Wk1_SLP2_Lists and dictionaries in Python v2

March 13, 2021

1 MA5851 Week 1 Self Learning Practical 2

[ ]:
```
student_name = "Nikki Fitzherbert"
student_id = "13848336"
```

The goal of this week's practical is to get you started using Python and Jupyter Notebooks two tools that you will use through the semester in your work.

Python is our language of choice in MA5851.

You are looking at a **Jupyter Notebook**, it is a document that mixes text, code and the output of the code. A lot of your work will be creating notebooks like this to present your analysis.

1.1 Python Basics

1.2 Lists and Dictionaries

First we look at some built in Python data structures: lists and dictionaries.

A list is a sequence of things, unlike strongly typed languages (Java, C#) a list can contain a mixture of different types - there is no type for a list of integers or a list of lists. Here are some lists:

[1]:
```
ages = [12, 99, 51, 3, 55]
names = ['steve', 'jim', 'mary', 'carrie', 'zin']
stuff = [12, 'eighteen', 6, ['another', 'list']]
```

1. write code to print the first and third elements of each list
2. write code to select and print everything except the first element of each list
3. write a for loop that prints each element of the 'names' list

[2]:
```
# 1. Printing first and third elements of each list
list_of_lists = [ages, names, stuff]

for list in list_of_lists:
    print(list[0], list[2])
```

```
12 51
steve mary
12 6
```

[3]:
```python
# 2. Printing everything except the first element of each list
for list in list_of_lists:
    print(list[1:])
```

```
[99, 51, 3, 55]
['jim', 'mary', 'carrie', 'zin']
['eighteen', 6, ['another', 'list']]
```

[4]:
```python
print([list[1:] for list in list_of_lists]) # Using a list comprehension
```

```
[[99, 51, 3, 55], ['jim', 'mary', 'carrie', 'zin'], ['eighteen', 6, ['another',
'list']]]
```

[5]:
```python
# 3. Using a for-loop to print each element of the 'names' list
for item in names:
    print(item)
```

```
steve
jim
mary
carrie
zin
```

A dictionary is an associative array - it associates a value (any Python data type) with a key. The key is usually a string but can be any immutable type (string, number, tuple). Here's some code that counts the occurence of words in a string. It stores the count for each word in a dictionary using the word as a key. If the word is already stored in the dictionary, it adds one to the count, if not, it initialises the count to one.

The second for loop iterates over the keys in the dictionary and prints one line per entry.

Modify this example to be a bit smarter: - make sure that punctuation characters are not included as parts of a word, be careful with hyphens - should they be included or not? - make the count use the lowercase version of a word, so that 'The' and 'the' are counted as the same word - **Challenge**: find the first and second most frequent words in the text - **Challenge**: take your code and write it as a function that takes a string and returns a list of words with their counts in order

[7]:
```python
description = """This unit introduces students to the fundamental techniques␣
 ↪and
tools of data science, such as the graphical display of data,
predictive models, evaluation methodologies, regression,
classification and clustering. The unit provides practical
experience applying these methods using industry-standard
software tools to real-world data sets. Students who have
completed this unit will be able to identify which data
science methods are most appropriate for a real-world data
set, apply these methods to the data set, and interpret the
results of the analysis they have performed. """
```

[ ]:
```
count = dict()
for word in description.split():
    if word in count:
        count[word] += 1
    else:
        count[word] = 1

for word in count:
    print(word, count[word])
```

[9]:
```
# modified code so punctuation characters not included (but hyphens are), and␣
 ↪lowercase used for the count
import re

count = dict()
for word in re.split(r'[\s+\.,]', description.lower()):
    if word in count:
        count[word] += 1
    else:
        count[word] = 1

for word in count:
    print(word, count[word])
```

```
this 2
unit 3
introduces 1
students 2
to 4
the 6
fundamental 1
techniques 1
and 3
 20
tools 2
of 3
data 6
science 2
such 1
as 1
graphical 1
display 1
predictive 1
models 1
evaluation 1
methodologies 1
regression 1
```

```
classification 1
clustering 1
provides 1
practical 1
experience 1
applying 1
these 2
methods 3
using 1
industry-standard 1
software 1
real-world 2
sets 1
who 1
have 2
completed 1
will 1
be 1
able 1
identify 1
which 1
are 1
most 1
appropriate 1
for 1
a 1
set 2
apply 1
interpret 1
results 1
analysis 1
they 1
performed 1
```

[10]:
```
# challenge: the two most-frequent words in the text
sorted_count = sorted(count.items(), key = lambda x: x[1], reverse = True)
print(sorted_count[1:3]) # first item in list is not a word so excluded
```

```
[('the', 6), ('data', 6)]
```

[11]:
```
# challenge: function that returns a sorted list of words with their counts␣
↪given a string
def sorted_word_count(string):
    count = dict()
    for word in re.split(r'[\s+\.,]', description.lower()):
        if word in count:
            count[word] += 1
        else:
            count[word] = 1
    sorted_count = sorted_count = sorted(count.items(), key = lambda x: x[1],␣
 ↪reverse = True)
    return sorted_count

print(sorted_word_count(description))
```

```
[('', 20), ('the', 6), ('data', 6), ('to', 4), ('unit', 3), ('and', 3), ('of',
3), ('methods', 3), ('this', 2), ('students', 2), ('tools', 2), ('science', 2),
('these', 2), ('real-world', 2), ('have', 2), ('set', 2), ('introduces', 1),
('fundamental', 1), ('techniques', 1), ('such', 1), ('as', 1), ('graphical', 1),
('display', 1), ('predictive', 1), ('models', 1), ('evaluation', 1),
('methodologies', 1), ('regression', 1), ('classification', 1), ('clustering',
1), ('provides', 1), ('practical', 1), ('experience', 1), ('applying', 1),
('using', 1), ('industry-standard', 1), ('software', 1), ('sets', 1), ('who',
1), ('completed', 1), ('will', 1), ('be', 1), ('able', 1), ('identify', 1),
('which', 1), ('are', 1), ('most', 1), ('appropriate', 1), ('for', 1), ('a', 1),
('apply', 1), ('interpret', 1), ('results', 1), ('analysis', 1), ('they', 1),
('performed', 1)]
```

1.3 String Manipulation

The next cell defines three strings that you will use in the first group of questions. Note that the first uses single quotes, the second uses double quotes and the third uses three double quotes since it includes newline characters. These are all valid ways of writing strings in Python and are equivalent.

[12]:
```
title = 'Data Science'
code = "MA5851"
description = """This unit introduces students to the fundamental techniques␣
 ↪and
tools of data science for natural language processing. The unit provides␣
 ↪practical
experience applying these methods using industry-standard
software tools to real-world data sets. Students who have
completed this unit will be able to identify which data
science methods are most appropriate for a real-world data
set, apply these methods to the data set, and interpret the
results of the analysis they have performed. """
```

Write code to print the length of these strings.

[13]:
```
print(len(title))       # length of 'title'
print(len(code))        # length of 'code'
print(len(description)) # length of 'description'
```

```
12
```

```
6
490
```

Write code to create a new string in a variable 'summary' that contains the code, title and the first 20 characters of the description, with a ':' character between each one (ie 'MA5851-Data Science-This unit…'

[14]:
```
summary = "-".join([title, code, description[:20]])

# alternative as the instructions are inconsistent as to the required separator
summary2 = ": ".join([title, code, description[:20]])
print(summary)
print(summary2)
```

```
Data Science-MA5851-This unit introduces
Data Science: MA5851: This unit introduces
```

Write code to find the number of words in the description. Hint, this is easy in Python since strings support the split method that returns a list of strings after splitting on whitespace (or another character if you wish). Try split on the string, then find out how many strings are in the resulting list.

[15]:
```
print(len(description.split(" "))) # number of words in the description
```

```
74
```

2 Control Structures

Here you will explore Python control structures - conditionals and loops.

Write a for loop over the words in the description and count how many times the word 'unit' occurs. Your solution will have an if statement inside the for loop.

Here you will encounter Python's required indentation for the first time. This will annoy you at first but you will learn to either love it or hate it with time…

[16]:
```
number_of_appearances = 0

for word in description.split(" "):
    if word == 'unit':
        number_of_appearances += 1
print(number_of_appearances)
```

```
3
```

[17]:
```
# alternative representation using a list comprehension
len([number_of_appearances + 1 for word in description.split(" ") if word ==␣
 ↪'unit'])
```

[17]:
```
3
```

You can iterate over any sequence with a for loop, including the characters in a string. Write a for loop over the characters in the description that prints out 'Comma!' every time it sees a comma.

[23]:
```
for char in description:
    if char == ',':
        print("Comma!")
```

```
Comma!
Comma!
```

[24]:
```
# again using a list comprehension to produce the same result
print(["Comma!" for char in description if char == ","])
```

```
['Comma!', 'Comma!']
```

2.1 Functions

Python is a dynamically typed language so we don't need to declare the type of a variable or declare the return type of a function (although Python 3 introduced optional type hints). Apart from that the idea of writing a function in Python is the same as in Processing or (methods in) Java.

Write a function that takes a single string argument and returns the number of words in the string using the code you wrote above to count words.

[25]:
```
def word_count(string):
    return len(string.split(" "))
```

Use your function to find the number of words in the description string defined above.

[26]:
```
word_count(description)
```

[26]:
```
74
```

2.2 Submission

Submit your workbook to the assessments folder