Assignment 1

1. Creating formulas

Write the following mathematical formula in Python:

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

In [1]:

```
a = 2
b = 3
c = 2
```

In [2]:

```
# Your formula here:
result = (6 * (a ** 3)) - ((8 * (b ** 2))/(4 * c)) + (11)
```

In [3]:

```
result
```

Out[3]:

```
50.0
```

In [4]:

```
assert result == 50
```

2. Floating point pitfalls

Show that `0.1 + 0.2 == 0.3`

In [5]:

```
# Your solution here

from decimal import Decimal
d1=Decimal('0.1')
d2=Decimal('0.2')
sum_of_decimals = d1 + d2
print(sum_of_decimals == Decimal('0.3'))

# This won't work:
# assert 0.1 + 0.2 == 0.3
```

```
True
```

Assignment 2

In [6]:

```
# EXECUTE THIS ONE FIRST!

import os

# Constants for the exercises:
WORKING_DIR = os.getcwd()
DATA_DIR = os.path.join(os.path.dirname(WORKING_DIR), 'data')
```

In [7]:

```
print(DATA_DIR)
```

```
C:\Users\mushrifah\Desktop\mtech sem1\APL\data
```

1. Sum numbers listed in a file

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 [8]:

```
def sum_numbers_in_file(input_file):
    sum_ = 0  # A common way to use variable names that collide with built-in/keyword w
ords is to add underscore
    with open(input_file, 'r') as f:
        for line in f:
            line = line.strip()  # Remove potential white space
            sum_ += float(line)
    return sum_
```

In [9]:

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

In [10]:

```
sum_numbers_in_file(in_file)
```

Out[10]:

```
189.5
```

In [11]:

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

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 [12]:

```
# Your implementation here
def find_first_words(input_file):
    l1=[]
    with open(input_file,'r') as fh:
        for line in fh:
            line=line.strip()
            l1.append(line.split(" ",1)[0])
    return l1
```

In [13]:

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

In [14]:

```
find_first_words(in_file1)
```

Out[14]:

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

In [15]:

```
in_file2 = os.path.join(DATA_DIR, 'simple_file_with_empty_lines_1.txt')
```

In [16]:

```
find_first_words(in_file2)
```

Out[16]:

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

In [17]:

```
in_file1 = os.path.join(DATA_DIR, 'simple_file.txt')
in_file2 = os.path.join(DATA_DIR, 'simple_file_with_empty_lines_1.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
```

Experiment 1:

Aim – Write a menu-Driven text Applications to solve five problems as a menu-driven text- based application.

It presents the user with a set of choices (that, e.g., (1) sum of input numbers, (2) average of input numbers, (3) mean of input numbers, (4) median of input numbers, (5) mode of input numbers and (X) Quit. The user makes a selection, which is then executed. The program exits when the user chooses the “quit” option. The great advantage of a program like this is that it allows the user to run as many iterations of your solutions without necessarily having to restart the same program over and over again.

In [18]:

```
lst=[]
n=int(input("Enter the number of elements"))
for i in range(0,n):
    l=int(input())
    lst.append(l)
print(lst)

import statistics as s
from statistics import StatisticsError
#lst=[34,30,32,40,45,30,36,41]
while True:
    print("Menu Driven Program")
    print("1.Sum of numbers")
    print("2.Average of numbers")
    print("3.Mean of numbers")
    print("4.Median of numbers")
    print("5.Mode of numbers")
    print("6.Quit")
    choice=int(input("Enter your choice:"))
    if choice==1:
        print("Sum of numbers",sum(lst))
    elif choice==2:
        print("Average of numbers",sum(lst)/len(lst))
    elif choice==3:
        print("mean of numbers:",s.mean(lst))
    elif choice==4:
        print("Median of numbers:",s.median(lst))
    elif choice==5:
        try:
            print("Mode of numbers:",s.mode(lst))

        except StatisticsError:
            print("no unique mode found")

    elif choice==6:
        break
    else:
        print("Wrong Choice")
```

```
Enter the number of elements5
1
2
3
4
5
[1, 2, 3, 4, 5]
Menu Driven Program
1.Sum of numbers
2.Average of numbers
3.Mean of numbers
4.Median of numbers
5.Mode of numbers
6.Quit
Enter your choice:1
Sum of numbers 15
Menu Driven Program
1.Sum of numbers
2.Average of numbers
3.Mean of numbers
4.Median of numbers
5.Mode of numbers
6.Quit
Enter your choice:2
Average of numbers 3.0
Menu Driven Program
1.Sum of numbers
2.Average of numbers
3.Mean of numbers
4.Median of numbers
5.Mode of numbers
6.Quit
Enter your choice:3
mean of numbers: 3
Menu Driven Program
1.Sum of numbers
2.Average of numbers
3.Mean of numbers
4.Median of numbers
5.Mode of numbers
6.Quit
Enter your choice:4
Median of numbers: 3
Menu Driven Program
1.Sum of numbers
2.Average of numbers
3.Mean of numbers
4.Median of numbers
5.Mode of numbers
6.Quit
Enter your choice:5
no unique mode found
Menu Driven Program
1.Sum of numbers
2.Average of numbers
3.Mean of numbers
4.Median of numbers
5.Mode of numbers
6.Quit
Enter your choice:6
```