week2_solution

July 7, 2020

1 Week 2 exercise solutions

1.1 Exercise 1

In the code block below, there is a list of 5 protein sequences, specified in the single amino acid code where one letter corresponds to one amino acid. Write a function that finds the most abundant amino acid in a given protein sequence, but prints a warning message if the protein sequence is shorter than 10 amino acids. Run your function on each of the proteins in the list.

[14]:
```
proteins = [
    "MEAGPSGAAAGAYLPPLQQ",
    "VFQAPRRPGIGTVGKPIKLLANYFEVDIPK",
    "IDVYHY",
    "EVDIKPDKCPRRVNREVV",
    "EYMVQHFKPQIFGDRKPVYDGKKNIYTVTALPIGNER"
]

def find_abundant_aa(protein):
    # check the length of the protein and print a warning if <10 amino acids
    if len(protein) < 10:
        print("WARNING: protein is <10 aa long")
    # count the number of times each amino acid occurs
    aa_abundances = {}
    for i in protein:
        if i in aa_abundances:
            aa_abundances[i] += 1
        else:
            aa_abundances[i] = 1
    # empty string for most abundant aa
    abundant_aa = ""
    # counter to keep track of abundance of the most-abundant aa
    abundance = 0
    for i in aa_abundances:
        if aa_abundances[i] > abundance:
            abundant_aa = i
            abundance = aa_abundances[i]
    # print out the abundances of all amino acids, to allow a manual check that␣
 ↪our counts are correct
```

```
    print(aa_abundances)
    return(abundant_aa)

# print out the most abundant amino acid in each protein
for i in proteins:
    print("Processing protein", i)
    print("The most abundant amino acid is", find_abundant_aa(i))
```

```
Processing protein MEAGPSGAAAGAYLPPLQQ
{'M': 1, 'E': 1, 'A': 5, 'G': 3, 'P': 3, 'S': 1, 'Y': 1, 'L': 2, 'Q': 2}
The most abundant amino acid is A
Processing protein VFQAPRRPGIGTVGKPIKLLANYFEVDIPK
{'V': 3, 'F': 2, 'Q': 1, 'A': 2, 'P': 4, 'R': 2, 'G': 3, 'I': 3, 'T': 1, 'K': 3,
'L': 2, 'N': 1, 'Y': 1, 'E': 1, 'D': 1}
The most abundant amino acid is P
Processing protein IDVYHY
WARNING: protein is <10 aa long
{'I': 1, 'D': 1, 'V': 1, 'Y': 2, 'H': 1}
The most abundant amino acid is Y
Processing protein EVDIKPDKCPRRVNREVV
{'E': 2, 'V': 4, 'D': 2, 'I': 1, 'K': 2, 'P': 2, 'C': 1, 'R': 3, 'N': 1}
The most abundant amino acid is V
Processing protein EYMVQHFKPQIFGDRKPVYDGKKNIYTVTALPIGNER
{'E': 2, 'Y': 3, 'M': 1, 'V': 3, 'Q': 2, 'H': 1, 'F': 2, 'K': 4, 'P': 3, 'I': 3,
'G': 3, 'D': 2, 'R': 2, 'N': 2, 'T': 2, 'A': 1, 'L': 1}
The most abundant amino acid is K
```

1.2 Exercise 2

The code below is intended to specify a function which looks up the capital city of a given country, and call this function on a list of two countries. However, it currently has a bug which stops it running. There are three possibilities for the nature of this bug: 1. Its arguments are in the wrong order 2. It uses a variable that is out of scope 3. It is missing the return statement

What is the bug?

[16]:
```
capital_cities = {
    "Sweden": "Stockholm",
    "UK": "London",
    "USA": "Washington DC"
}

# this version of the function doesn't work because of the argument order
# def find_capital_city(verbose=True, country):
#     if country in capital_cities:
#         capital_city = capital_cities[country]
#         if verbose:
#             print("Capital city located")
```

```
#     else:
#         capital_city = "CAPITAL CITY NOT FOUND"
#     return(capital_city)

# this version of the function does work, as the argument with a default value␣
 ↪is at the end
def find_capital_city(country, verbose=True):
    if country in capital_cities:
        capital_city = capital_cities[country]
        if verbose:
            print("Capital city located")
    else:
        capital_city = "CAPITAL CITY NOT FOUND"
    return(capital_city)

countries = ["USA", "UK", "Sweden", "Belgium"]
for i in countries:
     print(find_capital_city(i))
```

```
Capital city located
Washington DC
Capital city located
London
Capital city located
Stockholm
CAPITAL CITY NOT FOUND
```

1.3 Exercise 3

In the data folder, you will find a file "imagine_lyrics.txt", which contains the lyrics to the song Imagine by John Lennon. Your task is to find out which word is used most frequently in the lyrics. There are many ways to approach this, but however you solve it, remember to **break up your code into functions**!

[4]:
```
# a function to take a text file of lyrics, and return a list of all words in␣
 ↪the lyrics
def get_words(lyric_file):
    words = []
    with open(lyric_file) as lyrics:
        for i in lyrics:
            # remove the new line and split the line into words
            for j in i.strip("\n").split(" "):
                # remove commas from any words and convert to lowercase
                words.append(j.strip(",").lower())
    return(words)

# a function to take a list of words, and return a dictionary of counts per word
def count_words(word_list):
```

```
    word_counts = {}
    for i in word_list:
        if i in word_counts:
            word_counts[i] += 1
        else:
            word_counts[i] = 1
    return(word_counts)

# a function to take a text file of lyrics, and return the most frequent word
def find_most_frequent_word(lyric_file):
    # read the words
    words = get_words(lyric_file)
    # count the words
    counts = count_words(words)
    # find most frequent word
    most_frequent_word = ""
    highest_count = 0
    for i in counts:
        if counts[i] > highest_count:
            most_frequent_word = i
            highest_count = counts[i]
    return(most_frequent_word)

print("The most frequent word is '{}'".format(find_most_frequent_word("../data/
 ↪imagine_lyrics.txt")))
```

```
The most frequent word is 'the'
```

1.4 Exercise 4

Some words aren't very interesting ("the", "a", "and" etc), so we might want to exclude these from consideration when finding the most frequent word in a set of lyrics. Extend your code from Exercise 3 to include an option to exclude a custom list of words, and test how the results change when excluding "the", "a" & "and".

[5]:
```
# a function to take a text file of lyrics, and return a list of all words in␣
 ↪the lyrics
def get_words(lyric_file):
    words = []
    with open(lyric_file) as lyrics:
        for i in lyrics:
            # remove the new line and split the line into words
            for j in i.strip("\n").split(" "):
                # remove commas from any words and convert to lowercase
                words.append(j.strip(",").lower())
    return(words)

# a function to take a list of words, and return a dictionary of counts per word
```

```
def count_words(word_list):
    word_counts = {}
    for i in word_list:
        if i in word_counts:
            word_counts[i] += 1
        else:
            word_counts[i] = 1
    return(word_counts)

# a function to remove words from a list of words
def remove_words(word_list, words_to_remove):
    cleaned_words = []
    for i in word_list:
        if i not in words_to_remove:
            cleaned_words.append(i)
    return(cleaned_words)

# a function to take a text file of lyrics, and return the most frequent word
def find_most_frequent_word(lyric_file, words_to_remove = []):
    # read the words
    words = get_words(lyric_file)
    # exclude words if specified
    if len(words_to_remove)>0:
        words = remove_words(words, words_to_remove)
    # count the words
    counts = count_words(words)
    # find most frequent word
    most_frequent_word = ""
    highest_count = 0
    for i in counts:
        if counts[i] > highest_count:
            most_frequent_word = i
            highest_count = counts[i]
    return(most_frequent_word)

print("Without word filtering, the most frequent word is '{}'".
 ↪format(find_most_frequent_word("../data/imagine_lyrics.txt")))
print("After filtering out 'the', 'a' & 'and', the most frequent word is '{}'".
 ↪format(find_most_frequent_word("../data/imagine_lyrics.txt", ["the", "a",␣
 ↪"and"])))
```

```
Without word filtering, the most frequent word is 'the'
After filtering out 'the', 'a' & 'and', the most frequent word is 'imagine'
```

1.5 Lambda functions

Exercise 1 can be solved using a lambda function to extract the name of the most abundant amino acid. Lambda functions allow you to specify a very short function on the fly, and are sometimes

used as arguments to other functions. For example, we can use a lambda function as the argument to the **sorted()** function, to specify that we want to sort the items in the dictionary of amino acid counts by their abundance:

[20]:
```
proteins = [
    "MEAGPSGAAAGAYLPPLQQ",
    "VFQAPRRPGIGTVGKPIKLLANYFEVDIPK",
    "IDVYHY",
    "EVDIKPDKCPRRVNREVV",
    "EYMVQHFKPQIFGDRKPVYDGKKNIYTVTALPIGNER"
]

def find_abundant_aa(protein):
    # check the length of the protein and print a warning if <10 amino acids
    if len(protein) < 10:
        print("WARNING: protein is <10 aa long")
    # count the number of times each amino acid occurs
    aa_abundances = {}
    for i in protein:
        if i in aa_abundances:
            aa_abundances[i] += 1
        else:
            aa_abundances[i] = 1
    # use lambda functions to sort the dictionary and extract the key for the␣
 ↪highest value
    # the .items() call pulls out the key-value pairs as tuples
    #print(sorted(aa_abundances.items(), key=lambda item: item[1]))
    # the key argument specification is how to sort the items
    # the lambda function specifies that we want to sort the key-value pairs␣
 ↪based on their value (index 1 of each tuple)
    # -1 indexing selects the last key-value pair
    # 0 indexing selects the first item in the tuple i.e. the aa name
    abundant_aa = sorted(aa_abundances.items(), key=lambda item: item[1])[-1][0]
    return(abundant_aa)

# print out the most abundant amino acid in each protein
for i in proteins:
    print("Processing protein", i)
    print("The most abundant amino acid is", find_abundant_aa(i))
```

```
Processing protein MEAGPSGAAAGAYLPPLQQ
The most abundant amino acid is A
Processing protein VFQAPRRPGIGTVGKPIKLLANYFEVDIPK
The most abundant amino acid is P
Processing protein IDVYHY
WARNING: protein is <10 aa long
The most abundant amino acid is Y
Processing protein EVDIKPDKCPRRVNREVV
```

```
The most abundant amino acid is V
Processing protein EYMVQHFKPQIFGDRKPVYDGKKNIYTVTALPIGNER
The most abundant amino acid is K
```

See https://realpython.com/python-lambda/ for a full tutorial