Li_Shuying_hw1

October 6, 2022

1 STA 141B Homework 1

1.1 Writing Functions

Exercise 1.1 (2 points). Write a function `my_median` that takes a list of numbers as input and returns their median as output. Let the number of samples in data set `n`. If `n` is odd, median is the middle number. If `n` is even, median is the average of two middle values. Test your function on one odd sample size and one even sample size example.

Hint: use ***`sorted`*** *function.*

[1]:
```
def my_median(lst):
    lst = sorted(lst)
    n = len(lst)
    # Even: if the reminder is 0 the length of the list is even
    if n % 2 == 0: #
        # Get two middle numbers and calculate their mean
        return (lst[n//2] + lst[n//2 - 1]) / 2 # ((n/2)th + (n/2 + 1)th)/2
    # Odd: if the reminder is not 0
    else:
        return lst[n//2] #Directly get the middle position's number
```

[2]:
```
#Test case - odd - Exercise 1.1
## ONLY USE IN TEST CASES
import statistics

print(my_median([1,2,3]))
print(statistics.median([1,2,3]) == my_median([1,2,3]))
```

```
2
True
```

[3]:
```
#Test cases - even - Exercise 1.1
print(my_median([1,2,3,4]))
print(statistics.median([1,2,3,4]) == my_median([1,2,3,4]))
```

```
2.5
True
```

Exercise 1.2 (2 points). For the function you wrote in Exercise 1.3, what happens if the input list is empty or contains non-numeric elements? Create a new version of your function called

`better_median` that returns `None` when either of these unusual cases occur. (Be cautious not to print `"None"` string. The output should be empty.)

Hint: A similar problem is discussed in Section 6.8 of Think Python.

[4]:
```
def better_median(lst):
    if len(lst) > 0:
        # To test each item in the list
        for item in lst:
            # if one of the item is a string
            if isinstance(item, str) == True:
                return None
        # calculate the median from Exercise 1.1
        return my_median(lst)
    else: #If the length of the list is equal or less then 0
        return None
```

Complete code:

```
def better_median(lst):
    if len(lst) > 0:
        for item in lst:
            if isinstance(item, str) == True:
                return None
        lst = sorted(lst)
        n = len(lst)
        # Even
        if n % 2 == 0:
            return (lst[n//2] + lst[n//2 - 1]) / 2
        # Odd
        else:
            return lst[n//2]
    else:
        return None
```

[5]:
```
#Test case - Exercise 1.2
better_median([])
```

[6]:
```
#Test case - Exercise 1.2
better_median([1,2,3,'little', 'box'])
```

[7]:
```
#Test case - Exercise 1.2
print(better_median([1,3,3,5,6,7,2,3]))
print(statistics.median([1,3,3,5,6,7,2,3]) == better_median([1,3,3,5,6,7,2,3]))
```

```
3.0
True
```

[8]:
```
#Test case - Exercise 1.2
print(better_median([2,3,4]))
print(statistics.median([2,3,4]) == better_median([2,3,4]))
```

```
3
True
```

Exercise 1.3 (1 points). Read Section 4.9 of Think Python. Create a new version of your function from Exercise 1.4 called `best_median` that includes a docstring explaining how to use the function. Give an example to show that your docstring works with Python's built-in help system.

[9]:
```
def best_median(lst):
    """The best median function return the median (middle value) of numeric data

    The function will return to None if:
    1. the list contains non-numeric elements
    2. the length of the list is less or equal to 0

    If the length of the numeric list or array is odd, median is the middle␣
 ↪number.
    If the length of the numeric list or array is even, median is the average␣
 ↪of two middle values.

    Example:
    >>> best_median([1,2,2])
    2

    >>> best_median([1,1,'little', 'box'])
    None
    """
    return better_median(lst)
```

Complete Code:

```
def best_median(lst):
    """The best median function return the median (middle value) of numeric data

    The function will return to None if:
    1. the list contains non-numeric elements
    2. the length of the list is less or equal to 0

    If the length of the numeric list or array is odd, median is the middle number.
    If the length of the numeric list or array is even, median is the average of two middle va

    Example:
    >>> best_median([1,2,2])
    2

    >>> best_median([1,1,'little', 'box'])
```

```
    None
    """
    if len(lst) > 0:
        for item in lst:
            if isinstance(item, str) == True:
                return None
        lst = sorted(lst)
        n = len(lst)
        # Even
        if n % 2 == 0:
            return (lst[n//2] + lst[n//2 - 1]) / 2
        # Odd
        else:
            return lst[n//2]
    else:
        return None
```

[10]:
```
help(best_median)
```

```
Help on function best_median in module __main__:

best_median(lst)
    The best median function return the median (middle value) of numeric data

    The function will return to None if:
    1. the list contains non-numeric elements
    2. the length of the list is less or equal to 0

    If the length of the numeric list or array is odd, median is the middle
number.
    If the length of the numeric list or array is even, median is the average of
two middle values.

    Example:
    >>> best_median([1,2,2])
    2

    >>> best_median([1,1,'little', 'box'])
    None
```

1.2 1.2 Factorial

The Factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n.

For example,

5! = 5 x 4 x 3 x 2 x 1 = 120.

1.2.1 Exercise 1.4

Exercise 1.4 (2 points). Write a function `fact` that computes the factorial without recursion. Your function should take an argument `n`.

Use your function to print the 18! and 22!.

[11]:
```
def fact(n):
    # Initilization
    sum = 1
    # Use for loop to times from 1 to n
    for i in range(1, n + 1):
        sum *= i
    return sum
```

[12]:
```
# Test Case - Exercise 1.4
# ONLY USE IN TEST CASES
import math

print(fact(18))
print(math.factorial(18) == fact(18))
```

```
6402373705728000
True
```

[13]:
```
# Test Case - Exercise 1.4
print(fact(22))
print(math.factorial(22) == fact(22))
```

```
1124000727777607680000
True
```

1.3 1.3 Fibonacci Words

A Fibonacci word is a string of 0s and 1s constructed by repeatedly concatenating strings. The first two words are

```
S0 = "0"
S1 = "01"
```

Then each word is formed by concatenating the previous two words in the sequence. For instance, `S2`, is formed by concatenating `S1` and `S0`. So

```
S2 = "010"
S3 = "01001"
```

Exercise 1.5 (3 points). Write a function `fib` that computes Fibonacci words. Your function should take an argument `n` that specifies which word to compute.

Use your function to print the first 10 Fibonacci words.

[14]:
```
def fib(n):
    """ Compute nth element of a fibonacci words return nth fibonacci words
    """
    # Initilization
    s0 = '0'
    s1 = '01'
    for i in range(n):
        """
        Basic Logic: s0 returns to the next, and s1 contacts to the previous one
        -------
        s0 | s1
        -------
        0  | 01
        01 | 010
        010| 01001

        The importance of this code is to make sure they can run at the same␣
 ↪time,
        in other words, s1 + s0 (before changing to the next one)
        """
        s0, s1 = s1, s1 + s0
    return s0
```

[15]:
```
# Test Case - Exercise 1.5
for i in range(0,10):
    print(fib(i))
```

```
0
01
010
01001
01001010
0100101001001
01001010010010100
0100101001001010010100100101001001
0100101001001010010100100101001001010010100100101001010010100100101001010
0100101001001010010100100101001001010010100100101001010010100100101001010010100100101001010010100100
101001001
```

[16]:
```
help(fib)
```

```
Help on function fib in module __main__:

fib(n)
    Compute nth element of a fibonacci words return nth fibonacci words
```