text.
    hashtags (list): Hashtags in the tweet.
    urls (list): URLs in the tweet.
    user_id (str): Tweet author's unique identifier.
    is_retweet (bool): True if this tweet is a retweet.
    retweeted_tweet_id (str): If this is a retweet, the id of the original tweet.
    retweeted_user_id (str): If this is a retweet, the id of the original tweet's author.
    mentioned_ids (list): Ids of users mentioned in the tweet.
    reply_to_id (str): If this is a reply, the id of the tweet being replied to.
    reply_to_user_id (str): If this is a reply, the id of the user being replied to.
    '''
    def __init__(self, tweet_id, text, hashtags, urls, user_id,
                 is_retweet, retweeted_tweet_id,
                 retweeted_user_id, mentioned_ids,
                 reply_to_id, reply_to_user_id):
        '''
        Initialize a Tweet object.
        '''
        self.tweet_id = tweet_id
        self.text = text
        self.hashtags = hashtags
        self.urls = urls
        self.user_id = user_id
        self.is_retweet = is_retweet
        self.retweeted_tweet_id = retweeted_tweet_id
        self.retweeted_user_id = retweeted_user_id
        self.mentioned_ids = mentioned_ids
        self.reply_to_id = reply_to_id
        self.reply_to_user_id = reply_to_user_id
```

```
# A Tweet object represents one tweet.
class Tweet:
    '''
    Tweet: Tweet object represents one tweet.
    Attributes:
    tweet_id (str): Tweet's unique identifier.
    text (str): Tweet's text.
    hashtags (list): Hashtags in the tweet.
    urls (list): URLs in the tweet.
    user_id (str): Tweet author's unique identifier.
    is_retweet (bool): True if this tweet is a retweet.
    retweeted_tweet_id (str): If this is a retweet, the id of the original tweet.
    retweeted_user_id (str): If this is a retweet, the id of the original tweet's author.
    mentioned_ids (list): Ids of users mentioned in the tweet.
    reply_to_id (str): If this is a reply, the id of the tweet being replied to.
    reply_to_user_id (str): If this is a reply, the id of the user being replied to.
    '''
    def __init__(self, tweet_id, text, hashtags, urls, user_id,
                 is_retweet, retweeted_tweet_id,
                 retweeted_user_id, mentioned_ids,
                 reply_to_id, reply_to_user_id):
        '''
        Initialize a Tweet object.
        '''
        self.tweet_id = tweet_id
        self.text = text
        self.hashtags = hashtags
        self.urls = urls
        self.user_id = user_id
        self.is_retweet = is_retweet
        self.retweeted_tweet_id = retweeted_tweet_id
        self.retweeted_user_id = retweeted_user_id
        self.mentioned_ids = mentioned_ids
        self.reply_to_id = reply_to_id
        self.reply_to_user_id = reply_to_user_id
```

Part 02

```
# A Tweet object represents one tweet.
class Tweet:
    '''
    Tweet: Tweet object represents one tweet.
    Attributes:
    tweet_id (str): Tweet's unique identifier.
    text (str): Tweet's text.
    hashtags (list): Hashtags in the tweet.
    urls (list): URLs in the tweet.
    user_id (str): Tweet author's unique identifier.
    is_retweet (bool): True if this tweet is a retweet.
    retweeted_tweet_id (str): If this is a retweet, the id of the original tweet.
    retweeted_user_id (str): If this is a retweet, the id of the original tweet's author.
    mentioned_ids (list): Ids of users mentioned in the tweet.
    reply_to_id (str): If this is a reply, the id of the tweet being replied to.
    reply_to_user_id (str): If this is a reply, the id of the user being replied to.
    '''
    def __init__(self, tweet_id, text, hashtags, urls, user_id,
                 is_retweet, retweeted_tweet_id,
                 retweeted_user_id, mentioned_ids,
                 reply_to_id, reply_to_user_id):
        '''
        Initialize a Tweet object.
        '''
        self.tweet_id = tweet_id
        self.text = text
        self.hashtags = hashtags
        self.urls = urls
        self.user_id = user_id
        self.is_retweet = is_retweet
        self.retweeted_tweet_id = retweeted_tweet_id
        self.retweeted_user_id = retweeted_user_id
        self.mentioned_ids = mentioned_ids
        self.reply_to_id = reply_to_id
        self.reply_to_user_id = reply_to_user_id
```

	DESCRIPTION	DIFFICULTY	REQUIREMENTS	TIME
1	Write your first Minervan tweet	1%	0	1 min
2	Find your tweet in the data	2%	1	2 min
3	Find a hashtag in Chamberlain	4%	2	5 min
4	Find a hashtag in Hoover	6%	3	10 min
5	Find a hashtag in Lathrop	8%	4	15 min
6	Find a hashtag in Lathrop	10%	5	20 min
7	Find a hashtag in Lathrop	12%	6	25 min
8	Find a hashtag in Lathrop	14%	7	30 min
9	Find an hour-long hashtag	16%	8	35 min
10	Find an hour-long hashtag	18%	9	40 min

```
# A Tweet object represents one tweet.
class Tweet:
    '''
    Tweet: Tweet object represents one tweet.
    Attributes:
    tweet_id (str): Tweet's unique identifier.
    text (str): Tweet's text.
    hashtags (list): Hashtags in the tweet.
    urls (list): URLs in the tweet.
    user_id (str): Tweet author's unique identifier.
    is_retweet (bool): True if this tweet is a retweet.
    retweeted_tweet_id (str): If this is a retweet, the id of the original tweet.
    retweeted_user_id (str): If this is a retweet, the id of the original tweet's author.
    mentioned_ids (list): Ids of users mentioned in the tweet.
    reply_to_id (str): If this is a reply, the id of the tweet being replied to.
    reply_to_user_id (str): If this is a reply, the id of the user being replied to.
    '''
    def __init__(self, tweet_id, text, hashtags, urls, user_id,
                 is_retweet, retweeted_tweet_id,
                 retweeted_user_id, mentioned_ids,
                 reply_to_id, reply_to_user_id):
        '''
        Initialize a Tweet object.
        '''
        self.tweet_id = tweet_id
        self.text = text
        self.hashtags = hashtags
        self.urls = urls
        self.user_id = user_id
        self.is_retweet = is_retweet
        self.retweeted_tweet_id = retweeted_tweet_id
        self.retweeted_user_id = retweeted_user_id
        self.mentioned_ids = mentioned_ids
        self.reply_to_id = reply_to_id
        self.reply_to_user_id = reply_to_user_id
```

	DESCRIPTION	DIFFICULTY	REQUIREMENTS	TIME
1	Write your first Minervan tweet	1%	0	1 min
2	Find your tweet in the data	2%	1	2 min
3	Find a hashtag in Chamberlain	4%	2	5 min
4	Find a hashtag in Hoover	6%	3	10 min
5	Find a hashtag in Lathrop	8%	4	15 min
6	Find a hashtag in Lathrop	10%	5	20 min
7	Find a hashtag in Lathrop	12%	6	25 min
8	Find a hashtag in Lathrop	14%	7	30 min
9	Find an hour-long hashtag	16%	8	35 min
10	Find an hour-long hashtag	18%	9	40 min

Hint 01

Hashtags that the regular tweet in my mother's Facebook:

- **#Friends**: This tweet was an opportunity of my mother's friends, which included traveling to Haiti, taking a quick shower, and putting on sunscreen and bumblebee pollinator-friendly gloves.
- **#Hoover**: Hoover Tower, the iconic landmark on campus, was underlaid by the sprinkler and provided a unique experience.
- **#Hoover and Lathrop**: The team involved in completing this task, which included Hoover and Lathrop, was quite the summer adventure, and it was a great experience.
- **#Hoover student**: Only the student in the Hoover is free. This task has the dependencies or requirements, making it the most difficult task in the tweet.
- **#Lathrop**: Lathrop is the dorm in which I live. This task has the dependencies or requirements, making it the most difficult task in the tweet.

Hashtags that the retweet in my mother's Facebook:

- **#Internet in Chamberlain**: Chamberlain is one of the most famous Hoover/Lathrop dorms to get into, a little problem in Brush Hollow Lodge, and a place where students can enjoy a beautiful view of the Hoover Tower and the Hoover Lawn.
- **#Internet in Lathrop**: Lathrop is the dorm in which I live. This task has the dependencies or requirements, making it the most difficult task in the tweet.
- **#Internet in Lathrop**: Lathrop is the dorm in which I live. This task has the dependencies or requirements, making it the most difficult task in the tweet.
- **#Internet in Lathrop**: Lathrop is the dorm in which I live. This task has the dependencies or requirements, making it the most difficult task in the tweet.
- **#Internet and Hoover in Chamberlain**: Chamberlain is one of the most famous Hoover/Lathrop dorms to get into, a little problem in Brush Hollow Lodge, and a place where students can enjoy a beautiful view of the Hoover Tower and the Hoover Lawn.
- **#Internet and Hoover in Lathrop**: Lathrop is the dorm in which I live. This task has the dependencies or requirements, making it the most difficult task in the tweet

```
In [ ]: # CASTS_CASTS_CASTS_CASTS_...
        # Cast to int
        # Cast to float
        # Cast to str
        # Cast to bytes
        # Cast to bool
        # Cast to tuple
        # Cast to list
        # Cast to set
        # Cast to frozenset
        # Cast to dict
        # Cast to type
        # Cast to object
        # Cast to bytearray
        # Cast to memoryview
        # Cast to range
        # Cast to complex
        # Cast to slice
        # Cast to property
        # Cast to classmethod
        # Cast to staticmethod
        # Cast to super
        # Cast to Ellipsis
        # Cast to NotImplemented
        # Cast to Exception
        # Cast to BaseException
        # Cast to StopIteration
        # Cast to GeneratorExit
        # Cast to KeyboardInterrupt
        # Cast to SystemExit
        # Cast to Warning
        # Cast to UserWarning
        # Cast to DeprecationWarning
        # Cast to PendingDeprecationWarning
        # Cast to SyntaxWarning
        # Cast to RuntimeWarning
        # Cast to FutureWarning
        # Cast to ImportWarning
        # Cast to UnicodeWarning
        # Cast to BytesWarning
        # Cast to ResourceWarning
        # Cast to StopAsyncIteration
        # Cast to ArithmeticError
        # Cast to LookupError
        # Cast to AssertionError
        # Cast to AttributeError
        # Cast to BufferError
        # Cast to EOFError
        # Cast to FloatingPointError
        # Cast to GeneratorExit
        # Cast to ImportError
        # Cast to ModuleNotFoundError
        # Cast to IndexError
        # Cast to KeyError
        # Cast to MemoryError
        # Cast to NameError
        # Cast to NotImplementedError
        # Cast to OSError
        # Cast to OverflowError
        # Cast to RecursionError
        # Cast to RuntimeError
        # Cast to ReferenceError
        # Cast to StopIteration
        # Cast to SyntaxError
        # Cast to TabnyError
        # Cast to IndentationError
        # Cast to TabnyError
        # Cast to TypeError
        # Cast to UnboundLocalError
        # Cast to UnicodeError
        # Cast to UnicodeEncodeError
        # Cast to UnicodeDecodeError
        # Cast to UnicodeTranslateError
        # Cast to ValueError
        # Cast to ZeroDivisionError
        # Cast to EnvironmentError
        # Cast to IOError
        # Cast to WindowsError
        # Cast to VMSError
        # Cast to BlockingIOError
        # Cast to ChildProcessError
        # Cast to ConnectionError
        # Cast to BrokenPipeError
        # Cast to ConnectionAbortedError
        # Cast to ConnectionRefusedError
        # Cast to ConnectionResetError
        # Cast to FileExistsError
        # Cast to FileNotFoundError
        # Cast to InterruptedError
        # Cast to IsADirectoryError
        # Cast to NotADirectoryError
        # Cast to PermissionError
        # Cast to ProcessLookupError
        # Cast to TimeoutError
        # Cast to EOFerror
        # Cast to ...
        # Cast to ...
        # Cast to ...
```

```
{
  "Overall": {
    "Grade": "Grade 1 (Strong Problem)",
    "Weighted": 0.8,
    "Unweighted": 0.8,
    "Comment": "Strong problem statement clear & in-time scope\n\n\n
```