Li_Shuying_hw1

October 6, 2022

1 STA 141B Homework 1

1.1 Writing Functions

1.1.1 Exercise 1.1

- If `n` is odd, median is the middle number.
- if `n` is even, median is the average of two middle values.

[1]:
```
def my_median(lst):
    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]
```

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

statistics.median([1,2,3,4,5,6]) == my_median([1,2,3,4,5,6])
```

[2]:
```
True
```

[3]:
```
#Test cases - odd - Exercise 1.1
statistics.median([1,2,3,4,5]) == my_median([1,2,3,4,5])
```

[3]:
```
True
```

1.1.2 Exercise 1.2

- If the input list is empty or contains non-numeric elements, return `None`

[4]:
```
def better_median(lst):
    if len(lst) > 0:
        for item in lst:
            if isinstance(item, str) == True:
```

```
                return None
        return my_median(lst)
    else:
        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(statistics.median([1,3,3,5,6,7,2,3]) == better_median([1,3,3,5,6,7,2,3]))
better_median([1,3,3,5,6,7,2,3])
```

```
True
```

[7]:
```
3.0
```

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

```
True
```

[8]:
```
3
```

1.1.3 Exercise 1.3

- 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)
```

[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

1.2.1 Exercise 1.4

- Write a function `fact` that computes the factorial without recursion. Your function should take an argument `n`.

[11]:
```
def fact(n):
    sum = 1
    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

1.3.1 Exercise 1.5

- Write a function `fib` that computes Fibonacci words. Your function should take an argument `n` that specifies which word to compute.

[14]:
```
def fib(n):
    """ Compute nth element of a fibonacci words return nth fibonacci words
    """
    s0 = '0'
    s1 = '01'
    for i in range(n):
        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
010010100100101001010
0100101001001010010100100101001001
010010100100101001010010010100100101001010010100100101001010010010100100101001010
0100101001001010010100100101001001010010100100101001010010100100101001010010010100100101001010010100100
101001001
```

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

```
Help on function fib in module __main__:

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