Stats 21 - HW 3 - Due 4/29/2023 by 11:59PM

Blair Lee - 005721089

Homework is generally an opportunity to practice coding and to train your problem solving and critical thinking skills. Putting Python to use is where learning happens.

Copying and pasting another's solutions takes away your learning opportunities. It is also academic dishonesty.

ChatGPT is always allowed in this class, but do remember, it is not foolproof and if your solution looks too much like another submission, I am required to file a report

Please use this document as your homework template and submit both the modified .ipynb file and a PDF export.

Assignment

1. Functions/lists (2pts)

Define a function called `insert_into(listname, index, iterable)`. It will accept three arguments, a currently existing list, an index, and another list (or a tuple) that will be inserted at the index position.

Python's built-in function, `list.insert()` can only insert one object.

In [ ]:
```python
# please write your code here
def insert_into(listname, index, iterable):
    for i in range(0, len(iterable)):
        listname.insert(index + i, iterable[i])
    print(listname)
```

In [ ]:
```python
# do not modify. We will check this result for grading
l = [0,'a','b','c',4,5,6]
i = ['hello', 'there']
insert_into(l, 3, i)
```

```
[0, 'a', 'b', 'hello', 'there', 'c', 4, 5, 6]
```

2. Recursion (3pts)

The Ackermann function, A(m, n), is defined:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

See http://en.wikipedia.org/wiki/Ackermann_function . Write a function named `ack` that evaluates the Ackermann function. Use your function to evaluate a few test cases. Don't test with $m \geq 4$ as it grows very fast very quickly.

```
In [ ]: def ack(m,n):
            if m == 0:
                return n + 1
            elif m > 0 and n == 0:
                return ack(m - 1, 1)
            elif m > 0 and n > 0:
                return ack(m - 1, ack(m, n - 1))
```

```
In [ ]: # test case, should be 61
        ack(3, 3)
```

```
Out[ ]: 61
```

```
In [ ]: # test case, should be 125
        ack(3, 4)
```

```
Out[ ]: 125
```

3. String search (2 pts)

Please download list of words word_list.txt:

https://github.com/lewv/SP23STAT21/blob/main/WEEK03/word_list.txt

Then write code that reads `word_list.txt` and prints out only the words with at least 10 characters.

```
In [ ]: # please write your code here
        with open('word_list.txt', 'r') as f:
            word_list = f.read()
        words = word_list.split()
        ten_char = [word for word in words if len(word) >= 10]
        print(ten_char)
```

```
['administration', 'collection', 'commercial', 'conference', 'democratic', 'd
evelopment', 'difference', 'discussion', 'environment', 'environmental', 'esp
ecially', 'everything', 'experience', 'generation', 'government', 'individua
l', 'information', 'institution', 'interesting', 'international', 'investmen
t', 'management', 'opportunity', 'organization', 'participant', 'particular',
'particularly', 'performance', 'population', 'production', 'professional', 'r
elationship', 'Republican', 'responsibility', 'significant', 'successful', 't
echnology', 'television', 'themselves', 'throughout', 'traditional', 'underst
and']
```

4. Removing duplicates (3 pts)

Write a function called `no_dups` that takes a list and returns a new list with duplicates removed. It should not modify the original list.

You can assume that the list will not have nested lists.

In [ ]:
```python
def no_dups(t):
    no_dupes = []
    for value in t:
        if value not in no_dupes:
            no_dupes.append(value)
    return no_dupes
```

In [ ]:
```python
no_dups(['a','b','c'])
```

Out[ ]:
```
['a', 'b', 'c']
```

In [ ]:
```python
no_dups([-9, -5, 20, 12, 13, 4, 16, -7, 16, 12, 9, 8, 11,
         7, -7, 11, 2, 20, 8, -1, -3, 6, -4, 6, 0, 15, -10,
         2, 6, 4, 7, 1, -1, -1, -5, 5, 15, 14, 12, 9, 0, 2,
         6, -10, 7, 20, 5, -9, -3, 2, 19, 11, -8, 12, 15, 3,
         -2, 11, 14, 8, 5, 10, 8, -3, 2, 19, 4, 11, 20, -2, 0,
         -8, -8, 8, 7, -5, -1, 1, -9, 5, 19, -4, 14, 15, 16, 2,
         5, 8, 8, 15, 7, -10, 3, 10, -10, -6, -9, 6, -7, 9])
```

```
Out[ ]: [-9,
         -5,
         20,
         12,
         13,
         4,
         16,
         -7,
         9,
         8,
         11,
         7,
         2,
         -1,
         -3,
         6,
         -4,
         0,
         15,
         -10,
         1,
         5,
         14,
         19,
         -8,
         3,
         -2,
         10,
         -6]
```

```
In [ ]: no_dups(['a','b','c','a', 16, -7, 16, 12])
```

```
Out[ ]: ['a', 'b', 'c', 16, -7, 12]
```