Exam: *Python for text analysis*

Teachers:
Emiel van Miltenburg Marten Postma Chantal van Son

December 22, 2016

This exam marks the end of the first part of *Python for text analysis*. It consists of five parts:

1. Booleans (15 points)
2. General knowledge about Python (24 points)
3. Spot the error (16 points)
4. Tracing (20 points)
5. Writing code (15 points)

The total amount of points you can earn is 90. You will get 10 points for free. Your grade will be computed by dividing the total number of points by 10. For example: 99 points corresponds to a 9.9 out of 10.

1 Booleans

Please indicate for each of the cases whether they print `True` or `False`. Each question is worth 1 point.

1. ```
   a_string = 'airplane'
   another_string = 'airplanes'
   print(len(another_string) >= len(a_string))
   ```

2. ```
   print('100' == 100)
   ```

3. ```
   print( (100 - 7) in [5, 1, 92, 7, 17, 13])
   ```

4. ```
   a_number = 2
   another_number = 17
   print( (a_number + another_number) > 19)
   ```

5. ```
   print(not False == True)
   ```

6. ```
   print('a'.isupper() == (not False))
   ```

7. ```
   print(sorted([2, 4, 7, 1]) == [1, 2, 4, 7])
   ```

8. ```
   a_string = 'airplane'
   another_string = 'airplanes'
   print(a_string[0] != another_string[5])
   ```

9.
```
a_string = 'airplane'
another_string = 'airplanes'
print(a_string[3:] == another_string[3:8])
```

10.
```
a_list = ['hello', ['what', 'is'], 'happening', 'here']
appended = a_list.append('?')
print(appended == '?')
```

11.
```
a_dict = {'a': [1, 2], 'b': 'hello'}
print(len(a_dict['a']) <= 2)
```

12.
```
a_set = set([1, 4, 6, 8, 2, 3, 1, 1, 3, 5])
a_set.update([1, 3, 4, 1, 1])
print(len(a_set) == 7)
```

13.
```
letters = ['a','b','c','d']
numbers = set([1, 2, 3, 4, 5])
print(all([len(letters) == 4,
           len(numbers) != 5]))
```

14.
```
a_string = 'airplane'
another_string = 'airplanes'
print(any([len(a_string) == 7,
           a_string == another_string,
           (2 + 17 + 2) > 21]))
```

15.
```
print(all(['airplane'.startswith('air'),
           'airplane'.replace('a', 'b') == 'birplbne',
           'airplane' not in ['the airplane', 'is in the air']
          ]))
```

2 General knowledge about Python

Please answer the questions below. Each question is worth 2 points (total 24).

16. What is the output of `len([5, 2, [1, 2], 3])`?

17. Explain what the three values mean in Python's slicing notation. For example, what does the following print? `"Bananas are yummy."[1:-1:2]`

18. What is the difference between arguments (args) and keyword arguments (kwargs)?

19. Explain what the `continue, break` and `pass` statements do.

20. Which single change in the next line would print each name on a separate line?
```
print(" ".join(["Alan", "Maria", "Paul"]))
```

21. Shorten the following statement: `a < b and c > b`

22. What are the two types of objects that you can define using curly braces `{}`?

23. What does `max()` do? For example: `max([1,2,3])`

24. How can you turn a string representation of a number (e.g. `'1'`) into an integer?

25. How can you sort a list **from high to low** using `sorted()`?

26. What built-in function can you use to loop over all the whole numbers up to, for example, 100?

27. What built-in function can you use to learn more about how to use a particular function?

3 Spot the error

Below are ten pieces of code with a mistake in them. Please indicate the errors, and explain why they cause the code to break (you don't need to know the exact error message). Each question is worth 2 points (one for pointing where exactly the mistake is, one for the explanation), with a total of 16 points. For your convenience, we also printed line numbers next to each line of code. But we do expect you to **be more explicit than just specifying the line number.** (A one-sentence explanation is fine, though.)

Example question: where is the mistake here, and why doesn't it work?

```
1 my_list = [1,2,3]
2 2nd_item = my_list[1]
```

Answer: The mistake is in the variable name, Python doesn't allow them to start with numbers.

28. Where is the mistake, and why doesn't it work?

```
1 shopping_list = {"apples": 5, "bananas": 3, "pineapples": 2}
2 for product in shopping_list:
3     number = shopping_list[product]
4     print("We need " + number + " " + product)
```

29. Where is the mistake, and why doesn't it work?

```
1 fruit = ["apple", "banana", "strawberry", "pineapple"]
2 fruit.add("mango")
```

30. Where is the mistake, and why doesn't it work?

```
1 fruit = ["apple", "banana", "strawberry", "pineapple"]
2 fruit_string = ",".join(fruit)
3 fruit = fruit_string.split()
4 print(fruit[1])
```

31. Where is the mistake, and why doesn't it work?

```
1 fruit1 = fruit2 = ["apple", "banana", "strawberry", "pineapple"]
2 fruit2.remove("apple")
3 print(fruit1[3])
```

Continued on next page...

32. Where is the mistake, and why doesn't it work?

```
def count_items_in_list(a_list):
    count = 0
    for item in a_list:
        count += 1
    return count

fruit = ["apple", "banana", "strawberry", "pineapple"]
count_items_in_list(fruit)
print(count)
```

33. Where is the mistake, and why doesn't it work?

```
number = 5
if number < 10:
print(number, "is lower than 10")
```

34. Where is the mistake, and why doesn't it work?

```
numbers = {1: "one", 2: "two", 3: "three"}
print(numbers["1"])
```

35. Where is the mistake, and why doesn't it work?

```
number = 5413
if 1 in number:
    print("1 in", number)
```

4 Tracing

Read the code in appendix A and answer the questions below. Each question is worth 2 points (total 20).

Context

Before you start reading the code, here is some context to understand what the code does: I wrote this code to get all the sentences in which the author indicated some form of contrast, and then write those sentences to a file. An important consideration for me is that **I don't want to store the same sentence (that is: with the same sentence ID) multiple times.** Because I don't speak Turkish, I just went to Google Translate to find translations for *but* and *however*. It turns out that there are many ways to translate these words! They're listed in two sets: `single_word` has all the single-word translations, and `multiword` has all the multiword translations.

General information

On line 47, `get_and_write_sentences(...)` calls the function with three arguments: `single_word`, `multiword`, and a filename for the results file. The data comes from a file called 'tasviret8k_captions.json'. The function `get_and_write_sentences(...)` only loops over the image data in this file. Here is an example image (this is the 327$^{\text{th}}$ element of `tasviret['images']`):

```
1  {'filename': '2869253972_aa72df6bf3.jpg',
2   'imgid': 327,
3   'sentences': [{'imgid': 327,
4                  'raw': 'Bileğini başka bir adama gösteren kişi.',
5                  'sentid': 657,
6                  'tokens': ['Bileğini', 'başka', 'bir', 'adama', 'gösteren', 'kişi']},
7                 {'imgid': 327,
8                  'raw': 'İki adamdan birisi diğerine elini vermiş bir şeyleri '
9                         'inceliyorlar.',
10                 'sentid': 658,
11                 'tokens': ['İki', 'adamdan', 'birisi', 'diğerine', 'elini', 'vermiş',
12                            'bir', 'şeyleri', 'inceliyorlar']}],
13  'sentids': [657, 658],
14  'split': 'train'}
```

It's just a dictionary specifying different attributes of an image. Relevant for us is that:

- Each image has an image ID (`imgid`) that we want to store.
- Each image has multiple sentences describing it.
- Each sentence has a sentence ID (`sentid`) that we want to store.
- Each sentence is represented as a raw string (`raw`), and as a list of strings (`tokens`).
- Don't worry about the string on line 9: it's a feature of Python to be able to break up long strings like this to keep the text readable.

Questions about the code

Read the code in appendix A and answer the questions below. Each question is worth 2 points (total 20 points).

36. What is the **type** and **value** of `i` after it's defined in the first iteration of the for-loop on line 23?

37. What is the **type** and **value** of `overlap` when the function processes the sentence with ID 657?

38. What is the value of `relevant_sentences` after processing the example image above? (Assume this is the only image in the dataset.)

39. Why is `continue` used instead of `break` on line 34?

40. What might go wrong with the results if we replaced `break` with `continue` on line 39?

41. What will the first line of `contrast.tsv` look like after running the code?

42. Would the code still work if we removed '`filename=`' from line 47? Why (not)?

43. Would the code still work if we defined `single_word` as a list instead of a set? Why (not)?

44. Would the code still work if we defined `multiword` as a list instead of a set? Why (not)?

45. If we were to turn `row` (line 25) into a set instead, which **two things** might go wrong when we write the rows out to the TSV file? (Assume that the rest of the code would be fixed, replacing all `append` methods with `add`.)

5 Writing code

Below, you will find code descriptions. Please write the code according to these descriptions. Each piece of code is worth 5 points (total 15).

General information

Imagine that there are two files:

- somewords1.txt
- somewords2.txt

Each line in the file *somewords1.txt* contains one word (\n indicating the newline character), e.g.:

```
1 table\n
2 chair\n
3 tea\n
4 coffee\n
5 money\n
6 ....
```

Each line in the *somewords2.txt* can contain multiple words. The words on a line are separated by a tab \t in Python). \n indicates the newline character. The number of words on each line ranges from 1 to 4.

```
1 bed\tblanket\toffice\n
2 plate\tfork\n
3 jeans\n
4 ....
```

Code descriptions

46. Please write the code to:

    (a) define a set called **unique_somewords2**

    (b) loop through the lines of file *somewords2.txt*

    (c) remove the newline character from each line

    (d) add each individual word in each line to **unique_somewords2**

47. Please write the code to:

    (a) define a set called **filtered_uniquewords2**

    (b) loop through the lines of file *somewords2.txt*

    (c) remove the newline character from each line

    (d) checks for each word in the line if it:

        i. starts with the letter *a*

        ii. is longer than six letters

        iii. is shorter than or equal to two letters

    (e) If any of the checks in (d) are True for a word, we add the word to **filtered_uniquewords2**. Else, we do not add the word to **filtered_uniquewords2**.

48. Please write the code to:

    (a) define three sets: **set_a**, **set_b**, and **set_rest**

    (b) loop through the lines of file *somewords1.txt*

    (c) remove the newline character from each line

    (d) perform the following actions:

        i. condition A: if the word starts with *b* or *e* $\rightarrow$ add the word to **set_a**

        ii. condition B: only if condition A is not met, but if the word consists of 10 characters $\rightarrow$ add the word to **set_b**

        iii. condition C: in all other conditions (hence not condition A and not condition B) $\rightarrow$ add the word to **set_rest**

Page intentionally left blank. Appendix A on the next page.

A Contrast.py

```
import csv
import json

# Words indicating contrast (Google Translate results for "but" and "however")
single_word = {"ancak", "ama", "oysa", "halbuki", "fakat", "başka",
               "ki", "hariç", "sadece", "yalnızca", "yani"}

# Multiword expressions indicating contrast (more Google Translate results)
multiword = {"her ne şekilde", "her nasılsa", "her halükârda",
             "nasıl olursa olsun", "nasıl oldu da", "hiç olmazsa"}

def get_and_write_sentences(single_word, multiword, filename):
    """
    Function that loads the Tasviret8k data, and searches for sentences containing
    either single words or multiword expressions. The sentences are then saved to
    FILENAME.
    """
    with open('./tasviret8k_captions.json') as f:
        tasviret = json.load(f)

    relevant_sentences = []
    for image in tasviret['images']:
        for i, sentence in enumerate(image['sentences']):
            raw_sentence = sentence['raw']
            row = [str(image['imgid']),
                   str(i),
                   str(sentence['sentid']),
                   raw_sentence]
            tokens = set(sentence['tokens'])
            overlap = single_word & tokens
            if overlap:
                row.append(', '.join(overlap))
                relevant_sentences.append(row)
                continue
            for mwe in multiword:
                if mwe in raw_sentence.lower():
                    row.append(mwe)
                    relevant_sentences.append(row)
                    break

    header = "imgid sentence_number sentid raw_sentence match".split()
    with open(filename, 'w') as f:
        writer = csv.writer(f, delimiter='\t')
        writer.writerow(header)
        writer.writerows(relevant_sentences)

get_and_write_sentences(single_word, multiword, filename='contrast.tsv')
```