PythonExercises

January 12, 2023

1 Python Exercises

In this assignment, you will write some simple functions that work with lists, strings, and text files. My solutions are between one and seven lines of code.

1.1 Write Your Name Here:

2 Submission instructions

1. Click the Save button at the top of the Jupyter Notebook.
2. Please make sure to have entered your name above.
3. Select Cell -> All Output -> Clear. This will clear all the outputs from all cells (but will keep the content of ll cells).
4. Select Cell -> Run All. This will run all the cells in order, and will take several minutes.
5. Once you've rerun everything, select File -> Download as -> PDF via LaTeX and download a PDF version showing the code and the output of all cells, and save it in the same folder that contains the notebook file.
6. Look at the PDF file and make sure all your solutions are there, displayed correctly.
7. Submit **both** your PDF and the notebook file .ipynb on Canvas.
8. Make sure your your Canvas submission contains the correct files by downloading it after posting it on Canvas.

2.0.1 Simple histogram (10 points)

Write a function my_histogram(l) that takes as input a list l of integers and outputs a dictionary mapping each unique integer in the list to the number of times it appears in the list.

```
[ ]: def my_histogram(l):
         result = {}

         # YOUR CODE HERE




         return result

     # This call should return {1:4, 2:3, 3:1, 6:2}
```

```
my_histogram([1, 2, 1, 3, 2, 1, 6, 2, 6, 1])
```

2.0.2 Cumulative histogram (10 points)

Write a function my_chistogram(l) that takes as input a list l of integers and outputs a dictionary mapping each unique integer x in the list to the number of elements in the list that are less than or equal to x.

[ ]:
```
def my_chistogram(l):
    result = {}

    # YOUR CODE HERE




    return result

# This call should return {1:4, 2:7, 3:8, 6:10}
my_chistogram([1, 2, 1, 3, 2, 1, 6, 2, 6, 1])
```

2.0.3 Bonus (10 points)

Assume the input list l contains only non-negative integers. Write a function my_sort(l) that returns the sorted list and that has time complexity $O(M + N)$, where M is the maximum element in the list and N is the length of the list.

Hint: Use the histograms ...

[ ]:
```
def my_sort(l):
    result = []

    # YOUR CODE HERE




    return result

# This call should return [1, 1, 1, 1, 2, 2, 2, 3, 6, 6]
my_sort([1, 2, 1, 3, 2, 1, 6, 2, 6, 1])
```

2.0.4 Data structures and algorithms (10 points)

Write a Python function `treefun(t)` that takes as input a rooted tree represented as $[root, subtree_1, subtree_2, ..., subtree_n]$ and returns a tuple `(cnodes, leaves)` where `cnodes` is the number of nodes in the tree and `leaves` is a list of all the leaves of the tree, in left to right order.

```
[ ]: def treefun(t):
         result = (0, [])

         # YOUR CODE HERE




         return result


     # This call should return the tuple (9, [3, 4, 5, 8, 9])
     treefun([1, [2, [3], [4]], [5], [6, [7, [8]], [9]]])
```

2.0.5 Matrices (10 points)

Write a function `check_inverse(A, B)` that takes as input two square matrices and determines if they are the inverse of each other. A matrix is represelnted as a list of rows, where each row is represented as a list of numbers.

```
[ ]: def check_inverse(A, B):
         # YOUR CODE HERE


         return False


     # This code should return True.
     A = [[1, 2, 1], [4, 4, 5], [6, 7, 7]]
     B = [[-7, -7, 6], [2, 1, -1], [4, 5, -4]]
     check_inverse(A, B)
```

2.0.6 List comprehensions (10 points)

Define a function filter_elements(l1, l2) that takes as input two lists: a list l1 containing numbers and a list l2 containing Boolean values. The function should return all elements l1[j] such that l2[j] is True (basically, l2 specifies which elements in l1 should be returned). Assume the two lists have the same length. **You should do this with only one line of code.**

```
[ ]: def filter_elements(l1, l2):
         # YOUR CODE HERE

         return []

     # This call should return [1, 2, 4].
     filter_elements([1, 2, 3, 4, 5, 6, 7], [True, True, False, True, False, False,␣
      ↪False])
```

2.0.7 Working with text files (10)

Write a function text_stats(fname) that read a text file **line by line** and returns a tuple containing the following elements:

1. The total number of *lines* in the file.
2. The number of *words* in the file. A *word* is defined as a maximally contiguous sequence of one ore more letters. For example, the string 'The SARS-Covid-19 pandemic started in 2020' contains the words 'The', 'SARS', 'Covid', 'pandemic', 'started', and 'in', hence 6 words.
3. A histogram of the word lengths in the file, i.e. a dictionary that maps integers n to the number words of length n that are in the file. The keys n are between 1 and the maximum word length in the file.

My code has 13 lines.

[ ]:
```
def text_stats(fname):
    # YOUR CODE HERE





    return 0, 0, {}

# This calls should return
#(102,
# 833,
# {4: 230,
#  2: 145,
#  3: 162,
#  5: 91,
#  6: 47,
#  7: 49,
#  1: 44,
#  9: 8,
#  10: 11,
#  8: 33,
#  12: 5,
#  13: 2,
#  11: 6})
text_stats('../data/thinking-meat.txt')
```

[ ]:
```
```