COSC 1336: Fall 2022

Assignment for Chapter 10

DUE: November 15, 2022, by 1 AM

20% PENALTY PER EACH DAY

[10 points] Question 1

Write a Python function **is_homogeneous(mylist)** that returns **True** if all items in the list are of the same type, and **False** otherwise.

Case:

`is_homogeneous([3,4,5])` will return `True`
`is_homogeneous(['three',4,5])` will return `False`
`is_homogeneous([True,False,True])` will return `True`

Example:

```
print(is_homogeneous([3,4,5]))
print(is_homogeneous(['three',4,5]))
print(is_homogeneous([True,False,True]))
```

Output:

```
True
False
True
```

[30 points] Question 2

Write a Python function **common_items(reflist, mylist, inplace)** that returns a list containing items that occur in **mylist** as well as **reflist**.

The parameter **inplace** is a Boolean. If parameter **inplace** is **False**, then the value of **mylist** is unchanged by the function. If **inplace** is **True**, the value of **mylist** becomes same as the list of common items being returned.

Case:

```
Druglist = ['tylenol','ibuprofen','pepto-bismol','sudafed','robitussin']
myneeds = ['glucose','pepto-bismol','tylenol']

inplace = False
common_items(druglist, myneeds, inplace)
```

will return `['pepto-bismol', 'tylenol']`

`print(myneeds)` will print `['glucose', 'pepto-bismol', 'tylenol']`

`inplace = True`
`common_items(druglist, myneeds, inplace)` will return `['pepto-bismol', 'tylenol']`
`print(myneeds)` will print `['pepto-bismol', 'tylenol']`

`Druglist_1 = ['A','E','S','X']`
`Mylist_1 = ['B','S','Y']`
`inplace = False`
`common_items(Druglist_1, Mylist_1, inplace)` will return `['S']`
`print(Mylist_1)` will print `['B','S','Y']`
`inplace = True`
`common_items(Druglist_1, Mylist_1, inplace)` will return `['S']`
`print(Mylist_1)` will print `['S']`

Example:

```
druglist = ['tylenol','ibuprofen','pepto-bismol','sudafed','robitussin']
myneeds = ['glucose', 'pepto-bismol', 'tylenol']
inplace = False
print(common_items(druglist, myneeds, inplace))
print(myneeds)
inplace = True
print(common_items(druglist, myneeds, inplace))
print(myneeds)

druglist_1 = ['A','E','S','X']
mylist_1 = ['B','S','Y']
inplace = False
print(common_items(druglist_1, mylist_1, inplace))
print(Mylist_1)
inplace = True
print(common_items(druglist_1, mylist_1, inplace))
print(Mylist_1)
```

Output:

```
['pepto-bismol', 'tylenol']
['glucose', 'pepto-bismol', 'tylenol']
['pepto-bismol', 'tylenol']
['pepto-bismol', 'tylenol']
['S']
['B', 'S', 'Y']
['S']
['S']
```

[30 points] Question 3

You will write a series of functions that take a string parameter, which is expected to be a paragraph from a book. Assume words in the paragraph are separated by blank spaces. You need to write the following functions:

1. **`para_basics(mypara)`** will compute and return the *average word length*.
2. **`long_words(mypara)`** will return the list of longest words in **`mypara`**. (Note that there can be multiple longest words.)
3. **`special_words(mypara)`** will return a list of special words in **`mypara`**. A "special" word is any word that contains **at least one** of the "rare" letters: **`j, q, x, z`**.
4. **`cool_para(mypara)`** will return **`True`** if at least one of the longest words is also a special word, and **`False`** otherwise.

[Notes: Only use **`split()`** and **`len()`** methods/functions. You may consider any characters other than blanks, including punctuation and numbers, as parts of words.]

Case:

```
mypara = 'This animal is zebraaaaa. Other animal is foxxxxxxx.'
```

1. `para_basics(mypara)` will return `5.625`
2. `long_words(mypara)` will return `['zebraaaaa.', 'foxxxxxxx.']`
3. `special_words(mypara)` will return `['zebraaaaa.', 'foxxxxxxx.']`
4. `cool_para(mypara)` will return `True`

```
mypara = 'This is the second paragraph with a example input for test cases. We
will talk about nutrition today.'
```

1. `para_basics(mypara)` will return `4.667`
2. `long_words(mypara)` will return `['paragraph','nutrition']`
3. `special_words(mypara)` will return `['example']`
4. `cool_para(mypara)` will return `False`

```
mypara = 'In a hole in the ground there lived a hobbit. Not a nasty, dirty, w
et hole, filled with the ends of worms and an oozy smell, no yet a dry, bare,
 sandy hole with nothing in it to sit down on or to eat; it was a hobbit-
hole, and that means comfort.'
```

1. `para_basics(mypara)` will return `3.750`
2. `long_words(mypara)` will return `['hobbit-hole,']`
3. `special_words(mypara)` will return `['oozy']`
4. `cool_para(mypara)` will return `False`

Example:

```
mypara = 'This animal is zebraaaaa. Other animal is foxxxxxxx.'
print('%.3f' %para_basics(mypara))
print(long_words(mypara))
```

```
print(special_words(mypara))
print(cool_para(mypara))

mypara = 'This is the second paragraph with a example input for test cases. We
will talk about nutrition today.'
print('%.3f' %para_basics(mypara))
print(long_words(mypara))
print(special_words(mypara))
print(cool_para(mypara))

mypara = 'In a hole in the ground there lived a hobbit. Not a nasty, dirty, w
et hole, filled with the ends of worms and an oozy smell, no yet a dry, bare,
 sandy hole with nothing in it to sit down on or to eat; it was a hobbit-
hole, and that means comfort.'
print('%.3f' %para_basics(mypara))
print(long_words(mypara))
print(special_words(mypara))
print(cool_para(mypara))
```

Output:

```
5.625
['zebraaaaa.', 'foxxxxxxx.']
['zebraaaaa.', 'foxxxxxxx.']
True
4.667
['paragraph', 'nutrition']
['example']
False
3.750
['hobbit-hole,']
['oozy']
False
```

[30 points] Question 4

Write a function **`find_closest(intlist, key)`** that takes an integer **`key`**, and a list of integers as a parameter, and returns a list containing all locations in **`intlist`** where the value is an integral multiple of the key.

Case:

`intlist = [4, 21, 7, 22, 6]`
`find_closest(intlist, 7)` will return `[1, 2]`
`find_closest(intlist, 2)` will return `[0, 3, 4]`
`find_closest(intlist, 10)` will return `[]`

Example:

```
intlist = [4, 21, 7, 22, 6]
print(find_closest(intlist, 7))
print(find_closest(intlist, 2))
print(find_closest(intlist, 10))
```

Output:

```
[1, 2]
[0, 3, 4]
[]
```