Homework03

April 21, 2023

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

1.1 Haojie Liu 905943123

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.

1.2 Assignment

1.2.1 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.

[24]:
```
# please write your code here

def insert_into(listname,index,iterable):
    newlist = list(range(0,(len(listname)+len(iterable))))
    i = 0
    for j in range(index-1):
        newlist[i] = listname[j]
        i = i+1
    for j in range(len(iterable)):
        newlist[i] = iterable[j]
        i = i+1
    for j in range(index,len(listname)):
        newlist[i] = listname[j]
        i = i+1
```

```
    print(newlist)
```

[25]:
```
# 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', 'hello', 'there', 'c', 4, 5, 6, 8]
```

1.3 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.

[28]:
```
def ack(m,n):
    if m == 0:
        return(n+1)
    elif m>0 and n==0:
        return(ack(m-1,1))
    else:
        return(ack(m-1,ack(m,n-1)))
```

[29]:
```
# test case, should be 61
ack(3, 3)
```

[29]:
```
61
```

[30]:
```
# test case, should be 125
ack(3, 4)
```

[30]:
```
125
```

1.4 3. String search (2 pts)

Please download list of words word_list.txt:

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

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

[45]:
```
# please write your code here
```

```
file = open("word_list.txt", "r")

contents = file.read().split()


for i in contents:
    if len(i)>=10:
        print(i)
```

```
administration
collection
commercial
conference
democratic
development
difference
discussion
environment
environmental
especially
everything
experience
generation
government
individual
information
institution
interesting
international
investment
management
opportunity
organization
participant
particular
particularly
performance
population
production
professional
relationship
Republican
responsibility
significant
successful
technology
television
```

```
themselves
throughout
traditional
understand
```

1.5 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.

[70]:
```
import string

def no_dups(t):

    newlist = []

    for i in t:
        if i not in newlist:
            newlist.append(i)

    print(newlist)
```

[71]:
```
no_dups(['a','b','c'])
```

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

[72]:
```
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])
```

```
[-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]
```

[73]:
```
no_dups(['a','b','c','a', 16, -7, 16, 12])
```

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