Welcome to the Python_and_Deep_Learning_Course-CSEE5590 Lab-1 submission

```
                                                                                  By:
Mani Sai Srinivas, Kandukuri

Class-id: 20
```

Objective

This lab mainly consists of basics in python like strings,loops, Conditional Statements, Lists etc.

All of the code is commented for understanding. Code can be found here

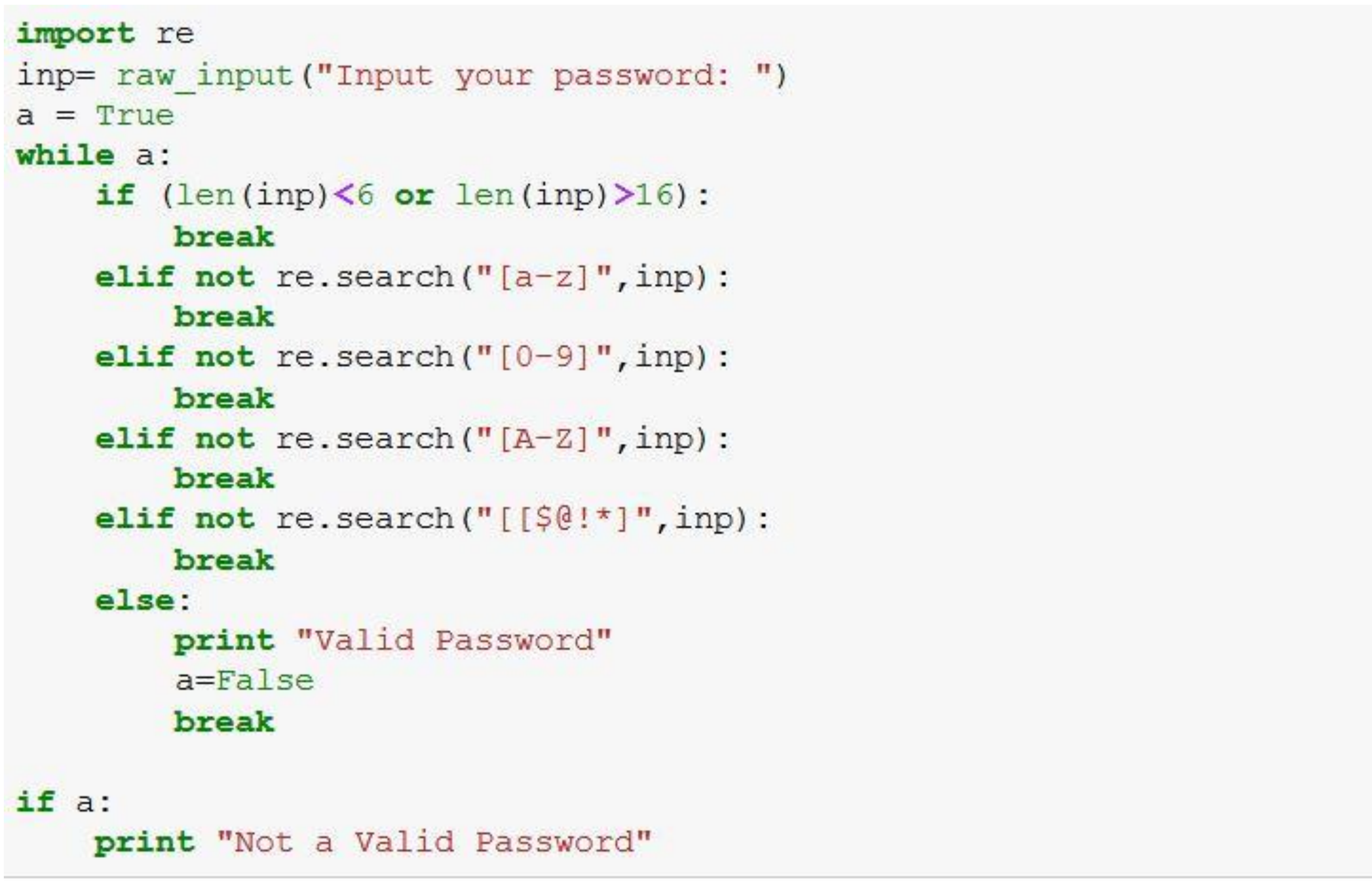
Configuration

- Python 2.7
- Jupyter Notebook

Task-1

The task is to evaluate password. The user input was taken and tested. Here regular expression operations module was imported and each criteria was solved using if and elif statements in python.

Here is the screenshot of the code:

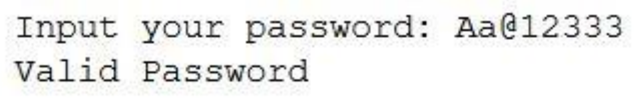


```
import re
inp= raw_input("Input your password: ")
a = True
while a:
    if (len(inp)<6 or len(inp)>16):
        break
    elif not re.search("[a-z]",inp):
        break
    elif not re.search("[0-9]",inp):
        break
    elif not re.search("[A-Z]",inp):
        break
    elif not re.search("[[$@!*]",inp):
        break
    else:
        print "Valid Password"
        a=False
        break

if a:
    print "Not a Valid Password"
```

Code Snippet:

```
import re
inp= raw_input("Input your password: ")
a = True
while a:
    if (len(inp)<6 or len(inp)>16):
        break
    elif not re.search("[a-z]",inp):
        break
    elif not re.search("[0-9]",inp):
        break
    elif not re.search("[A-Z]",inp):
        break
    elif not re.search("[[$@!*]",inp):
        break
    else:
        print "Valid Password"
        a=False
        break

if a:
    print "Not a Valid Password"
```

Output:

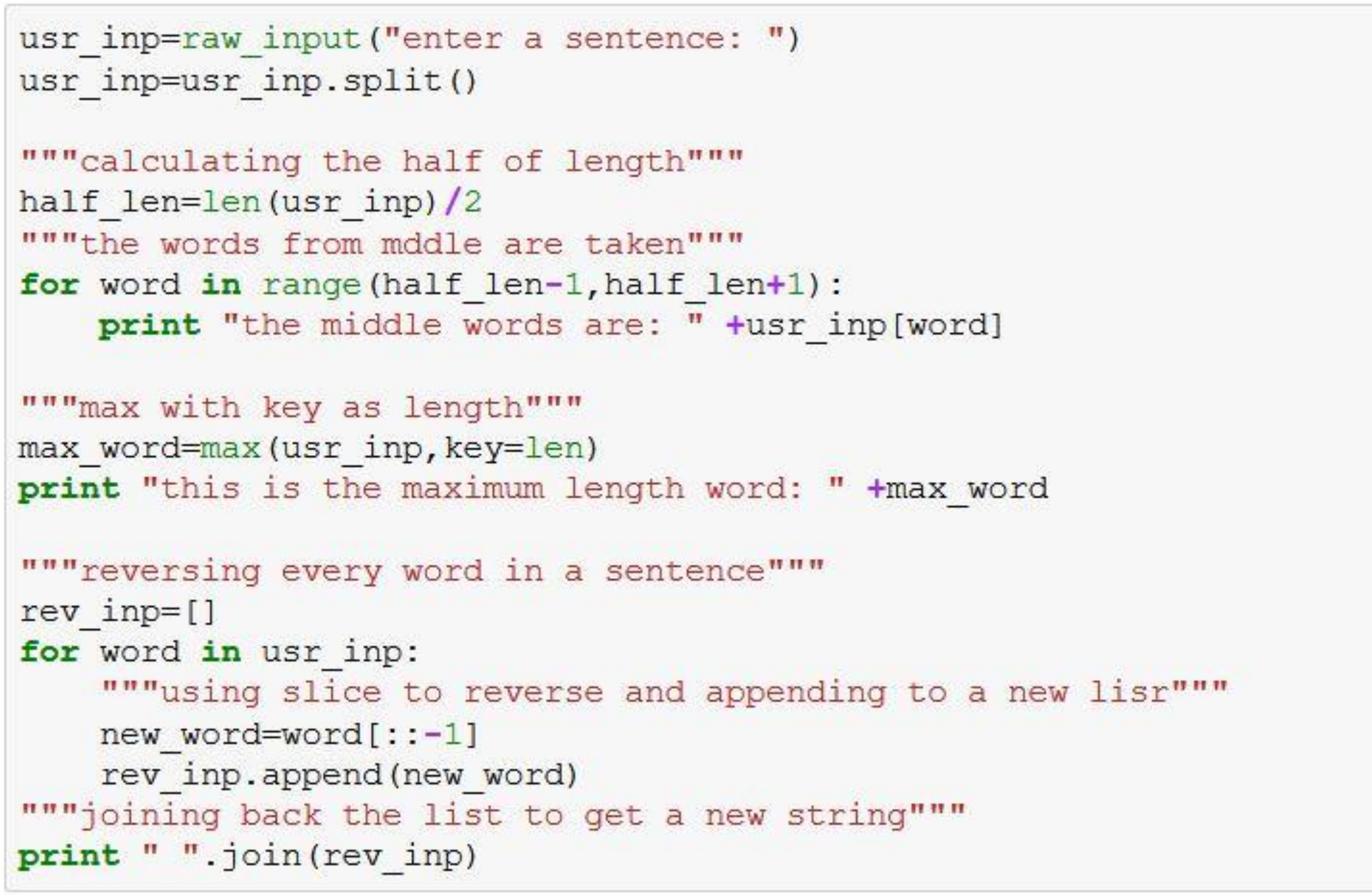


```
Input your password: Aa@12333
Valid Password
```

Task-2

The task is to take a sentence and find the middle, longest words and also reverse all the words in the sentence. A list was created and the half way was found. then using range operator middle word was found. Using max operator the maximum length was found. [::-1] was used to reverse the words and appended to a new list.

This is a screenshot of the code:

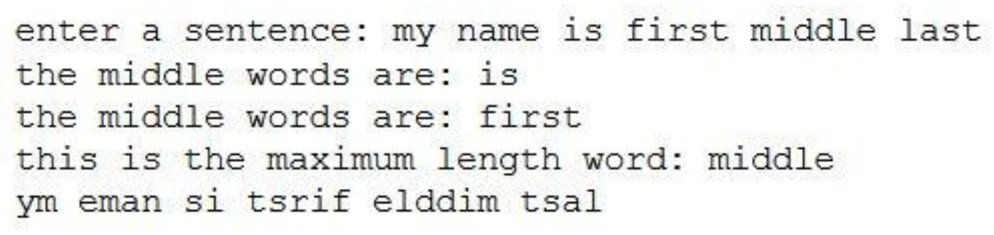


```
usr_inp=raw_input("enter a sentence: ")
usr_inp=usr_inp.split()

"""calculating the half of length"""
half_len=len(usr_inp)/2
"""the words from mddle are taken"""
for word in range(half_len-1,half_len+1):
    print "the middle words are: " +usr_inp[word]

"""max with key as length"""
max_word=max(usr_inp,key=len)
print "this is the maximum length word: " +max_word

"""reversing every word in a sentence"""
rev_inp=[]
for word in usr_inp:
    """using slice to reverse and appending to a new lisr"""
    new_word=word[::-1]
    rev_inp.append(new_word)
"""joining back the list to get a new string"""
print " ".join(rev_inp)
```

Code Snippet:

```
usr_inp=raw_input("enter a sentence: ")
usr_inp=usr_inp.split()

`usr_inp=raw_input("enter a sentence: ")
usr_inp=usr_inp.split()

"""calculating the half of length"""
 half_len=len(usr_inp)/2
"""the words from mddle are taken"""
for word in range(half_len-1,half_len+1):
    print "the middle words are: " +usr_inp[word]

"""max with key as length"""
max_word=max(usr_inp,key=len)
print "this is the maximum length word: " +max_word

"""reversing every word in a sentence"""
rev_inp=[]
for word in usr_inp:
    """using slice to reverse and appending to a new lisr"""
     new_word=word[::-1]
     rev_inp.append(new_word)
"""joining back the list to get a new string"""
print " ".join(rev_inp)`
```

Output:

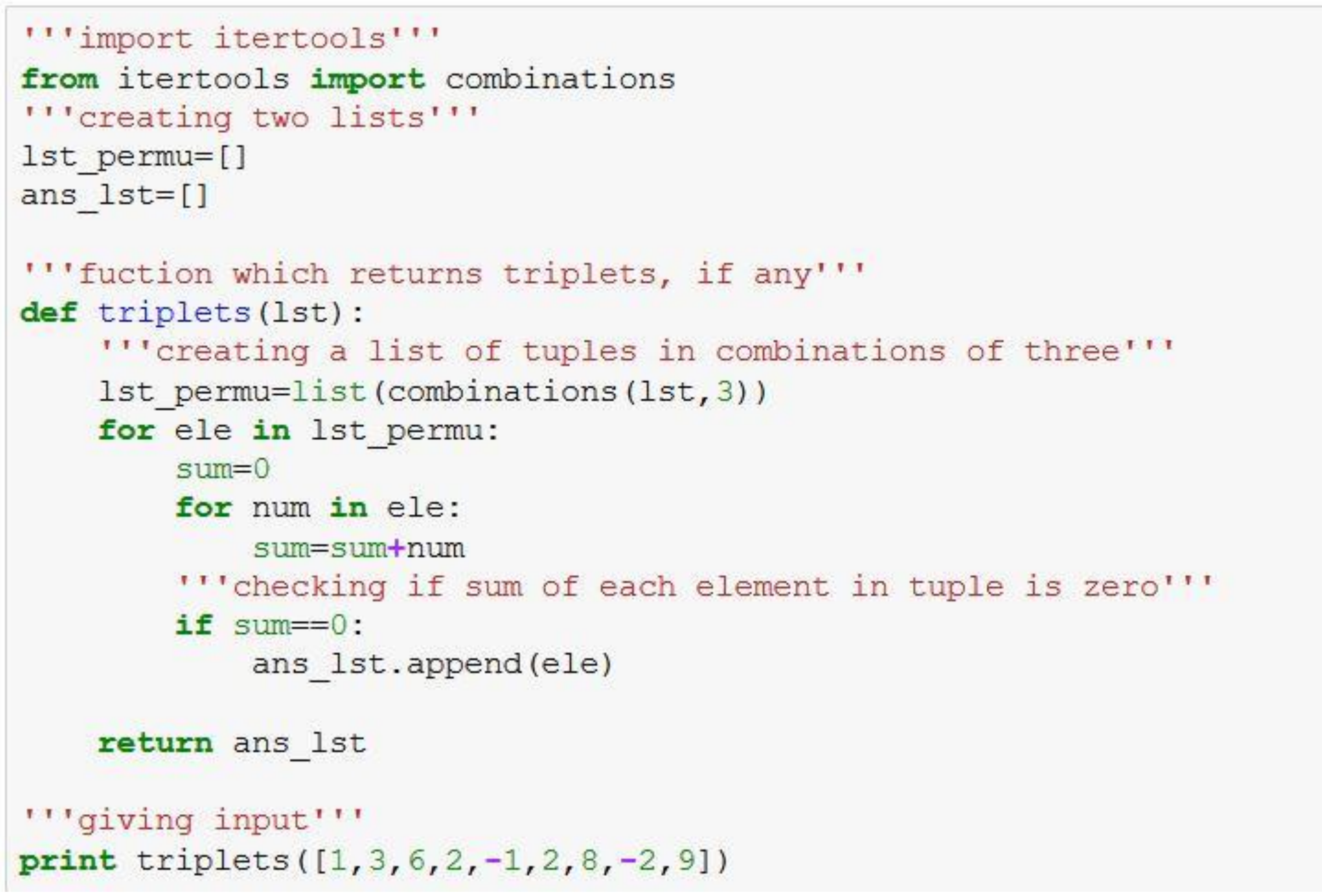


```
enter a sentence: my name is first middle last
the middle words are: is
the middle words are: first
this is the maximum length word: middle
ym eman si tsrif elddim tsal
```

Task-3

Task is to find triplets whose sum is zero. Here itertools was used and we took advantage of itertools.combinatins here

Here is screenshot for the code:

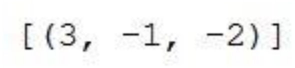


```python
'''import itertools'''
from itertools import combinations
'''creating two lists'''
lst_permu=[]
ans_lst=[]

'''fuction which returns triplets, if any'''
def triplets(lst):
    '''creating a list of tuples in combinations of three'''
    lst_permu=list(combinations(lst,3))
    for ele in lst_permu:
        sum=0
        for num in ele:
            sum=sum+num
        '''checking if sum of each element in tuple is zero'''
        if sum==0:
            ans_lst.append(ele)

    return ans_lst

'''giving input'''
print triplets([1,3,6,2,-1,2,8,-2,9])
```

Code Snippet:

```python
'''import itertools'''
from itertools import combinations
'''creating two lists'''
lst_permu=[]
ans_lst=[]

'''fuction which returns triplets, if any'''
def triplets(lst):
    '''creating a list of tuples in combinations of three'''
    lst_permu=list(combinations(lst,3))
    for ele in lst_permu:
        sum=0
        for num in ele:
            sum=sum+num
        '''checking if sum of each element in tuple is zero'''
        if sum==0:
            ans_lst.append(ele)

    return ans_lst

'''giving input'''
print triplets([1,3,6,2,-1,2,8,-2,9])
```

Output:

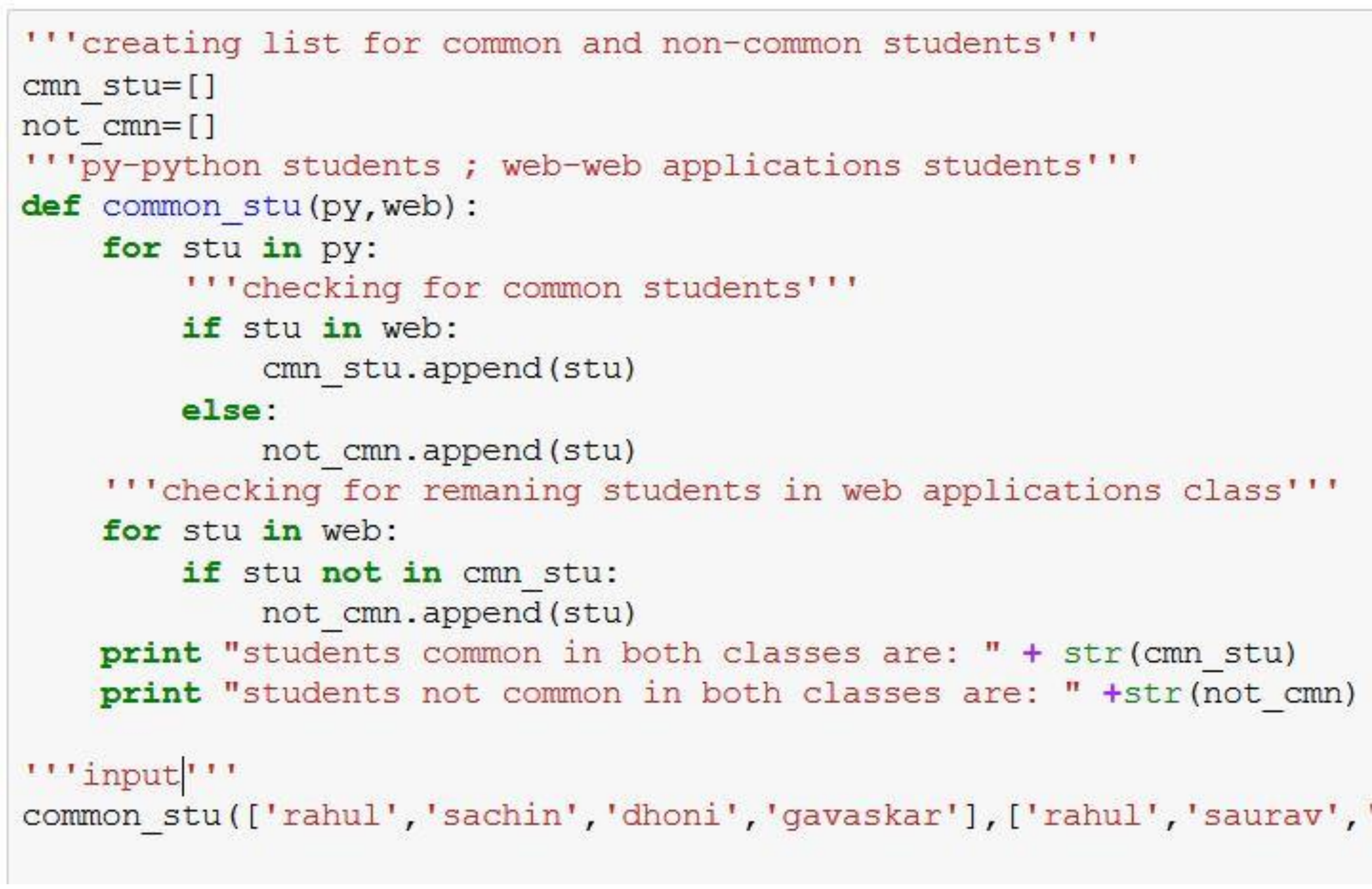


```
[(3, -1, -2)]
```

Task-4

Find common students in "web applications" and "python" course. Loops were used to find if there are common students in both the courses taking advantage of the list structure.

Screenshot of the code:

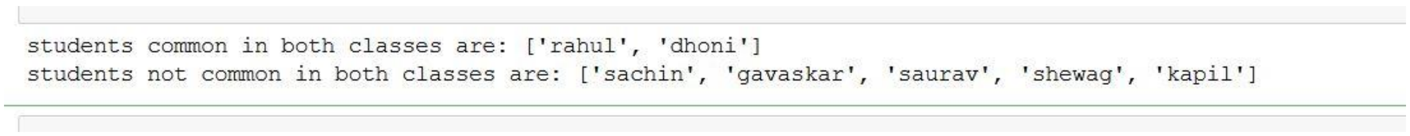


```
'''creating list for common and non-common students'''
cmn_stu=[]
not_cmn=[]
'''py-python students ; web-web applications students'''
def common_stu(py,web):
    for stu in py:
        '''checking for common students'''
        if stu in web:
            cmn_stu.append(stu)
        else:
            not_cmn.append(stu)
    '''checking for remaning students in web applications class'''
    for stu in web:
        if stu not in cmn_stu:
            not_cmn.append(stu)
    print "students common in both classes are: " + str(cmn_stu)
    print "students not common in both classes are: " +str(not_cmn)

'''input'''
common_stu(['rahul','sachin','dhoni','gavaskar'],['rahul','saurav','
```

Code Snippet:

```
cmn_stu=[]
not_cmn=[]
'''py-python students ; web-web applications students'''
def common_stu(py,web):
    for stu in py:
        '''checking for common students'''
        if stu in web:
            cmn_stu.append(stu)
       else:
        not_cmn.append(stu)
'''checking for remaning students in web applications class'''
    for stu in web:
        if stu not in cmn_stu:
            not_cmn.append(stu)
    print "students common in both classes are: " + str(cmn_stu)
    print "students not common in both classes are: " +str(not_cmn)

'''input'''
common_stu(['rahul','sachin','dhoni','gavaskar'],['rahul','saurav','shewag','
kapil','dhoni'])
```

Output:



```
students common in both classes are: ['rahul', 'dhoni']
students not common in both classes are: ['sachin', 'gavaskar', 'saurav', 'shewag', 'kapil']
```

Limitations:

- Many modules were imported here, but the programs can be done without importing modules too.

References: https://stackoverflow.com/