Python for Text Analysis
Programming for Humanities and Social Sciences
Exam 2017-2018

Teachers:
Filip Ilievski Marten Postma Chantal van Son

December 18, 2017

This exam marks the end of *Programming for Humanities and Social Sciences* (6 ECTS) and the first part of *Python for Text Analysis* (9 ECTS). 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. ```
   print("10" == str(10))
   ```

2. ```
   print((14/7) in (1, 3, 5, 7, 9, 12, 14))
   ```

3. ```
   str1 = "earth"
   str2 = "jupyter"
   print(str1[0] == str2[-1])
   ```

4. ```
   a_list = ["a", 1, "8"]
   b_list = ["8", 1, "8", "a"]
   print(a_list != b_list)
   ```

5. ```
   a_list=["a", 1, "8"]
   b_list=["8", 1, "8", "a"]
   print(set(a_list) != set(b_list))
   ```

6. ```
   one_string = "piece"
   another_string = "pie"
   print(one_string[0:2] == another_string)
   ```

7.
```
str1 = "maps.google.com"
str2 = "https://news.ycombinator.com"
print(len(str1.split(".")) == len(str2.split(".")))
```

8.
```
dict1 = {"Michelle": 9, "Jerry": 8, "Mark": 10, "Annette": 5}
dict1["Mark"] = 8
dict1["John"] = 9
print(len(dict1) == 4)
```

9.
```
a_list = ["Python", "master", "here"]
appended = a_list.append(":)")
print(appended == ":)")
```

10.
```
set1 = {"a", "b", "c"}
set1.update({"b", "c", "d"})
print(len(set1) < 5)
```

11.
```
dict1 = {"name": "John", "surname": "Doe", "friends": [{"name": "John",
"surname": "Smith"}]}
print(len(dict1["friends"][0]) == 1)
```

12.
```
a = 4
b = 9
c = 13
d = 19
print((a+b>=c) and (a+c>=d))
```

13.
```
a_list = list()
a_set = set()
print(all([(10+5-10)>10, "me" not in "team", len(a_list)==len(a_set)]))
```

14.
```
a = "Letter"
b= "nUMBER"
print(any([a.lower()==b.lower(), len(a)>=len(b)]))
```

15.
```
a_str = "John and Marry jumped on a jetplane"
new_str = a_str.replace("j", "g")
print(new_str == "Gohn and Marry gumped on a getplane")
```

2 General knowledge about Python

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

16. Can you use a list as a dictionary key? And as a dictionary value?

17. One way to combine strings is to concatenate them using the + sign:
`print("Hello " + user + "!  You were last online in the month of " + month + ".")`
Can you show an alternative way to print the same output without string concatenation?

18. What is the difference between parameters and arguments of a function?

19. What is the value and type of 'y' after the following code:

```
x = "Hello! Good afternoon! Bye"
y = x.split('!')
```

20. Give a line of code to unpack the tuple `band_name = ("Red Hot", "Chili Peppers")` into variables. The number of variables should be equal to `len(band_name)`.

21. Give two mutable and two immutable variable types.

22. For each letter in the alphabet, you want to store a unique set of words starting with this letter (e.g. "airplane", "air", and "apple" for "a", "bubble" and "blossom" for "b", etc.). Which data container would you use to store that data (both the letters and the corresponding words together)?

23. How can we access the last element of a list?

24. What is the purpose of the `return` statement in a function?

25. Which module and which function can we use to list all files from a directory?

26. Consider a list, for example: `a = ["You", "are", "what", "you", "eat"]`.
Please explain the difference between `break` and `continue`. Illustrate the difference by giving an example in which you iterate over list `a`.

27. Please list and briefly explain three differences in the properties of lists and sets.

3 Spot the error

Below are eight 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 `2nd_item` (line 2). Python doesn't allow variable names to start with numbers.

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

```
1 animals = {"zebra": 5, "elephant": 2, "lion": 1,"chicken": 10}
2 for animal in animals.values():
3     print(animals[animal])
```

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

```
1 animals = ["kangar00", "kitten", "peac0ck", "k1w1", "penguin"]
2 for animal in animals:
3     if animal.isalpha():
4         alpha_animals += 1
```

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

```
animals = {"lizzard", "spider", "mouse"}
print(animals[2])
```

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

```
animals = ("tiger", "leopard", "cheetah")
first, second, third, fourth = animals
```

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

```
animals = ["shark", "wolf", "frog"]
if "shark" in animals:
print("help")
```

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

```
animals = ["giraffe", "duck", "bee"]
animals.append(["horse", "panda"])
print(animals[5])
```

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

```
animals = ("tiger", "leopard", "chimpansee")
if animals.startswith("tiger"):
    print("roar")
```

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

```
def count_animals(list_animals):
    for animal in list_animals:
        total += 1
    return total

animals = ["walrus", "seal", "polar bear"]
total = count_animals(animals)
print(total)
```

4 Tracing

Context

FrameNet is a lexical database describing *semantic frames*, which are representations of events or situations and the participants in it. For example, cooking typically involves a person doing the cooking (COOK), the food that is to be cooked (FOOD), something to hold the food while cooking (CONTAINER) and a source of heat (HEATING_INSTRUMENT). In FrameNet, this is represented as a frame called APPLY_HEAT. The COOK, FOOD, HEATING_INSTRUMENT and CONTAINER are called *frame elements (FEs)*. Words that evoke this frame, such as *fry*, *bake*, *boil*, and *broil*, are called *lexical units (LUs)* of the APPLY_HEAT frame. FrameNet also contains relations between frames. For example, APPLY_HEAT has relations with the ABSORB_HEAT, COOKING_CREATION and INTENTIONALLY_AFFECT frames. In FrameNet, frame descriptions are stored in XML format.

Appendices: Code & File

Have a look at the following appendices:

- **APPENDIX A:** `framenet.py` contains four functions to extract some information about a frame:
  - `get_basic_info()`
  - `get_frame_elements()`
  - `get_frame_relations()`
  - `get_lexical_units()`

  These functions are all called on the root element of the XML representation of WAKING_UP.
- **APPENDIX B:** `Waking_up.xml` is a (simplified) XML representation of the frame WAKING_UP.

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** of `frame_def` after it's defined on line 5? And after it's re-defined on line 6?
37. What are the **values** of `frame_def` and `frame_ex` after they're defined on line 7?
38. What is the **type** and **value** of `fe_name` after it's defined in the first iteration of the for-loop on line 12?
39. Would the code still work if on line 11 we defined `fes` as a set instead of a list and used the `add` method instead of `append`? Why (not)?
40. What kind of error is prevented by line 28?
41. How many **keys** will `relations` contain once the for-loop on line 24 is finished?
42. What would be the **length** of `relations["Is Preceded By"]` once the for-loop on line 24 is finished?
43. What is the **value** of `lex_units` after the first iteration of the for-loop on line 35?
44. What should be changed to prevent an error in the second iteration of the for-loop on line 35?
45. Would the code still work if on line 34 we defined `lex_units` as a list instead of a set and used the `append` method instead of `add`? Why (not)?

5 Writing code

Below, you will find code descriptions A and B. Please write the code according to these descriptions. This exercise brings 15 points in total.

General information

Let's look at the 40 best duet songs of all time, according to `https://www.billboard.com/articles/list/473075/the-40-biggest-duets-of-all-time`. Imagine that information on each of these duets is stored in a file called `best_duets.tsv`, which is in TSV format and contains five columns:

1. unique numeric identifier of a song
2. song title
3. song length in seconds
4. year of release
5. singer(s)

The contents of this file look as follows:

```
40\tThe Closer I Get To You\t283\t1978\tRoberta Flack & Donny Hathaway\n
39\tI just can't Stop Loving You\t231\t1987\tMichael Jackson & Siedah Garrett\n
...
1\tEndless Love\t270\t1981\tDiana Ross & Lionel Richie\n
```

So, number #1 on the billboard is the song "Endless Love", which is 270 seconds long, is released in 1981, and is sang by Diana Ross and Lionel Richie. In this part of the exam, we will work with the data stored in this file.

Code A (4 points)

Please write a function, according to the following specifications:

46. The function should be called `dates_between`.
47. It should have one positional parameter `year`.
48. The function should have two keyword parameters: `start` with a default value 1980 (integer) and `end` with a default value 1990 (integer).
49. The function should return `True` if the year has a value between `start` and `end` (including the start and end year). Otherwise it should return `False`.

Code B (11 points)

Please write code to:

50. Load the module/package that allows us to work with JSON files.
51. Create an empty dictionary `my_duets`.
52. Open the file `best_duets.tsv` for reading (assume that the file is in the current working directory).
53. Loop through the lines of this TSV file.
54. Remove the newline character from the end of each line.
55. Split each line into a list with 5 elements (the five elements should be the ones explained in the general information).

56. Call the boolean function `dates_between` (which we defined in part A), to check if the year of release is later than 2000 and earlier than 2017. If this function returns `True`, then print `"Song X is from my age"`, where `X` should be replaced with the song title.

57. Split the last column (the one describing singers) of each row/line into a list of all singers. For example, for the #1 song we will have `["Diana Ross", "Lionel Richie"]`

58. For each line, store the information on a song identifier and its singers (as a list) in the dictionary you created in step 2. This should result in the following information:

```
my_duets= {"1": ["Diana Ross", "Lionel Richie"], ...,
           "40": ["Roberta Flack", "Donny Hathaway"]}
```

59. Modify the authors of the song at place #40 (identifier "40"), by appending your name and surname to the list of authors, like this:

```
my_duets= {"1": ["Diana Ross", "Lionel Richie"], ...,
           "40": ["Roberta Flack", "Donny Hathaway", "Name Surname"]}
```

60. Open a JSON file `best_duets.json` for writing. Store the dictionary `my_duets` to this file.

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

Appendix A `framenet.py`

```
from lxml import etree

def get_basic_info(root):
    frame_name = root.get("name")
    frame_def = root.find("definition").text
    frame_def = frame_def.split(" EXAMPLE: ")
    frame_def, frame_ex = frame_def
    return frame_name, frame_def, frame_ex

def get_frame_elements(root):
    fes = list()
    for fe in root.findall("FE"):
        fe_type = fe.get("coreType")
        fe_name = fe.get("name")
        fe_def = fe.find("definition").text
        dict_fe = {"type": fe_type,
                   "name": fe_name,
                   "definition:": fe_def}
        fes.append(dict_fe)
    return fes

def get_frame_relations(root):
    relations = dict()
    for relation in root.findall("frameRelation"):
        for frame in relation.findall("relatedFrame"):
            rel_type = relation.get("type")
            rel_name = frame.text
            if not rel_type in relations:
                relations[rel_type] = list()
            relations[rel_type].append(rel_name)
    return relations

def get_lexical_units(root):
    lex_units = set()
    for lex_unit in root.findall("lexUnit"):
        name = lex_unit.get("name")
        lex_units = lex_units.add(name)
    return lex_units

tree = etree.parse("./Waking_up.xml")
root = tree.getroot()
frame_name, frame_def, frame_ex = get_basic_info(root)
fes = get_frame_elements(root)
relations = get_frame_relations(root)
lex_units = get_lexical_units(root)
```

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

Appendix B `Waking_up.xml`

```
<frame name="Waking_up" id="295">
    <definition>A Sleeper transitions from a state of consciousness where they
        are largely unaware of their environment to a wakeful state. EXAMPLE:
        William awoke from a dream.</definition>
    <FE coreType="Peripheral" name="Place" id="2527">
        <definition>This FE identifies the Place where the event occurs.</
            definition>
        <semType name="Locative_relation" id="182"/>
    </FE>
    <FE coreType="Core" name="Sleeper" id="2528">
        <definition>This FE identifies the sleeping entity.</definition>
    </FE>
    <FE coreType="Peripheral" name="Time" id="2529">
        <definition>This FE identifies the Time when the event occurs.</
            definition>
        <semType name="Time" id="141"/>
    </FE>
    <FE coreType="Core" name="Sleep_state" id="3471">
        <definition>This describes the state out of which the Sleeper awakens.</
            definition>
    </FE>
    <FE coreType="Peripheral" name="Manner" id="3472">
        <definition>The Manner in which the Sleeper awakens.</definition>
        <semType name="Manner" id="173"/>
    </FE>
    <FE coreType="Extra-Thematic" name="Depictive" id="3473">
        <definition>The state of the Sleeper upon waking up.</definition>
    </FE>
    <FE coreType="Extra-Thematic" name="New_situation" id="3474">
        <definition>The New_situation is the situation the Sleeper finds him or
            herself in upon awaking.</definition>
    </FE>
    <FE coreType="Extra-Thematic" name="Circumstances" id="10457">
        <definition>Circumstances describe the state of the world (at a
            particular time and place) which is specifically independent of the
            event itself and any of its participants.</definition>
    </FE>
    <FE coreType="Extra-Thematic" name="Particular_iteration" id="10458">
        <definition>This FE indicates an instance of the Waking_up event within
            an iterated series of similar events or states.</definition>
    </FE>
    <FE coreType="Extra-Thematic" name="Explanation" id="10459">
        <definition>The reason for which the Sleeper awakes from the Sleep_state
            .</definition>
    </FE>
    <FE coreType="Extra-Thematic" name="Frequency" id="11980">
        <definition>This frame element is defined as the number of times an
            event occurs per some unit of time.</definition>
    </FE>
    <frameRelation type="Inherits from">
        <relatedFrame id="187">Event</relatedFrame>
    </frameRelation>
    <frameRelation type="Is Inherited by"/>
    <frameRelation type="Perspective on"/>
    <frameRelation type="Is Perspectivized in"/>
    <frameRelation type="Uses"/>
    <frameRelation type="Is Used by"/>
    <frameRelation type="Subframe of">
        <relatedFrame id="294">Sleep_wake_cycle</relatedFrame>
    </frameRelation>
    <frameRelation type="Has Subframe(s)"/>
```

```
    <frameRelation type="Precedes">
        <relatedFrame id="1681">Being_awake</relatedFrame>
    </frameRelation>
    <frameRelation type="Is Preceded by">
        <relatedFrame id="264">Sleep</relatedFrame>
    </frameRelation>
    <frameRelation type="Is Inchoative of"/>
    <frameRelation type="Is Causative of"/>
    <frameRelation type="See also"/>
    <lexUnit POS="V" name="come to.v" id="5375" lemmaid="45583">
        <definition>FN: wake up</definition>
        <lexeme order="1" headword="true" breakBefore="false" POS="V" name="come
            "/>
        <lexeme order="2" headword="false" breakBefore="false" POS="ADV" name="
            to"/>
    </lexUnit>
    <lexUnit POS="V" name="awake.v" id="6926" lemmaid="2506">
        <definition>COD: stop sleeping. </definition>
        <lexeme order="1" headword="false" breakBefore="false" POS="V" name="
            awake"/>
    </lexUnit>
    <lexUnit POS="V" name="wake.v" id="6927" lemmaid="43355">
        <definition>COD: emerge from a state of sleep</definition>
        <lexeme order="1" headword="false" breakBefore="false" POS="V" name="
            wake"/>
    </lexUnit>
    <lexUnit POS="V" name="wake up.v" id="6928" lemmaid="45568">
        <definition>COD: emerge from a state of sleep</definition>
        <lexeme order="1" headword="true" breakBefore="false" POS="V" name="wake
            "/>
        <lexeme order="2" headword="false" breakBefore="true" POS="ADV" name="up
            "/>
    </lexUnit>
    <lexUnit POS="V" name="get up.v" id="9130" lemmaid="45569">
        <definition>FN: to wake up</definition>
        <lexeme order="1" headword="true" breakBefore="false" POS="V" name="get"
            />
        <lexeme order="2" headword="false" breakBefore="true" POS="ADV" name="up
            "/>
    </lexUnit>
    <lexUnit POS="V" name="revive.v" id="12099" lemmaid="33121">
        <definition>FN: to return to consciousness</definition>
        <lexeme order="1" headword="false" breakBefore="false" POS="V" name="
            revive"/>
    </lexUnit>
    <lexUnit POS="V" name="come back around.v" id="15503" lemmaid="51026">
        <definition>FN: return to consciousness with difficulty.</definition>
        <lexeme order="1" headword="true" breakBefore="false" POS="V" name="come
            "/>
        <lexeme order="2" headword="false" breakBefore="true" POS="ADV" name="
            back"/>
        <lexeme order="3" headword="false" breakBefore="false" POS="ADV" name="
            around"/>
    </lexUnit>
</frame>
```