Python Basics - Experiment 1- 1st Semester- Feb 2021

Aim – Write a menu-Driven text Applications to solve five problems as a menu-driven textbased application. It presents the user with a set of choices

(1) sum of input numbers, (2) average of input numbers, (3) mean of input numbers, (4) median of input numbers, (2) mode of input numbers and (X) Quit.

In [1]:

```
import statistics
import sys

def sumup():
    print('sum is', sum(sum_lst))

def avggear():
    avg = sum(sum_lst) / len(sum_lst)
    print("Average is:", avg)

def mean():
    x = statistics.mean(sum_lst)
    print("Mean is :", x)

def median():
    x = statistics.median(sum_lst)
    print("Median is :", x)

def mode():
    try:
        x = statistics.mode(sum_lst)
        print("Mode is :", x)
    except StatisticsError:
        print("There is no mode")

def exit():
    sys.exit


print("Welcome to number operations")
print("1.Sum input list")
print("2.Average input list")
print("3.Mean input list")
print("4.Median input list")
print("5.Mpde input list")
print("6.Exit")

sum_lst = []
n = int(input("Enter number of elements : "))
for i in range(0, n):
        ele = int(input("Enter %d input:" % i))
        sum_lst.append(ele)

inp = int(input('input a choice : '))
if inp == 1:
    sumup()
elif inp == 2:
    avggear()
elif inp == 3:
    mean()
elif inp == 4:
    median()
elif inp == 5:
    mode()
elif inp == 6:
    exit()
else:
    print("Wrong Choice")
```

```
Welcome to number operations
1.Sum input list
```

```
1.Sum input list
2.Average input list
3.Mean input list
4.Median input list
5.Mpde input list
6.Exit
Enter number of elements : 2
Enter 0 input:1
Enter 1 input:2
input a choice : 1
sum is 3
```

Edmodo - file_io_exercise

2. Reading first word from each line of a file

Implement `find_first_words` function which takes an input file path as argument. The function should find the first word of each line in the file and return these words as a list. If a line is empty, the returned list should contain an empty string for that line.

In [2]:

```python
import os
DATA_DIR = r'C:\Users\Hamza'

def find_first_words(file):
    WordList=[]
    with open(file,mode='r') as x:
        for word in x:
            word=word.strip()
            WordList.append(word.split(" ",1)[0])
    return WordList
```

In [3]:

```python
in_file1 = os.path.join(DATA_DIR, 'simple_file.txt')
```

In [4]:

```python
find_first_words(in_file1)
```

Out[4]:

```
['First', 'Second', 'Third', 'And']
```

In [5]:

```python
in_file2 = os.path.join(DATA_DIR, 'simple_file_with_empty_lines1.txt')
```

In [6]:

```python
find_first_words(in_file2)
```

Out[6]:

```
['The', '', 'First', 'Funny', '', 'Then']
```

In [7]:

```python
in_file1 = os.path.join(DATA_DIR, 'simple_file.txt')
in_file2 = os.path.join(DATA_DIR, 'simple_file_with_empty_lines1.txt')
expected_file_1 = ['First', 'Second', 'Third', 'And']
assert find_first_words(in_file1) == expected_file_1

expected_file_2 = ['The', '', 'First', 'Funny', '', 'Then']
assert find_first_words(in_file2) == expected_file_2
```

1. Sum numbers listed in a file

1) Fill __ pieces of the code below. sum_numbers_in_file function takes a input file path as argument, reads the numbers listed in the input file and returns the sum of those numbers. You can assume that each line contains exactly one numeric value.

In [2]:

```
import os
DATA_DIR = r'C:\Users\Hamza'
def sum_numbers_in_file(input_file):
    sum_ = 0  # A common way to use variable names that collide with built-in/keyword words is to
add underscore
    with open(input_file, mode='r') as x:
        for line in x:
            x = line.strip()  # Remove potential white space
            sum_ += float(line)

    return sum_

in_file = os.path.join(DATA_DIR, 'numbers.txt')
assert sum_numbers_in_file(in_file) == 189.5
sum_numbers_in_file(in_file) == 189.5
```

Out[2]:

```
True
```

Edmodo Numbers_exercise

1. Creating formulas

Write the following mathematical formula in Python:

$$result = 6a^3 - \frac{8b^2}{4c} + 11$$

In [52]:

```
a = 2
b = 3
c = 2
result = (6*(a)**3) - (8*((b)**2))/(4*c) + 11
if result == 50:
    print(True)
```

```
True
```

2. Floating point pitfalls

Show that `0.1 + 0.2 == 0.3`

In [1]:

```
x=0.1
y=0.2
sum = x+y
sum=round(sum,2)
if sum == 0.3:
    print(True)
```

```
True
```

In [ ]: