20-00-0546-iv Foundations of Language Technology

Handout 3
Writing structured programs

19. November 2020

3.1 Warm up

Question 3.1 *Datatype: Which expression results in a list?*

(1) `a[1:]`
(2) `a | b`
(3) `a & b`
(4) `(a,b)`

Question 3.2 *What are the differences between lists and dictionaries? Decide, whether the following statements are true or false.*

(1) Dictionaries have a fixed number of items, while lists may have variable length.
(2) Dictionaries and lists can be indexed using strings.
(3) Dictionaries can be nested in contrast to lists, i.e. you can have a "dictionary of dictionaries" but no "list of lists".
(4) Dictionaries can contain only strings, while lists can contain any object.
(5) Dictionaries are mutable, while lists are not.

Question 3.3 *Default Dictionaries: What is the output of the following program? Explain what happens.*

```
1  import nltk
2
3  words = ['I', 'like', 'football', 'you', 'hate', 'football', 'you', 'like', '
       climbing']
4
5  default_dict = nltk.defaultdict(int)
6
7  for word in words:
8    word = word.lower()
9    default_dict[word] +=1
10
11 print(default_dict['like'])
12 print(default_dict['tennis'])
13
14
15 normal_dict = {}
16
17 # this will throw a KeyError!
18 for word in words:
19   word = word.lower()
20   normal_dict[word] +=1
21
22 print(normal_dict['like'])
23 print(normal_dict['tennis'])
```

3.2 Writing structured programs

Task 3.1 *Within any programming language it is essential to understand how variables are handled. Roughly said, either a variable refers to its actual value or it refers to a place, an address in the memory. Hence, when you assign one variable to another and the variable holds an address, any change will effect all variables holding this address. The following tasks should show you the effects this might have.*

(a) Create a list of words and store it in a variable `sent1`*. Now assign* `sent2=sent1`*. Modify one of the items in* `sent1` *and verify that* `sent2` *has changed.*[1]

[1] See NLTK-book page chapter 4, exercise 11, page 174

(b) *Do the same with the following datatypes: integer, string, set and tuple. E.g. assign an integer to one variable etc.*

(c) *Now try the same exercise as (a), but instead assign* `sent2=sent1[:]`. *Modify* `sent1` *again and see what happens to* `sent2`. *Explain.*

(d) *Now define* `text1` *to be a list of lists of strings (e.g. to represent a text consisting of multiple sentences). Now assign* `text2=text1[:]`, *assign a new value to one of the words, e.g.* `text1[1]='Monty'`. *Check whether this also changed* `text2`. *Explain.*

Task 3.2 *Dictionaries and default dictionaries are important data types for NLP-tasks. Run, explore, and explain the following lines of code.*

```
import nltk, collections

pos = {'colorless': 'ADJ', 'ideas': 'N', 'sleep': 'V', 'furiously': 'ADV'}
pos2 = dict((value, key) for (key, value) in pos.items())
print(pos2['ADV'])
pos.update({'cats': 'N', 'scratch': 'V', 'peacefully': 'ADV', 'old': 'ADJ'})
print(pos)
pos3 = dict((value, key) for (key, value) in pos.items())
print(pos3['ADV'])
pos4 = collections.defaultdict(list)
for key, value in pos.items():
  pos4[value].append(key)
print(pos4['ADV'])
pos5 = collections.defaultdict(list)
pos5 = nltk.Index((value, key) for (key, value) in pos.items())
print(pos5['ADV'])
pos6 = nltk.defaultdict(lambda: 'N')
pos6['colorless'] = 'ADJ'
print(pos6['blog'])
print(pos6.items())
last_letters = nltk.defaultdict(list)
for word in nltk.corpus.words.words('en'):
  last_letters[word[-2:]].append(word)
print(last_letters['ly'][:10])
```

Task 3.3 *Good programming style includes a reasonable choice of the used data types.*

(a) *Which datatypes do you know in Python by now?*

(b) *Give three examples for each of the types from (a):*

(c) *Compare the pairs <list, set>, <list, tuple>, <set, tuple>, <dictionary, defaultdictionary>, <set, dictionary>. What are the differences, what are the similarities?*

Task 3.4 *During the development process and while finding errors, it is helpful to let the program print out more information. It is tedious to write and remove those print-statements. Python offers a module named logging, which may be used in its simplest form as follows:*

```
import logging
logging.basicConfig(level = logging.DEBUG)
logging.error("error: ...")
logging.info("info: ...")
logging.debug("debug: ...")
```

The log message is only emitted if the logger is configured to emit messages of ***that level or higher****. For example, if a message is CRITICAL, and the logger is set to ERROR, the message is emitted. If a message is a WARNING, and the logger is set to produce only ERRORs, the message is not emitted. The ordering (from highest to lowest) of levels is CRITICAL, ERROR, WARNING, INFO, DEBUG, NOTSET.*

Change your library functions to use this code, i.e. remove all print statements (except those of the required output) and use the logging methods instead. For those familiar with log4net or log4j, you may dive into the details of the logging module, it offers a similar flexibility. Consult the following documentation:

- *Python documentation, logging*
- *Galileo Computing, Python, chapter 21.4*
- *O'Reilly, Python, standard logging*

3.3 Using structured programs to extract keywords

Task 3.5 *Write a function* `shorten(text,n)` *to process a text, omitting the n most frequently occurring words of the text.* [2]

(a) Apply the function `shorten` *to three texts of your choice using* $n = 10$*. How readable is the result?*
Hint: You may choose any text and get its tokens with `nltk.word_tokenize(text)`

(b) Experiment with the values of n to find a useful limit.

[2] See NLTK-book page chapter 4, exercise 17, page 175