In [1]:
```
#Scenario:

#jerome(a teacher at grand rapids school) teaches:
# Math, chem, bio and phy for 8th grade students
# the school term which consists of 96 class sessions with attendance taken each day
# each class attendance = 1pt

# jerome computes the following instances and categories;
# 1.
# %class attendance
#and award max. 12pts for the class attendance performance to [the total performance f

 # 2.
#with class quizzes every week
# * 19quizzes in total == 19 weeks (of the term)
# each quiz is graded over 10 max pts
# with a max aggregrate of 30 for the 'quiz contribution' to [the total performance fo

# 3.
# 19 homework assigned in total
# each graded over max 10pts
# and max aggregrate score of 15pts

# 4.
# while exams account for the remaining points


# so from the information above, we have four categories of jerome computation:
# 1. class attendance
# 1pts each for 96class sessions
# with max aggregate score of 12pts
#2. 19 class quizzes
# each quiz graded over 10pts
#with overall aggregrate of max 30pts
#3. 19 homework
# each homework graded over max 10pts
#with max aggregrate score of 15pt   and finally

#4. exams accounting for the remaining points for the school term performance
```

In [2]:
```
studName = ('bett james', 'namukolo abrams', 'vera abutu', 'kwame doga', 'lukman ahmad
         'luke brant', 'james kenyata', 'ngugi tionga', 'okoro eze', 'agatha chiluba',
        'longe jethro', 'florence giwa', 'veiva lucent', 'melody braimoh', 'victor iha
        'maguel peter', 'wellington zuba')
studID = ('GR-0483', 'GR-0484', 'GR-0485', 'GR-0486', 'GR-0487', 'GR-0488', 'GR-0489',
         'GR-0491', 'GR-0492', 'GR-0493', 'GR-0494', 'GR-0495', 'GR-0496', 'GR-0497',
         'GR-0499', 'GR-0500', 'GR-0501', 'GR-0502')

studRefs = dict(zip(studName, studID))
```

In [3]:
```
studRefs['bett james']
```

Out[3]:
```
'GR-0483'
```

In [4]:
```
studRefs.keys()
```

Out[4]:
```
dict_keys(['bett james', 'namukolo abrams', 'vera abutu', 'kwame doga', 'lukman ahma
d', 'akin torey', 'luke brant', 'james kenyata', 'ngugi tionga', 'okoro eze', 'agatha
chiluba', 'mangu joseph', 'longe jethro', 'florence giwa', 'veiva lucent', 'melody br
aimoh', 'victor ihab', 'mim trucker', 'maguel peter', 'wellington zuba'])
```

In [5]:
```
import numpy as np
import pandas as pd
```

In [6]:
```
name = ('bett james', 'namukolo abrams', 'vera abutu', 'kwame doga', 'lukman ahmad', '
         'luke brant', 'james kenyata', 'ngugi tionga', 'okoro eze', 'agatha chiluba',
        'longe jethro', 'florence giwa', 'veiva lucent', 'melody braimoh', 'victor iha
        'maguel peter', 'wellington zuba')

ID = ('GR-0483', 'GR-0484', 'GR-0485', 'GR-0486', 'GR-0487', 'GR-0488', 'GR-0489', 'GR
         'GR-0491', 'GR-0492', 'GR-0493', 'GR-0494', 'GR-0495', 'GR-0496', 'GR-0497',
         'GR-0499', 'GR-0500', 'GR-0501', 'GR-0502')
```

In [7]:
```
name_ID = ['Bett James_GR-0483', 'namukolo abrams_GR-0484', 'vera abutu_GR-0485', 'kwa
         'luke brant_GR-0489', 'james kenyata_GR-0490', 'ngugi tionga_GR-0491', 'okoro
        'longe jethro_GR-0495', 'florence giwa_GR-0496', 'veiva lucent_GR-0497', 'melo
        'maguel peter_0501', 'wellington zuba_0502']
subjects = ['chem', 'phy', 'bio', 'math']
```

In [8]:
```
#functions to hold student data (names, ID and subjects)
def name_ID(subjects):
    {name_ID:{subjects:[]}}
```

In [9]:
```
{['Bett James_GR-0483':{subjects:['chem','bio','phy','math']}],['Namukolo Abrams_GR-04
```

```
  File "<ipython-input-9-e4fe9cab4e72>", line 1
    {['Bett James_GR-0483':{subjects:['chem','bio','phy','math']}],['Namukolo Abrams_
GR-0484':{subjects:['chem','bio','math']}] }
                          ^
SyntaxError: invalid syntax
```

In [10]:
```
fullDict = dict()

def studRecCreator2(name, ID, tempDict):

    studKey = name + '_'+str(ID)
    fullDict[studKey] = tempDict
    #print(fullDict)
    return fullDict


def studRecCreator1(name, ID,subjects):
    tempDict = dict()
    tempDict['subjects']= subjects
    print(tempDict)
    resp = studRecCreator2(name, ID, tempDict)
    print(resp)
```

In [11]:
```
name = 'bett james'
ID = 'GR-0483'
subjects = ['chem', 'phy', 'bio', 'math']
studRecCreator1(name, ID, subjects)
```

```
{'subjects': ['chem', 'phy', 'bio', 'math']}
{'bett james_GR-0483': {'subjects': ['chem', 'phy', 'bio', 'math']}}
```

In [12]:
```
fullDict = dict()


def studRecCreator(name, ID,subjects):
    tempDict = dict()
    tempDict['subjects']= subjects
    #print(tempDict)
    studKey = name + '_'+str(ID)
    fullDict[studKey] = tempDict
    #print(fullDict)
    return fullDict
```

In [27]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namebett james
student IDGR-0483
list of subjects seperated by a commachem, phy, math, bio
```
Out[27]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']}}
```

In [13]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namebett james
student IDGR-0483
list of subjects seperated by a commachem, phy, math, bio
```
Out[13]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']}}
```

In [14]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namenamukolo abrams
student IDGR-0484
list of subjects seperated by a commachem, phy, math, bio
```
Out[14]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']}}
```

In [15]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namevera abutu
student IDGR-0485
list of subjects seperated by a commachem, phy, math, bio
```

Out[15]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']}}
```

In [16]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

Out[16]:
```
student namekwame doga
student IDGR-0486
list of subjects seperated by a commachem, phy, math, bio
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']}}
```

In [17]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

Out[17]:
```
student namelukeman ahmad
student IDGR-0487
list of subjects seperated by a commachem, math, phy, bio
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']}}
```

In [18]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

Out[18]:
```
student nameakin torey
student IDGR-0488
list of subjects seperated by a commachem, phy, math, bio
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']}}
```

In [19]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student nameluke brant
student IDGR-0489
list of subjects seperated by a commachem, math, phy, bio
```

Out[19]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']}}
```

In [20]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namejames kenyata
student IDGR-0490
list of subjects seperated by a commachem, phy, math, bio
```

Out[20]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']}}
```

In [21]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namengugi tionga
student IDGR-0491
list of subjects seperated by a commachem, phy, math, bio
```

Out[21]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']}}
```

In [22]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student nameokoro eze
student IDGR-0492
list of subjects seperated by a commachem, phy, bio, math
```

Out[22]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']}}
```

In [23]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

Out[23]:
```
student nameagatha chiluba
student IDGR-0493
list of subjects seperated by a commachem, phy, math, bio
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']}}
```

In [24]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

Out[24]:
```
student namemangu joseph
student IDGR-0494
list of subjects seperated by a commachem, bio, phy, math
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']}}
```

In [25]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namelonge jethro
student IDGR-0495
list of subjects seperated by a commachem, phy, bio, math
```

Out[25]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']}}
```

In [ ]:
```
```

In [26]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student nameflorence giwa
student IDGR-0496
list of subjects seperated by a commachem, phy, math, bio
```

Out[26]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']}}
```

In [27]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namevetiva lucent
student IDGR-0497
list of subjects seperated by a commachem, phy, math, bio
```

Out[27]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']},
 'vetiva lucent_GR-0497': {'subjects': ['chem, phy, math, bio']}}
```

In [28]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namemelody braimoh
student IDGR-0498
list of subjects seperated by a commachem, math, phy, bio
```

Out[28]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']},
 'vetiva lucent_GR-0497': {'subjects': ['chem, phy, math, bio']},
 'melody braimoh_GR-0498': {'subjects': ['chem, math, phy, bio']}}
```

In [29]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namevictor ihab
student IDGR-0499
list of subjects seperated by a commachem, phy, math, bio
```

Out[29]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']},
 'vetiva lucent_GR-0497': {'subjects': ['chem, phy, math, bio']},
 'melody braimoh_GR-0498': {'subjects': ['chem, math, phy, bio']},
 'victor ihab_GR-0499': {'subjects': ['chem, phy, math, bio']}}
```

In [30]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namemimi trucker
student IDGR-0500
list of subjects seperated by a commachem, phy, math, bio
```

Out[30]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']},
 'vetiva lucent_GR-0497': {'subjects': ['chem, phy, math, bio']},
 'melody braimoh_GR-0498': {'subjects': ['chem, math, phy, bio']},
 'victor ihab_GR-0499': {'subjects': ['chem, phy, math, bio']},
 'mimi trucker_GR-0500': {'subjects': ['chem, phy, math, bio']}}
```

In [31]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namemaguel peter
student IDGR-0501
list of subjects seperated by a commachem, math, phy, bio
```

Out[31]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']},
 'vetiva lucent_GR-0497': {'subjects': ['chem, phy, math, bio']},
 'melody braimoh_GR-0498': {'subjects': ['chem, math, phy, bio']},
 'victor ihab_GR-0499': {'subjects': ['chem, phy, math, bio']},
 'mimi trucker_GR-0500': {'subjects': ['chem, phy, math, bio']},
 'maguel peter_GR-0501': {'subjects': ['chem, math, phy, bio']}}
```

In [32]:
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
subjects = [subjects]
studRecCreator(name, ID,subjects)
```

```
student namewellington zuba
student IDGR-0502
list of subjects seperated by a commachem, math, phy, bio
```

Out[32]:
```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']},
 'vetiva lucent_GR-0497': {'subjects': ['chem, phy, math, bio']},
 'melody braimoh_GR-0498': {'subjects': ['chem, math, phy, bio']},
 'victor ihab_GR-0499': {'subjects': ['chem, phy, math, bio']},
 'mimi trucker_GR-0500': {'subjects': ['chem, phy, math, bio']},
 'maguel peter_GR-0501': {'subjects': ['chem, math, phy, bio']},
 'wellington zuba_GR-0502': {'subjects': ['chem, math, phy, bio']}}
```

In [33]:
```
fullDict
```

Out[33]:

```
{'bett james_GR-0483': {'subjects': ['chem, phy, math, bio']},
 'namukolo abrams_GR-0484': {'subjects': ['chem, phy, math, bio']},
 'vera abutu_GR-0485': {'subjects': ['chem, phy, math, bio']},
 'kwame doga_GR-0486': {'subjects': ['chem, phy, math, bio']},
 'lukeman ahmad_GR-0487': {'subjects': ['chem, math, phy, bio']},
 'akin torey_GR-0488': {'subjects': ['chem, phy, math, bio']},
 'luke brant_GR-0489': {'subjects': ['chem, math, phy, bio']},
 'james kenyata_GR-0490': {'subjects': ['chem, phy, math, bio']},
 'ngugi tionga_GR-0491': {'subjects': ['chem, phy, math, bio']},
 'okoro eze_GR-0492': {'subjects': ['chem, phy, bio, math']},
 'agatha chiluba_GR-0493': {'subjects': ['chem, phy, math, bio']},
 'mangu joseph_GR-0494': {'subjects': ['chem, bio, phy, math']},
 'longe jethro_GR-0495': {'subjects': ['chem, phy, bio, math']},
 'florence giwa_GR-0496': {'subjects': ['chem, phy, math, bio']},
 'vetiva lucent_GR-0497': {'subjects': ['chem, phy, math, bio']},
 'melody braimoh_GR-0498': {'subjects': ['chem, math, phy, bio']},
 'victor ihab_GR-0499': {'subjects': ['chem, phy, math, bio']},
 'mimi trucker_GR-0500': {'subjects': ['chem, phy, math, bio']},
 'maguel peter_GR-0501': {'subjects': ['chem, math, phy, bio']},
 'wellington zuba_GR-0502': {'subjects': ['chem, math, phy, bio']}}
```


In [34]:
```python
def subjectSearch1(ID, name):
    studNamesList = fullDict.keys()
    print(studNamesList)
    for KEY in studNamesList:
        print(KEY)
        nameparts = KEY.split('_')
        print(nameparts)
        if ID in nameparts or name in nameparts:
            print('we have found the student')
            subjects = fullDict[KEY].values()

            print(subjects)
```

In [35]:
```python
subjectSearch1('GR-0484', 'namukolo abrams')
```

```
dict_keys(['bett james_GR-0483', 'namukolo abrams_GR-0484', 'vera abutu_GR-0485', 'kw
ame doga_GR-0486', 'lukeman ahmad_GR-0487', 'akin torey_GR-0488', 'luke brant_GR-048
9', 'james kenyata_GR-0490', 'ngugi tionga_GR-0491', 'okoro eze_GR-0492', 'agatha chi
luba_GR-0493', 'mangu joseph_GR-0494', 'longe jethro_GR-0495', 'florence giwa_GR-049
6', 'vetiva lucent_GR-0497', 'melody braimoh_GR-0498', 'victor ihab_GR-0499', 'mimi t
rucker_GR-0500', 'maguel peter_GR-0501', 'wellington zuba_GR-0502'])
bett james_GR-0483
['bett james', 'GR-0483']
namukolo abrams_GR-0484
['namukolo abrams', 'GR-0484']
we have found the student
dict_values([['chem, phy, math, bio']])
vera abutu_GR-0485
['vera abutu', 'GR-0485']
kwame doga_GR-0486
['kwame doga', 'GR-0486']
lukeman ahmad_GR-0487
['lukeman ahmad', 'GR-0487']
akin torey_GR-0488
['akin torey', 'GR-0488']
luke brant_GR-0489
['luke brant', 'GR-0489']
james kenyata_GR-0490
['james kenyata', 'GR-0490']
ngugi tionga_GR-0491
['ngugi tionga', 'GR-0491']
okoro eze_GR-0492
['okoro eze', 'GR-0492']
agatha chiluba_GR-0493
['agatha chiluba', 'GR-0493']
mangu joseph_GR-0494
['mangu joseph', 'GR-0494']
longe jethro_GR-0495
['longe jethro', 'GR-0495']
florence giwa_GR-0496
['florence giwa', 'GR-0496']
vetiva lucent_GR-0497
['vetiva lucent', 'GR-0497']
melody braimoh_GR-0498
['melody braimoh', 'GR-0498']
victor ihab_GR-0499
['victor ihab', 'GR-0499']
mimi trucker_GR-0500
['mimi trucker', 'GR-0500']
maguel peter_GR-0501
['maguel peter', 'GR-0501']
wellington zuba_GR-0502
['wellington zuba', 'GR-0502']
```

In [36]:

```
if  subs = subjects.values():
    subsTaken = (f'subjects taken by {name} with ID No: {ID} are {subs}')
else:
    subsTaken = (f'records for {name} with ID No: {ID} not found')
    return subsTaken
```

```
  File "<ipython-input-36-6d215fc997e6>", line 1
    if  subs = subjects.values():
            ^
SyntaxError: invalid syntax
```

In [39]:
```
fullDict['namukolo abrams_GR-0484']
```

Out[39]:
```
{'subjects': ['chem, phy, math, bio']}
```

In [40]:
```
subjs = fullDict['bett james_GR-0483']

subjs

list(subjs.values())
```

Out[40]:
```
[['chem, phy, math, bio']]
```

In [41]:
```
subsList = fullDict['namukolo abrams_GR-0484'].values()

list(subsList)
```

Out[41]:
```
[['chem, phy, math, bio']]
```

In [42]:
```
def subjectSearch2(ID, name):
    studNamesList = fullDict.keys()
    #print(studNamesList)
    for names in studNamesList:
        #print(names)
        nameparts = names.split('_')
        #print(nameparts)
        if ID in nameparts or name in nameparts:
            print('we have found the student')
            subjects = fullDict[names] # obtaining the subjects dictionary for the fou

            print(subjects)
            print(subjects.values()) # obtaining the keys (list of subjects) from the
            print(list(subjects.values())) # converting the values to a list
```

In [43]:
```
subjectSearch2('GR-0486', 'kwame doga')
```

```
we have found the student
{'subjects': ['chem, phy, math, bio']}
dict_values([['chem, phy, math, bio']])
[['chem, phy, math, bio']]
```

In [227…
```
#What we need from Jerome:
 #   1. student name or ID
  #  2. subject for which records are to be entered
   # 3. assessment type for which score will be entered
    # 4. score
```

In [219…
```
# Design of what function should do
# {nameID:{'chem':{'Quiz':56, 'HW':89, 'ATTND':76, 'Exam':43},
 #       'phy':{'Quiz':77, 'HW':59, 'ATTND':56, 'Exam':83},
    #     'math':{'Quiz':56, 'HW':23, 'ATTND':76, 'Exam':65},
    #     'bio':{'Quiz':44, 'HW':89, 'ATTND':65, 'Exam':43}
     #   }}
```

In [228…
```
# for one subject
def scoreAggregator(name, ID, subject, assessTypeList, scoreList):
    tempDict = dict()

    #myDict = dict(zip(keys, values)) structure of code to convert two lists into a di
```

```
    resDict= dict(zip(assessTypeList,scoreList ))
    tempDict[subject] = resDict



    print(tempDict)
```

In [229…
```
studSubsDict = dict()
perfList = []
def scoreAggregator(name, ID, subject, assessTypeList, scoreList):
    tempDict = dict()

    #myDict = dict(zip(keys, values)) structure of code to convert two lists into a di


    resDict= dict(zip(assessTypeList,scoreList ))
    tempDict[subject] = resDict
    perfList.append(tempDict)

    print(perfList)
```

In [190…
```
def avgScore(name, ID, subject, assessTypeList, scoreList):
    avgScore[name] = (scoreList)/4
    GPA = (avgScore/100)*5
```

In [214…

In [230…
```
name = 'bett james'
ID = 'GR-0483'
subject = 'chem'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [127,135,17,46]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'chem': {'Quiz': 127, 'HW': 135, 'ATTND': 17, 'Exam': 46}}]
```

In [ ]:

In [231…
```
name = 'bett james'
ID = 'GR-0483'
subject = 'bio'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [184,186,58,97]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'chem': {'Quiz': 127, 'HW': 135, 'ATTND': 17, 'Exam': 46}}, {'bio': {'Quiz': 184,
'HW': 186, 'ATTND': 58, 'Exam': 97}}]
```

In [232…
```
name = 'bett james'
ID = 'GR-0483'
subject = 'phy'
```

```
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [52,142,29,73]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'chem': {'Quiz': 127, 'HW': 135, 'ATTND': 17, 'Exam': 46}}, {'bio': {'Quiz': 184,
'HW': 186, 'ATTND': 58, 'Exam': 97}}, {'phy': {'Quiz': 52, 'HW': 142, 'ATTND': 29, 'E
xam': 73}}]
```

In [233…

```
def perfAggregator(name, ID,perfList):
    name_ID = name + '_'+ str(ID)
    studSubsDict[name_ID]=perfList
```

In [234…

```
name = 'bett james'
ID = 'GR-0483'
subject = 'math'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [133,97,34,45]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'chem': {'Quiz': 127, 'HW': 135, 'ATTND': 17, 'Exam': 46}}, {'bio': {'Quiz': 184,
'HW': 186, 'ATTND': 58, 'Exam': 97}}, {'phy': {'Quiz': 52, 'HW': 142, 'ATTND': 29, 'E
xam': 73}}, {'math': {'Quiz': 133, 'HW': 97, 'ATTND': 34, 'Exam': 45}}]
```

In [235…

```
perfAggregator(name, ID,perfList)
print(studSubsDict)
```

```
{'bett james_GR-0483': [{'chem': {'Quiz': 127, 'HW': 135, 'ATTND': 17, 'Exam': 46}},
{'bio': {'Quiz': 184, 'HW': 186, 'ATTND': 58, 'Exam': 97}}, {'phy': {'Quiz': 52, 'H
W': 142, 'ATTND': 29, 'Exam': 73}}, {'math': {'Quiz': 133, 'HW': 97, 'ATTND': 34, 'Ex
am': 45}}]}
```

In [236…

```
name = 'bett james'
ID = 'GR-0483'
subject = 'math'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [133,97,34,45]

flag = 2 # when set to 1, code aggregates subjectwise score. when set to 2 it compust

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoreList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    studSubsDict[name_ID]=perfList



print(studSubsDict)
```

```
{'bett james_GR-0483': [{'chem': {'Quiz': 127, 'HW': 135, 'ATTND': 17, 'Exam': 46}},
{'bio': {'Quiz': 184, 'HW': 186, 'ATTND': 58, 'Exam': 97}}, {'phy': {'Quiz': 52, 'H
W': 142, 'ATTND': 29, 'Exam': 73}}, {'math': {'Quiz': 133, 'HW': 97, 'ATTND': 34, 'Ex
am': 45}}]}
```

In [ ]:

In [108…
```
name = input('student name')
ID = str(input('student ID'))
subjects = input('list of subjects seperated by a comma')
assessType = input('assessment type')
scoresList = input('scoresList')
subjects = [subjects]
```

```
student namenamukolo abrams
student IDGR-0484
list of subjects seperated by a commachem, phy, bio, math
assessment typeQuiz
scoresList141, 21, 177, 136
```

In [237…
```
def perfAggregator(name, ID,perfList):
    name_ID = name + '_'+ str(ID)
    studSubsDict[name_ID]=perfList
    return studSubsDict


def scoreAggregator(name, ID, subject, assessTypeList, scoreList):
    #perfList = []

    tempDict = dict()
    resDict= dict(zip(assessTypeList,scoreList ))
    tempDict[subject] = resDict

    #print(tempDict)

    return tempDict
```

In [238…
```
studSubsDict = dict()
perfList = []
def scoreAggregator(name, ID, subject, assessTypeList, scoreList):
    tempDict = dict()

    #myDict = dict(zip(keys, values)) structure of code to convert two lists into a di


    resDict= dict(zip(assessTypeList,scoreList ))
    tempDict[subject] = resDict
    perfList.append(tempDict)

    print(perfList)
```

In [239…
```
name = 'namukolo abrams'
ID = 'GR-0484'
subject = 'math'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [136,180,48,31]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'math': {'Quiz': 136, 'HW': 180, 'ATTND': 48, 'Exam': 31}}]
```

In [ ]:

In [240…
```
name = 'namukolo abrams'
ID = 'GR-0484'
subject = 'phy'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [21,12,43,31]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'math': {'Quiz': 136, 'HW': 180, 'ATTND': 48, 'Exam': 31}}, {'phy': {'Quiz': 21, 'H
W': 12, 'ATTND': 43, 'Exam': 31}}]
```

In [241…
```
name = 'namukolo abrams'
ID = 'GR-0484'
subject = 'bio'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [177,170,57,41]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'math': {'Quiz': 136, 'HW': 180, 'ATTND': 48, 'Exam': 31}}, {'phy': {'Quiz': 21, 'H
W': 12, 'ATTND': 43, 'Exam': 31}}, {'bio': {'Quiz': 177, 'HW': 170, 'ATTND': 57, 'Exa
m': 41}}]
```

In [242…
```
name = 'namukolo abrams'
ID = 'GR-0484'
subject = 'chem'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [141,82,95,52]

scoreAggregator(name, ID, subject, assessTypeList, scoreList)
```

```
[{'math': {'Quiz': 136, 'HW': 180, 'ATTND': 48, 'Exam': 31}}, {'phy': {'Quiz': 21, 'H
W': 12, 'ATTND': 43, 'Exam': 31}}, {'bio': {'Quiz': 177, 'HW': 170, 'ATTND': 57, 'Exa
m': 41}}, {'chem': {'Quiz': 141, 'HW': 82, 'ATTND': 95, 'Exam': 52}}]
```

In [243…
```
perfAggregator(name, ID,perfList)
print(studSubsDict)
```

```
{'namukolo abrams_GR-0484': [{'math': {'Quiz': 136, 'HW': 180, 'ATTND': 48, 'Exam': 3
1}}, {'phy': {'Quiz': 21, 'HW': 12, 'ATTND': 43, 'Exam': 31}}, {'bio': {'Quiz': 177,
'HW': 170, 'ATTND': 57, 'Exam': 41}}, {'chem': {'Quiz': 141, 'HW': 82, 'ATTND': 95,
'Exam': 52}}]}
```

In [244…
```
name = 'namukolo abrams'
ID = 'GR-0484'
subject = 'chem'
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam']
scoreList = [141,82,95,52]

flag = 2 # when set to 1, code aggregates subjectwise score. when set to 2 it compust

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoreList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
```

```
    name_ID = name + '_'+ str(ID)
    studSubsDict[name_ID]=perfList



print(studSubsDict)
```

```
{'namukolo abrams_GR-0484': [{'math': {'Quiz': 136, 'HW': 180, 'ATTND': 48, 'Exam': 3
1}}, {'phy': {'Quiz': 21, 'HW': 12, 'ATTND': 43, 'Exam': 31}}, {'bio': {'Quiz': 177,
'HW': 170, 'ATTND': 57, 'Exam': 41}}, {'chem': {'Quiz': 141, 'HW': 82, 'ATTND': 95,
'Exam': 52}}]}
```

In [245…
```
studSubsDict.keys()
```

Out[245]:
```
dict_keys(['namukolo abrams_GR-0484'])
```

In [246…
```
studSubsDict.values()
```

Out[246]:
```
dict_values([[{'math': {'Quiz': 136, 'HW': 180, 'ATTND': 48, 'Exam': 31}}, {'phy':
{'Quiz': 21, 'HW': 12, 'ATTND': 43, 'Exam': 31}}, {'bio': {'Quiz': 177, 'HW': 170, 'A
TTND': 57, 'Exam': 41}}, {'chem': {'Quiz': 141, 'HW': 82, 'ATTND': 95, 'Exam': 5
2}}]])
```

In [ ]:
```

```

In [131…
```
name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = input('assessment type List')
scoresList = input('scoresList')
flag = 1

scoreAggregator(name, ID, subject, assessTypeList, scoreList, flag)
```

```
student namenamukolo abrams
student IDGR-0484
subjectchem
assessment type ListQuiz, HW, ATTND, Exam
scoresList141, 82, 95, 52
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-131-918cdd5ccb16> in <module>
      6 flag = 1
      7
----> 8 scoreAggregator(name, ID, subject, assessTypeList, scoreList, flag)

TypeError: scoreAggregator() takes 5 positional arguments but 6 were given
```

In [247…
```
def perfAggregator(name, ID,perfList):
    name_ID = name + '_'+ str(ID)
    studSubsDict[name_ID]=perfList
    return studSubsDict


def scoreAggregator(name, ID, subject, assessTypeList, scoreList, flag):
    #perfList = []

    tempDict = dict()
    resDict= dict(zip(assessTypeList,scoreList))
    tempDict[subject] = resDict
```

```
    perfList.append(tempDict)


    #print(tempDict)

    return tempDict
```

In [248…
```
scoreRec = studSubsDict['namukolo abrams_GR-0484']
for items in list(scoreRec):
    if list(items.keys())[0] == 'Phy':
        print(items.values())
```

In [249…
```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = input('assessment type List')
scoresList = input('scoresList')
```

```
student namekwame doga
student IDGR-0486
subjectchem
assessment type ListQuiz, HW, ATTND, Exam
scoresList26, 166, 75, 73
```

In [250…
```
assessTypeList
```

Out[250]:
```
'Quiz, HW, ATTND, Exam'
```

In [251…
```
assessTypeList[0]
```

Out[251]:
```
'Q'
```

In [252…
```
aTypeList = assessTypeList.split(',')
```

In [253…
```
aTypeList
```

Out[253]:
```
['Quiz', ' HW', ' ATTND', ' Exam']
```

In [254…
```
scoresList
```

Out[254]:
```
'26, 166, 75, 73'
```

In [255…
```
sList = scoresList.split(',')

int(sList[0])
```

Out[255]:
```
26
```

In [292…
```
import statistics

perfList = []
studSubsDict = dict()
```

```
def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
```

```
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [293…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namebett james
student IDGR-0483
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam127, 135, 17, 46
set flag1
[{'chem': {'Quiz': 127.0, 'HW': 135.0, 'ATTND': 17.0, 'Exam': 46.0}}]
{}
```

In [294…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namebett james
student IDGR-0483
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam52, 142, 29, 73
set flag1
[{'chem': {'Quiz': 127.0, 'HW': 135.0, 'ATTND': 17.0, 'Exam': 46.0}}, {'phy': {'Qui
z': 52.0, 'HW': 142.0, 'ATTND': 29.0, 'Exam': 73.0}}]
{}
```

In [295…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
```

```
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namebett james
student IDGR-0483
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam184, 186, 58, 97
set flag1
[{'chem': {'Quiz': 127.0, 'HW': 135.0, 'ATTND': 17.0, 'Exam': 46.0}}, {'phy': {'Qui
z': 52.0, 'HW': 142.0, 'ATTND': 29.0, 'Exam': 73.0}}, {'bio': {'Quiz': 184.0, 'HW': 1
86.0, 'ATTND': 58.0, 'Exam': 97.0}}]
{}
```

In [296…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
```

```
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namebett james
student IDGR-0483
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam133, 97, 34, 45
set flag1
[{'chem': {'Quiz': 127.0, 'HW': 135.0, 'ATTND': 17.0, 'Exam': 46.0}}, {'phy': {'Qui
z': 52.0, 'HW': 142.0, 'ATTND': 29.0, 'Exam': 73.0}}, {'bio': {'Quiz': 184.0, 'HW': 1
86.0, 'ATTND': 58.0, 'Exam': 97.0}}, {'math': {'Quiz': 133.0, 'HW': 97.0, 'ATTND': 3
4.0, 'Exam': 45.0}}]
{}
```

In [297…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []
```

```
print(perfList)
print(studSubsDict)
```

```
student namebett james
student IDGR-0483
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam133, 97, 34, 45
set flag2
52.615526315789474
54.436052631578946
92.69684210526316
52.25789473684211
this is the mean score of the student: 63.0
([52.615526315789474, 54.436052631578946, 92.69684210526316, 52.25789473684211], 63.
0, 'D', 3.15, 'Pass')
[]
{'bett james_GR-0483': [{'chem': {'Quiz': 127.0, 'HW': 135.0, 'ATTND': 17.0, 'Exam':
46.0}}, {'phy': {'Quiz': 52.0, 'HW': 142.0, 'ATTND': 29.0, 'Exam': 73.0}}, {'bio':
{'Quiz': 184.0, 'HW': 186.0, 'ATTND': 58.0, 'Exam': 97.0}}, {'math': {'Quiz': 133.0,
'HW': 97.0, 'ATTND': 34.0, 'Exam': 45.0}}, {'Subject Scores': [52.615526315789474, 5
4.436052631578946, 92.69684210526316, 52.25789473684211], 'Overall Score': 63.0, 'Gra
de': 'D', 'GPA': 3.15, 'Status': 'Pass'}]}
```

In [ ]:

In [ ]:

In [ ]:

In [258…

```
#Quiz, HW, ATTND, Exam
```

In [298…

```
import statistics
def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12
```

```
            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 80:
        grade = 'A'
    elif Score > 70:
        grade = 'B'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [270…
```
{'bett james_GR-0483': [{'chem': {'Quiz': 127, 'HW': 135, 'ATTND': 17, 'Exam': 46}}, {
```

Out[270]:
```
{'bett james_GR-0483': [{'chem': {'Quiz': 127,
    'HW': 135,
    'ATTND': 17,
    'Exam': 46}},
  {'bio': {'Quiz': 184, 'HW': 186, 'ATTND': 58, 'Exam': 97}},
  {'phy': {'Quiz': 52, 'HW': 142, 'ATTND': 29, 'Exam': 73}},
  {'math': {'Quiz': 133, 'HW': 97, 'ATTND': 34, 'Exam': 45}}]}
```

In [ ]:
```
res = avgScoreComput(name_ID, perfList)
```

In [299…
```
res = avgScoreComput('bett james_GR-0483', studSubsDict['bett james_GR-0483'])
```

```
52.615526315789474
54.436052631578946
92.69684210526316
52.25789473684211
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-299-71c1fdddb949> in <module>
----> 1 res = avgScoreComput('bett james_GR-0483', studSubsDict['bett james_GR-0483']
)

<ipython-input-298-fda39a247b13> in avgScoreComput(studname, sDictKeys)
     17         #print(subjScores.keys())
     18         #print(subjScores['Quiz'])
---> 19         for sub in list(subjScores.keys()):
     20             #print(sub)
     21             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [300…
```
res
```

Out[300]:
```
([52.615526315789474,
  54.436052631578946,
  92.69684210526316,
  52.25789473684211],
 63.0,
 'D',
 3.15,
 'Pass')
```

In [301…
```
studSubsDict['bett james_GR-0483']
```

Out[301]:
```
[{'chem': {'Quiz': 127.0, 'HW': 135.0, 'ATTND': 17.0, 'Exam': 46.0}},
 {'phy': {'Quiz': 52.0, 'HW': 142.0, 'ATTND': 29.0, 'Exam': 73.0}},
 {'bio': {'Quiz': 184.0, 'HW': 186.0, 'ATTND': 58.0, 'Exam': 97.0}},
 {'math': {'Quiz': 133.0, 'HW': 97.0, 'ATTND': 34.0, 'Exam': 45.0}},
 {'Subject Scores': [52.615526315789474,
   54.436052631578946,
   92.69684210526316,
   52.25789473684211],
  'Overall Score': 63.0,
  'Grade': 'D',
  'GPA': 3.15,
  'Status': 'Pass'}]
```

In [303…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
```

```
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [304…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec
```

```
if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namenamukolo abrams
student IDGR-0484
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam141, 82, 95, 52
set flag1
[{'chem': {'Quiz': 141.0, 'HW': 82.0, 'ATTND': 95.0, 'Exam': 52.0}}]
{}
```

In [305…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
```

```
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namenamukolo abrams
student IDGR-0484
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam21, 12, 43, 31
set flag1
[{'chem': {'Quiz': 141.0, 'HW': 82.0, 'ATTND': 95.0, 'Exam': 52.0}}, {'phy': {'Quiz':
21.0, 'HW': 12.0, 'ATTND': 43.0, 'Exam': 31.0}}]
{}
```

In [306…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namenamukolo abrams
student IDGR-0484
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam177, 170, 57, 41
set flag1
[{'chem': {'Quiz': 141.0, 'HW': 82.0, 'ATTND': 95.0, 'Exam': 52.0}}, {'phy': {'Quiz':
21.0, 'HW': 12.0, 'ATTND': 43.0, 'Exam': 31.0}}, {'bio': {'Quiz': 177.0, 'HW': 170.0,
'ATTND': 57.0, 'Exam': 41.0}}]
{}
```

In [307…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namenamukolo abrams
student IDGR-0484
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam136, 180, 48, 31
set flag1
[{'chem': {'Quiz': 141.0, 'HW': 82.0, 'ATTND': 95.0, 'Exam': 52.0}}, {'phy': {'Quiz':
21.0, 'HW': 12.0, 'ATTND': 43.0, 'Exam': 31.0}}, {'bio': {'Quiz': 177.0, 'HW': 170.0,
'ATTND': 57.0, 'Exam': 41.0}}, {'math': {'Quiz': 136.0, 'HW': 180.0, 'ATTND': 48.0,
'Exam': 31.0}}]
{}
```

In [308…

```
# automating inputs

name = input('student name')
```

```
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namenamukolo abrams
student IDGR-0484
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam136, 180, 48, 31
set flag2
62.97184210526316
22.96815789473684
66.12342105263157
55.014210526315786
this is the mean score of the student: 51.77
([62.97184210526316, 22.96815789473684, 66.12342105263157, 55.014210526315786], 51.7
7, 'E', 2.5885000000000002, 'Retake')
[]
{'namukolo abrams_GR-0484': [{'chem': {'Quiz': 141.0, 'HW': 82.0, 'ATTND': 95.0, 'Exa
m': 52.0}}, {'phy': {'Quiz': 21.0, 'HW': 12.0, 'ATTND': 43.0, 'Exam': 31.0}}, {'bio':
{'Quiz': 177.0, 'HW': 170.0, 'ATTND': 57.0, 'Exam': 41.0}}, {'math': {'Quiz': 136.0,
'HW': 180.0, 'ATTND': 48.0, 'Exam': 31.0}}, {'Subject Scores': [62.97184210526316, 2
2.96815789473684, 66.12342105263157, 55.014210526315786], 'Overall Score': 51.77, 'Gr
ade': 'E', 'GPA': 2.5885000000000002, 'Status': 'Retake'}]}
```

In [309…

```
res = avgScoreComput('namukolo abrams_GR-0484', studSubsDict['namukolo abrams_GR-0484'
```

```
62.97184210526316
22.96815789473684
66.12342105263157
55.014210526315786
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-309-20a25c72bf14> in <module>
----> 1 res = avgScoreComput('namukolo abrams_GR-0484', studSubsDict['namukolo abrams
_GR-0484'])

<ipython-input-303-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [310…
```
res
```

Out[310]:
```
([62.97184210526316, 22.96815789473684, 66.12342105263157, 55.014210526315786],
 51.77,
 'E',
 2.5885000000000002,
 'Retake')
```

In [311…
```
studSubsDict['namukolo abrams_GR-0484']
```

Out[311]:
```
[{'chem': {'Quiz': 141.0, 'HW': 82.0, 'ATTND': 95.0, 'Exam': 52.0}},
 {'phy': {'Quiz': 21.0, 'HW': 12.0, 'ATTND': 43.0, 'Exam': 31.0}},
 {'bio': {'Quiz': 177.0, 'HW': 170.0, 'ATTND': 57.0, 'Exam': 41.0}},
 {'math': {'Quiz': 136.0, 'HW': 180.0, 'ATTND': 48.0, 'Exam': 31.0}},
 {'Subject Scores': [62.97184210526316,
   22.96815789473684,
   66.12342105263157,
   55.014210526315786],
  'Overall Score': 51.77,
  'Grade': 'E',
  'GPA': 2.5885000000000002,
  'Status': 'Retake'}]
```

In [332…
```
studSubsDict['namukolo abrams_GR-0484']
```

Out[332]:
```
[{'chem': {'Quiz': 141.0, 'HW': 82.0, 'ATTND': 95.0, 'Exam': 52.0}},
 {'phy': {'Quiz': 21.0, 'HW': 12.0, 'ATTND': 43.0, 'Exam': 31.0}},
 {'bio': {'Quiz': 177.0, 'HW': 170.0, 'ATTND': 57.0, 'Exam': 41.0}},
 {'math': {'Quiz': 136.0, 'HW': 180.0, 'ATTND': 48.0, 'Exam': 31.0}},
 {'Subject Scores': [62.97184210526316,
   22.96815789473684,
   66.12342105263157,
   55.014210526315786],
  'Overall Score': 51.77,
  'Grade': 'E',
  'GPA': 2.5885000000000002,
  'Status': 'Retake'}]
```

In [333…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):
```

```
    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
```

```
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [334…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelukeman ahmad
student IDGR-0487
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam29, 180, 38, 80
set flag1
[{'chem': {'Quiz': 29.0, 'HW': 180.0, 'ATTND': 38.0, 'Exam': 80.0}}]
{}
```

In [335…
```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelukeman ahmad
student IDGR-0487
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam177, 11, 38, 51
set flag1
[{'chem': {'Quiz': 29.0, 'HW': 180.0, 'ATTND': 38.0, 'Exam': 80.0}}, {'phy': {'Quiz':
177.0, 'HW': 11.0, 'ATTND': 38.0, 'Exam': 51.0}}]
{}
```

In [336…
```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
```

```
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelukeman ahmad
student IDGR-0487
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam21, 123, 47, 90
set flag1
[{'chem': {'Quiz': 29.0, 'HW': 180.0, 'ATTND': 38.0, 'Exam': 80.0}}, {'phy': {'Quiz':
177.0, 'HW': 11.0, 'ATTND': 38.0, 'Exam': 51.0}}, {'bio': {'Quiz': 21.0, 'HW': 123.0,
'ATTND': 47.0, 'Exam': 90.0}}]
{}
```

In [337…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)
```

```
    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelukeman ahmad
student IDGR-0487
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam50, 18, 34, 68
set flag1
[{'chem': {'Quiz': 29.0, 'HW': 180.0, 'ATTND': 38.0, 'Exam': 80.0}}, {'phy': {'Quiz':
177.0, 'HW': 11.0, 'ATTND': 38.0, 'Exam': 51.0}}, {'bio': {'Quiz': 21.0, 'HW': 123.0,
'ATTND': 47.0, 'Exam': 90.0}}, {'math': {'Quiz': 50.0, 'HW': 18.0, 'ATTND': 34.0, 'Ex
am': 68.0}}]
{}
```

In [338…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelukeman ahmad
student IDGR-0487
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam50, 18, 34, 68
set flag2
57.939473684210526
55.495789473684205
57.60131578947369
42.805789473684214
this is the mean score of the student: 53.46
([57.939473684210526, 55.495789473684205, 57.60131578947369, 42.805789473684214], 53.
46, 'E', 2.673, 'Retake')
[]
{'lukeman ahmad_GR-0487': [{'chem': {'Quiz': 29.0, 'HW': 180.0, 'ATTND': 38.0, 'Exa
m': 80.0}}, {'phy': {'Quiz': 177.0, 'HW': 11.0, 'ATTND': 38.0, 'Exam': 51.0}}, {'bi
o': {'Quiz': 21.0, 'HW': 123.0, 'ATTND': 47.0, 'Exam': 90.0}}, {'math': {'Quiz': 50.
0, 'HW': 18.0, 'ATTND': 34.0, 'Exam': 68.0}}, {'Subject Scores': [57.939473684210526,
55.495789473684205, 57.60131578947369, 42.805789473684214], 'Overall Score': 53.46,
'Grade': 'E', 'GPA': 2.673, 'Status': 'Retake'}]}
```

In [339…
```
res = avgScoreComput('lukeman ahmad_GR-0487', studSubsDict['lukeman ahmad_GR-0487'])
```

```
57.939473684210526
55.495789473684205
57.60131578947369
42.805789473684214
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-339-f030ad2f3393> in <module>
----> 1 res = avgScoreComput('lukeman ahmad_GR-0487', studSubsDict['lukeman ahmad_GR-
0487'])

<ipython-input-333-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [340…
```
res
```

Out[340]:
```
([57.939473684210526,
  55.495789473684205,
  57.60131578947369,
  42.805789473684214],
 53.46,
 'E',
 2.673,
 'Retake')
```

In [341…
```
studSubsDict['lukeman ahmad_GR-0487']
```

Out[341]:
```
[{'chem': {'Quiz': 29.0, 'HW': 180.0, 'ATTND': 38.0, 'Exam': 80.0}},
 {'phy': {'Quiz': 177.0, 'HW': 11.0, 'ATTND': 38.0, 'Exam': 51.0}},
 {'bio': {'Quiz': 21.0, 'HW': 123.0, 'ATTND': 47.0, 'Exam': 90.0}},
 {'math': {'Quiz': 50.0, 'HW': 18.0, 'ATTND': 34.0, 'Exam': 68.0}},
 {'Subject Scores': [57.939473684210526,
   55.495789473684205,
   57.60131578947369,
   42.805789473684214],
  'Overall Score': 53.46,
  'Grade': 'E',
  'GPA': 2.673,
  'Status': 'Retake'}]
```

In [ ]:

In [342…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
```

```
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [343…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
```

```
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevera abutu
student IDGR-0485
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam51, 102, 96, 24
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 102.0, 'ATTND': 96.0, 'Exam': 24.0}}]
{}
```

In [344…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevera abutu
student IDGR-0485
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam61, 105, 83, 98
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 102.0, 'ATTND': 96.0, 'Exam': 24.0}}, {'phy': {'Quiz':
61.0, 'HW': 105.0, 'ATTND': 83.0, 'Exam': 98.0}}]
{}
```

In [345…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevera abutu
student IDGR-0485
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam112, 55, 75, 51
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 102.0, 'ATTND': 96.0, 'Exam': 24.0}}, {'phy': {'Quiz':
61.0, 'HW': 105.0, 'ATTND': 83.0, 'Exam': 98.0}}, {'bio': {'Quiz': 112.0, 'HW': 55.0,
'ATTND': 75.0, 'Exam': 51.0}}]
{}
```

In [346…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
```

```
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevera abutu
student IDGR-0485
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam74, 50, 93, 94
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 102.0, 'ATTND': 96.0, 'Exam': 24.0}}, {'phy': {'Quiz':
61.0, 'HW': 105.0, 'ATTND': 83.0, 'Exam': 98.0}}, {'bio': {'Quiz': 112.0, 'HW': 55.0,
'ATTND': 75.0, 'Exam': 51.0}}, {'math': {'Quiz': 74.0, 'HW': 50.0, 'ATTND': 93.0, 'Ex
am': 94.0}}]
{}
```

In [347…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevera abutu
student IDGR-0485
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam74, 50, 93, 94
set flag2
38.42526315789473
70.43605263157895
53.33131578947368
67.67657894736841
this is the mean score of the student: 57.47
([38.42526315789473, 70.43605263157895, 53.33131578947368, 67.67657894736841], 57.47,
'D', 2.8735, 'Pass')
[]
{'vera abutu_GR-0485': [{'chem': {'Quiz': 51.0, 'HW': 102.0, 'ATTND': 96.0, 'Exam': 2
4.0}}, {'phy': {'Quiz': 61.0, 'HW': 105.0, 'ATTND': 83.0, 'Exam': 98.0}}, {'bio': {'Q
uiz': 112.0, 'HW': 55.0, 'ATTND': 75.0, 'Exam': 51.0}}, {'math': {'Quiz': 74.0, 'HW':
50.0, 'ATTND': 93.0, 'Exam': 94.0}}, {'Subject Scores': [38.42526315789473, 70.436052
63157895, 53.33131578947368, 67.67657894736841], 'Overall Score': 57.47, 'Grade':
'D', 'GPA': 2.8735, 'Status': 'Pass'}]}
```

In [348…

```
res = avgScoreComput('vera abutu_GR-0485', studSubsDict['vera abutu_GR-0485'])
```

```
38.42526315789473
70.43605263157895
53.33131578947368
67.67657894736841
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-348-e5674e57db96> in <module>
----> 1 res = avgScoreComput('vera abutu_GR-0485', studSubsDict['vera abutu_GR-0485']
)

<ipython-input-342-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [349…

```
res
```

Out[349]:
```
([38.42526315789473, 70.43605263157895, 53.33131578947368, 67.67657894736841],
 57.47,
 'D',
 2.8735,
 'Pass')
```

In [350…
```
studSubsDict['vera abutu_GR-0485']
```

Out[350]:
```
[{'chem': {'Quiz': 51.0, 'HW': 102.0, 'ATTND': 96.0, 'Exam': 24.0}},
 {'phy': {'Quiz': 61.0, 'HW': 105.0, 'ATTND': 83.0, 'Exam': 98.0}},
 {'bio': {'Quiz': 112.0, 'HW': 55.0, 'ATTND': 75.0, 'Exam': 51.0}},
 {'math': {'Quiz': 74.0, 'HW': 50.0, 'ATTND': 93.0, 'Exam': 94.0}},
 {'Subject Scores': [38.42526315789473,
   70.43605263157895,
   53.33131578947368,
   67.67657894736841],
  'Overall Score': 57.47,
  'Grade': 'D',
  'GPA': 2.8735,
  'Status': 'Pass'}]
```

In [360…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
```

```
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [352…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
```

```
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namekwame doga
student IDGR-0486
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam26, 166, 75, 73
set flag1
[{'chem': {'Quiz': 26.0, 'HW': 166.0, 'ATTND': 75.0, 'Exam': 73.0}}]
{}
```

In [353…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
```

```
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namekwame doga
student IDGR-0486
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam62, 59, 94, 65
set flag1
[{'chem': {'Quiz': 26.0, 'HW': 166.0, 'ATTND': 75.0, 'Exam': 73.0}}, {'phy': {'Quiz':
62.0, 'HW': 59.0, 'ATTND': 94.0, 'Exam': 65.0}}]
{}
```

In [354…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevera abutu
student IDGR-0486
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam133, 108, 52, 68
set flag1
[{'chem': {'Quiz': 26.0, 'HW': 166.0, 'ATTND': 75.0, 'Exam': 73.0}}, {'phy': {'Quiz':
62.0, 'HW': 59.0, 'ATTND': 94.0, 'Exam': 65.0}}, {'bio': {'Quiz': 133.0, 'HW': 108.0,
'ATTND': 52.0, 'Exam': 68.0}}]
{}
```

In [355…
```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namekwame doga
student IDGR-0486
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam86, 181, 66, 67
set flag1
[{'chem': {'Quiz': 26.0, 'HW': 166.0, 'ATTND': 75.0, 'Exam': 73.0}}, {'phy': {'Quiz':
62.0, 'HW': 59.0, 'ATTND': 94.0, 'Exam': 65.0}}, {'bio': {'Quiz': 133.0, 'HW': 108.0,
'ATTND': 52.0, 'Exam': 68.0}}, {'math': {'Quiz': 86.0, 'HW': 181.0, 'ATTND': 66.0, 'E
xam': 67.0}}]
{}
```

In [356…
```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
```

```
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namekwame doga
student IDGR-0486
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam86, 181, 66, 67
set flag2
57.97552631578948
54.14736842105263
65.26631578947368
64.92842105263158
this is the mean score of the student: 60.58
([57.97552631578948, 54.14736842105263, 65.26631578947368, 64.92842105263158], 60.58,
'D', 3.029, 'Pass')
[]
{'kwame doga_GR-0486': [{'chem': {'Quiz': 26.0, 'HW': 166.0, 'ATTND': 75.0, 'Exam': 7
3.0}}, {'phy': {'Quiz': 62.0, 'HW': 59.0, 'ATTND': 94.0, 'Exam': 65.0}}, {'bio': {'Qu
iz': 133.0, 'HW': 108.0, 'ATTND': 52.0, 'Exam': 68.0}}, {'math': {'Quiz': 86.0, 'HW':
181.0, 'ATTND': 66.0, 'Exam': 67.0}}, {'Subject Scores': [57.97552631578948, 54.14736
842105263, 65.26631578947368, 64.92842105263158], 'Overall Score': 60.58, 'Grade':
'D', 'GPA': 3.029, 'Status': 'Pass'}]}
```

In [398…

```
{'kwame doga_GR-0486': [{'chem': {'Quiz': 26.0, 'HW': 166.0, 'ATTND': 75.0, 'Exam': 73
```

Out[398]:

```
{'kwame doga_GR-0486': [{'chem': {'Quiz': 26.0,
    'HW': 166.0,
    'ATTND': 75.0,
    'Exam': 73.0}},
  {'phy': {'Quiz': 62.0, 'HW': 59.0, 'ATTND': 94.0, 'Exam': 65.0}},
  {'bio': {'Quiz': 133.0, 'HW': 108.0, 'ATTND': 52.0, 'Exam': 68.0}},
  {'math': {'Quiz': 86.0, 'HW': 181.0, 'ATTND': 66.0, 'Exam': 67.0}},
  {'Subject Scores': [57.97552631578948,
    54.14736842105263,
    65.26631578947368,
    64.92842105263158],
   'Overall Score': 60.58,
   'Grade': 'D',
   'GPA': 3.029,
   'Status': 'Pass'}]}
```

In [ ]:
```
res = avgScoreComput('kwame doga_GR-0486', studSubsDict['kwame doga_GR-0486'])
```

In [358…
```
res
```

Out[358]:
```
([57.97552631578948, 54.14736842105263, 65.26631578947368, 64.92842105263158],
 60.58,
 'D',
 3.029,
 'Pass')
```

In [ ]:
```
studSubsDict['kwame doga_GR-0486']
```

In [ ]:
```

```

In [379…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15
```

```
            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [380…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
```

```
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameakin torey
student IDGR-0488
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam99, 125, 26, 83
set flag1
[{'chem': {'Quiz': 99.0, 'HW': 125.0, 'ATTND': 26.0, 'Exam': 83.0}}]
{}
```

In [385…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameakin torey
student IDGR-0488
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam0, 119, 37, 44
set flag1
[{'chem': {'Quiz': 99.0, 'HW': 125.0, 'ATTND': 26.0, 'Exam': 83.0}}, {'phy': {'Quiz':
0.0, 'HW': 119.0, 'ATTND': 37.0, 'Exam': 44.0}}]
{}
```

In [386…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameakin torey
student IDGR-0488
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam90, 47, 88, 18
set flag1
[{'chem': {'Quiz': 99.0, 'HW': 125.0, 'ATTND': 26.0, 'Exam': 83.0}}, {'phy': {'Quiz':
0.0, 'HW': 119.0, 'ATTND': 37.0, 'Exam': 44.0}}, {'bio': {'Quiz': 90.0, 'HW': 47.0,
'ATTND': 88.0, 'Exam': 18.0}}]
{}
```

In [387…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
```

```
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameakin torey
student IDGR-0488
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam109, 144, 14, 65
set flag1
[{'chem': {'Quiz': 99.0, 'HW': 125.0, 'ATTND': 26.0, 'Exam': 83.0}}, {'phy': {'Quiz':
0.0, 'HW': 119.0, 'ATTND': 37.0, 'Exam': 44.0}}, {'bio': {'Quiz': 90.0, 'HW': 47.0,
'ATTND': 88.0, 'Exam': 18.0}}, {'math': {'Quiz': 109.0, 'HW': 144.0, 'ATTND': 14.0,
'Exam': 65.0}}]
{}
```

In [388…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameakin torey
student IDGR-0488
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam109, 144, 14, 65
set flag2
64.44
32.93973684210526
36.66105263157895
58.27894736842105
this is the mean score of the student: 48.08
([64.44, 32.93973684210526, 36.66105263157895, 58.27894736842105], 48.08, 'F', 2.404,
'Fail')
[]
{'akin torey_GR-0488': [{'chem': {'Quiz': 99.0, 'HW': 125.0, 'ATTND': 26.0, 'Exam': 8
3.0}}, {'phy': {'Quiz': 0.0, 'HW': 119.0, 'ATTND': 37.0, 'Exam': 44.0}}, {'bio': {'Qu
iz': 90.0, 'HW': 47.0, 'ATTND': 88.0, 'Exam': 18.0}}, {'math': {'Quiz': 109.0, 'HW':
144.0, 'ATTND': 14.0, 'Exam': 65.0}}, {'Subject Scores': [64.44, 32.93973684210526, 3
6.66105263157895, 58.27894736842105], 'Overall Score': 48.08, 'Grade': 'F', 'GPA': 2.
404, 'Status': 'Fail'}]}
```

In [389…

```
res = avgScoreComput('akin torey_GR-0488', studSubsDict['akin torey_GR-0488'])
```

```
64.44
32.93973684210526
36.66105263157895
58.27894736842105
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-389-fb00e7b7f40b> in <module>
----> 1 res = avgScoreComput('akin torey_GR-0488', studSubsDict['akin torey_GR-0488']
)

<ipython-input-379-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [390…

```
res
```

Out[390]:
```
([64.44, 32.93973684210526, 36.66105263157895, 58.27894736842105],
 48.08,
 'F',
 2.404,
 'Fail')
```

In [391…
```
studSubsDict['akin torey_GR-0488']
```

Out[391]:
```
[{'chem': {'Quiz': 99.0, 'HW': 125.0, 'ATTND': 26.0, 'Exam': 83.0}},
 {'phy': {'Quiz': 0.0, 'HW': 119.0, 'ATTND': 37.0, 'Exam': 44.0}},
 {'bio': {'Quiz': 90.0, 'HW': 47.0, 'ATTND': 88.0, 'Exam': 18.0}},
 {'math': {'Quiz': 109.0, 'HW': 144.0, 'ATTND': 14.0, 'Exam': 65.0}},
 {'Subject Scores': [64.44,
   32.93973684210526,
   36.66105263157895,
   58.27894736842105],
  'Overall Score': 48.08,
  'Grade': 'F',
  'GPA': 2.404,
  'Status': 'Fail'}]
```

In [ ]:
```
```

In [402…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
```

```
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [403…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
```

```
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameluke brant
student IDGR-0489
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam171, 129, 91, 70
set flag1
[{'chem': {'Quiz': 171.0, 'HW': 129.0, 'ATTND': 91.0, 'Exam': 70.0}}]
{}
```

In [404…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)
```

```
    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameluke brant
student IDGR-0489
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam151, 63, 93, 19
set flag1
[{'chem': {'Quiz': 171.0, 'HW': 129.0, 'ATTND': 91.0, 'Exam': 70.0}}, {'phy': {'Qui
z': 151.0, 'HW': 63.0, 'ATTND': 93.0, 'Exam': 19.0}}]
{}
```

In [405…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameluke brant
student IDGR-0489
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam117, 47, 51, 25
set flag1
[{'chem': {'Quiz': 171.0, 'HW': 129.0, 'ATTND': 91.0, 'Exam': 70.0}}, {'phy': {'Qui
z': 151.0, 'HW': 63.0, 'ATTND': 93.0, 'Exam': 19.0}}, {'bio': {'Quiz': 117.0, 'HW': 4
7.0, 'ATTND': 51.0, 'Exam': 25.0}}]
{}
```

In [406…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameluke brant
student IDGR-0489
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam121, 16, 22, 41
set flag1
[{'chem': {'Quiz': 171.0, 'HW': 129.0, 'ATTND': 91.0, 'Exam': 70.0}}, {'phy': {'Qui
z': 151.0, 'HW': 63.0, 'ATTND': 93.0, 'Exam': 19.0}}, {'bio': {'Quiz': 117.0, 'HW': 4
7.0, 'ATTND': 51.0, 'Exam': 25.0}}, {'math': {'Quiz': 121.0, 'HW': 16.0, 'ATTND': 22.
0, 'Exam': 41.0}}]
{}
```

In [407…

```
# automating inputs

name = input('student name')
```

```
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameluke brant
student IDGR-0489
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam121, 16, 22, 41
set flag2
78.65921052631579
48.610789473684214
39.30921052631579
40.74842105263158
this is the mean score of the student: 51.83
([78.65921052631579, 48.610789473684214, 39.30921052631579, 40.74842105263158], 51.8
3, 'E', 2.5915, 'Retake')
[]
{'luke brant_GR-0489': [{'chem': {'Quiz': 171.0, 'HW': 129.0, 'ATTND': 91.0, 'Exam':
70.0}}, {'phy': {'Quiz': 151.0, 'HW': 63.0, 'ATTND': 93.0, 'Exam': 19.0}}, {'bio':
{'Quiz': 117.0, 'HW': 47.0, 'ATTND': 51.0, 'Exam': 25.0}}, {'math': {'Quiz': 121.0,
'HW': 16.0, 'ATTND': 22.0, 'Exam': 41.0}}, {'Subject Scores': [78.65921052631579, 48.
610789473684214, 39.30921052631579, 40.74842105263158], 'Overall Score': 51.83, 'Grad
e': 'E', 'GPA': 2.5915, 'Status': 'Retake'}]}
```

In [408…

```
res = avgScoreComput('luke brant_GR-0489', studSubsDict['luke brant_GR-0489'])
```

```
78.65921052631579
48.610789473684214
39.30921052631579
40.74842105263158
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-408-de15ecf8247b> in <module>
----> 1 res = avgScoreComput('luke brant_GR-0489', studSubsDict['luke brant_GR-0489']
)

<ipython-input-402-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [409…
```
res
```

Out[409]:
```
([78.65921052631579, 48.610789473684214, 39.30921052631579, 40.74842105263158],
 51.83,
 'E',
 2.5915,
 'Retake')
```

In [410…
```
studSubsDict['luke brant_GR-0489']
```

Out[410]:
```
[{'chem': {'Quiz': 171.0, 'HW': 129.0, 'ATTND': 91.0, 'Exam': 70.0}},
 {'phy': {'Quiz': 151.0, 'HW': 63.0, 'ATTND': 93.0, 'Exam': 19.0}},
 {'bio': {'Quiz': 117.0, 'HW': 47.0, 'ATTND': 51.0, 'Exam': 25.0}},
 {'math': {'Quiz': 121.0, 'HW': 16.0, 'ATTND': 22.0, 'Exam': 41.0}},
 {'Subject Scores': [78.65921052631579,
   48.610789473684214,
   39.30921052631579,
   40.74842105263158],
  'Overall Score': 51.83,
  'Grade': 'E',
  'GPA': 2.5915,
  'Status': 'Retake'}]
```

In [412…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
```

```
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [413…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namejames kenyata
student IDGR-0490
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam51, 103, 39, 98
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 103.0, 'ATTND': 39.0, 'Exam': 98.0}}]
{}
```

In [414…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
```

```
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namejames kenyata
student IDGR-0490
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam41, 31, 90, 18
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 103.0, 'ATTND': 39.0, 'Exam': 98.0}}, {'phy': {'Quiz':
41.0, 'HW': 31.0, 'ATTND': 90.0, 'Exam': 18.0}}]
{}
```

In [415…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)
```

```
    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namejames kenyata
student IDGR-0490
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam188, 156, 13, 72
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 103.0, 'ATTND': 39.0, 'Exam': 98.0}}, {'phy': {'Quiz':
41.0, 'HW': 31.0, 'ATTND': 90.0, 'Exam': 18.0}}, {'bio': {'Quiz': 188.0, 'HW': 156.0,
'ATTND': 13.0, 'Exam': 72.0}}]
{}
```

In [416…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namejames kenyata
student IDGR-0490
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam120, 188, 25, 52
set flag1
[{'chem': {'Quiz': 51.0, 'HW': 103.0, 'ATTND': 39.0, 'Exam': 98.0}}, {'phy': {'Quiz':
41.0, 'HW': 31.0, 'ATTND': 90.0, 'Exam': 18.0}}, {'bio': {'Quiz': 188.0, 'HW': 156.0,
'ATTND': 13.0, 'Exam': 72.0}}, {'math': {'Quiz': 120.0, 'HW': 188.0, 'ATTND': 25.0,
'Exam': 52.0}}]
{}
```

In [417…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namejames kenyata
student IDGR-0490
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam120, 188, 25, 52
set flag2
63.19921052631579
27.911052631578947
74.585
59.27447368421052
this is the mean score of the student: 56.24
([63.19921052631579, 27.911052631578947, 74.585, 59.27447368421052], 56.24, 'D', 2.81
20000000000003, 'Pass')
[]
{'james kenyata_GR-0490': [{'chem': {'Quiz': 51.0, 'HW': 103.0, 'ATTND': 39.0, 'Exa
m': 98.0}}, {'phy': {'Quiz': 41.0, 'HW': 31.0, 'ATTND': 90.0, 'Exam': 18.0}}, {'bio':
{'Quiz': 188.0, 'HW': 156.0, 'ATTND': 13.0, 'Exam': 72.0}}, {'math': {'Quiz': 120.0,
'HW': 188.0, 'ATTND': 25.0, 'Exam': 52.0}}, {'Subject Scores': [63.19921052631579, 2
7.911052631578947, 74.585, 59.27447368421052], 'Overall Score': 56.24, 'Grade': 'D',
'GPA': 2.8120000000000003, 'Status': 'Pass'}]}
```

In [418…
```
res = avgScoreComput('james kenyata_GR-0490', studSubsDict['james kenyata_GR-0490'])
```

```
63.19921052631579
27.911052631578947
74.585
59.27447368421052
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-418-0d8609813bf9> in <module>
----> 1 res = avgScoreComput('james kenyata_GR-0490', studSubsDict['james kenyata_GR-
0490'])

<ipython-input-412-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [419…
```
res
```

Out[419]:
```
([63.19921052631579, 27.911052631578947, 74.585, 59.27447368421052],
 56.24,
 'D',
 2.8120000000000003,
 'Pass')
```

In [ ]:
```
studSubsDict['james kenyata_GR-0490']
```

In [ ]:
```

```

In [420…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):
```

```
    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
```

```
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [423…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namengugi tionga
student IDGR-0491
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam155, 111, 14, 96
set flag1
[{'chem': {'Quiz': 155.0, 'HW': 111.0, 'ATTND': 14.0, 'Exam': 96.0}}]
{}
```

In [424…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namengugi tionga
student IDGR-0491
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam12, 77, 70, 43
set flag1
[{'chem': {'Quiz': 155.0, 'HW': 111.0, 'ATTND': 14.0, 'Exam': 96.0}}, {'phy': {'Qui
z': 12.0, 'HW': 77.0, 'ATTND': 70.0, 'Exam': 43.0}}]
{}
```

In [425…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
```

```
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namengugi tionga
student IDGR-0491
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam17, 136, 42, 27
set flag1
[{'chem': {'Quiz': 155.0, 'HW': 111.0, 'ATTND': 14.0, 'Exam': 96.0}}, {'phy': {'Qui
z': 12.0, 'HW': 77.0, 'ATTND': 70.0, 'Exam': 43.0}}, {'bio': {'Quiz': 17.0, 'HW': 13
6.0, 'ATTND': 42.0, 'Exam': 27.0}}]
{}
```

In [426…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status
```

```
    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namengugi tionga
student IDGR-0491
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam51, 145, 36, 22
set flag1
[{'chem': {'Quiz': 155.0, 'HW': 111.0, 'ATTND': 14.0, 'Exam': 96.0}}, {'phy': {'Qui
z': 12.0, 'HW': 77.0, 'ATTND': 70.0, 'Exam': 43.0}}, {'bio': {'Quiz': 17.0, 'HW': 13
6.0, 'ATTND': 42.0, 'Exam': 27.0}}, {'math': {'Quiz': 51.0, 'HW': 145.0, 'ATTND': 36.
0, 'Exam': 22.0}}]
{}
```

In [427…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namengugi tionga
student IDGR-0491
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam51, 145, 36, 22
set flag2
76.26684210526315
35.21368421052631
30.281052631578945
33.46
this is the mean score of the student: 43.81
([76.26684210526315, 35.21368421052631, 30.281052631578945, 33.46], 43.81, 'F', 2.190
5, 'Fail')
[]
{'ngugi tionga_GR-0491': [{'chem': {'Quiz': 155.0, 'HW': 111.0, 'ATTND': 14.0, 'Exa
m': 96.0}}, {'phy': {'Quiz': 12.0, 'HW': 77.0, 'ATTND': 70.0, 'Exam': 43.0}}, {'bio':
{'Quiz': 17.0, 'HW': 136.0, 'ATTND': 42.0, 'Exam': 27.0}}, {'math': {'Quiz': 51.0, 'H
W': 145.0, 'ATTND': 36.0, 'Exam': 22.0}}, {'Subject Scores': [76.26684210526315, 35.2
1368421052631, 30.281052631578945, 33.46], 'Overall Score': 43.81, 'Grade': 'F', 'GP
A': 2.1905, 'Status': 'Fail'}]}
```

In [428…
```
res = avgScoreComput('ngugi tionga_GR-0491', studSubsDict['ngugi tionga_GR-0491'])
```

```
76.26684210526315
35.21368421052631
30.281052631578945
33.46
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-428-b65481262756> in <module>
----> 1 res = avgScoreComput('ngugi tionga_GR-0491', studSubsDict['ngugi tionga_GR-04
91'])

<ipython-input-420-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [429…
```
res
```

Out[429]:
```
([76.26684210526315, 35.21368421052631, 30.281052631578945, 33.46],
 43.81,
 'F',
 2.1905,
 'Fail')
```

In [430…
```
studSubsDict['ngugi tionga_GR-0491']
```

Out[430]:
```
[{'chem': {'Quiz': 155.0, 'HW': 111.0, 'ATTND': 14.0, 'Exam': 96.0}},
 {'phy': {'Quiz': 12.0, 'HW': 77.0, 'ATTND': 70.0, 'Exam': 43.0}},
 {'bio': {'Quiz': 17.0, 'HW': 136.0, 'ATTND': 42.0, 'Exam': 27.0}},
 {'math': {'Quiz': 51.0, 'HW': 145.0, 'ATTND': 36.0, 'Exam': 22.0}},
 {'Subject Scores': [76.26684210526315,
   35.21368421052631,
   30.281052631578945,
   33.46],
  'Overall Score': 43.81,
  'Grade': 'F',
  'GPA': 2.1905,
  'Status': 'Fail'}]
```

In [431…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
```

```
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [432…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
```

```
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameokoro eze
student IDGR-0492
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam147, 153, 31, 48
set flag1
[{'chem': {'Quiz': 147.0, 'HW': 153.0, 'ATTND': 31.0, 'Exam': 48.0}}]
{}
```

In [433…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameokoro eze
student IDGR-0492
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam0, 97, 54, 75
set flag1
[{'chem': {'Quiz': 147.0, 'HW': 153.0, 'ATTND': 31.0, 'Exam': 48.0}}, {'phy': {'Qui
z': 0.0, 'HW': 97.0, 'ATTND': 54.0, 'Exam': 75.0}}]
{}
```

In [434…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameokoro eze
student IDGR-0492
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam61, 66, 58, 50
set flag1
[{'chem': {'Quiz': 147.0, 'HW': 153.0, 'ATTND': 31.0, 'Exam': 48.0}}, {'phy': {'Qui
z': 0.0, 'HW': 97.0, 'ATTND': 54.0, 'Exam': 75.0}}, {'bio': {'Quiz': 61.0, 'HW': 66.
0, 'ATTND': 58.0, 'Exam': 50.0}}]
{}
```

In [436…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
```

```
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameokoro eze
student IDGR-0492
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam79, 149, 71, 76
set flag1
[{'chem': {'Quiz': 147.0, 'HW': 153.0, 'ATTND': 31.0, 'Exam': 48.0}}, {'phy': {'Qui
z': 0.0, 'HW': 97.0, 'ATTND': 54.0, 'Exam': 75.0}}, {'bio': {'Quiz': 61.0, 'HW': 66.
0, 'ATTND': 58.0, 'Exam': 50.0}}, {'math': {'Quiz': 79.0, 'HW': 149.0, 'ATTND': 71.0,
'Exam': 76.0}}]
{}
```

In [437…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameokoro eze
student IDGR-0492
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam79, 149, 71, 76
set flag2
59.80447368421052
46.65789473684211
43.5921052631579
65.79184210526316
this is the mean score of the student: 53.96
([59.80447368421052, 46.65789473684211, 43.5921052631579, 65.79184210526316], 53.96,
'E', 2.698, 'Retake')
[]
{'okoro eze_GR-0492': [{'chem': {'Quiz': 147.0, 'HW': 153.0, 'ATTND': 31.0, 'Exam': 4
8.0}}, {'phy': {'Quiz': 0.0, 'HW': 97.0, 'ATTND': 54.0, 'Exam': 75.0}}, {'bio': {'Qui
z': 61.0, 'HW': 66.0, 'ATTND': 58.0, 'Exam': 50.0}}, {'math': {'Quiz': 79.0, 'HW': 14
9.0, 'ATTND': 71.0, 'Exam': 76.0}}, {'Subject Scores': [59.80447368421052, 46.6578947
3684211, 43.5921052631579, 65.79184210526316], 'Overall Score': 53.96, 'Grade': 'E',
'GPA': 2.698, 'Status': 'Retake'}]}
```

In [439…

```
res = avgScoreComput('okoro eze_GR-0492', studSubsDict['okoro eze_GR-0492'])
```

```
59.80447368421052
46.65789473684211
43.5921052631579
65.79184210526316
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-439-4ff1c03b846d> in <module>
----> 1 res = avgScoreComput('okoro eze_GR-0492', studSubsDict['okoro eze_GR-0492'])

<ipython-input-431-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [440…

```
res
```

Out[440]:
```
([59.80447368421052, 46.65789473684211, 43.5921052631579, 65.79184210526316],
 53.96,
 'E',
 2.698,
 'Retake')
```

In [441…
```
studSubsDict['okoro eze_GR-0492']
```

Out[441]:
```
[{'chem': {'Quiz': 147.0, 'HW': 153.0, 'ATTND': 31.0, 'Exam': 48.0}},
 {'phy': {'Quiz': 0.0, 'HW': 97.0, 'ATTND': 54.0, 'Exam': 75.0}},
 {'bio': {'Quiz': 61.0, 'HW': 66.0, 'ATTND': 58.0, 'Exam': 50.0}},
 {'math': {'Quiz': 79.0, 'HW': 149.0, 'ATTND': 71.0, 'Exam': 76.0}},
 {'Subject Scores': [59.80447368421052,
   46.65789473684211,
   43.5921052631579,
   65.79184210526316],
  'Overall Score': 53.96,
  'Grade': 'E',
  'GPA': 2.698,
  'Status': 'Retake'}]
```

In [442…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
```

```
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [444…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
```

```
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameagatha chiluba
student IDGR-0493
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam187, 106, 67, 83
set flag1
[{'chem': {'Quiz': 187.0, 'HW': 106.0, 'ATTND': 67.0, 'Exam': 83.0}}]
{}
```

In [445…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
```

```
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameagatha chiluba
student IDGR-0493
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam117, 88, 86, 61
set flag1
[{'chem': {'Quiz': 187.0, 'HW': 106.0, 'ATTND': 67.0, 'Exam': 83.0}}, {'phy': {'Qui
z': 117.0, 'HW': 88.0, 'ATTND': 86.0, 'Exam': 61.0}}]
{}
```

In [446…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameagatha chiluba
student IDGR-0493
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam149, 100, 64, 80
set flag1
[{'chem': {'Quiz': 187.0, 'HW': 106.0, 'ATTND': 67.0, 'Exam': 83.0}}, {'phy': {'Qui
z': 117.0, 'HW': 88.0, 'ATTND': 86.0, 'Exam': 61.0}}, {'bio': {'Quiz': 149.0, 'HW': 1
00.0, 'ATTND': 64.0, 'Exam': 80.0}}]
{}
```

In [447…
```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameagatha chiluba
student IDGR-0493
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam133, 79, 55, 70
set flag1
[{'chem': {'Quiz': 187.0, 'HW': 106.0, 'ATTND': 67.0, 'Exam': 83.0}}, {'phy': {'Qui
z': 117.0, 'HW': 88.0, 'ATTND': 86.0, 'Exam': 61.0}}, {'bio': {'Quiz': 149.0, 'HW': 1
00.0, 'ATTND': 64.0, 'Exam': 80.0}}, {'math': {'Quiz': 133.0, 'HW': 79.0, 'ATTND': 5
5.0, 'Exam': 70.0}}]
{}
```

In [448…
```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
```

```
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameagatha chiluba
student IDGR-0493
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam133, 79, 55, 70
set flag2
81.95973684210526
62.40105263157895
73.82105263157894
64.21184210526316
this is the mean score of the student: 70.6
([81.95973684210526, 62.40105263157895, 73.82105263157894, 64.21184210526316], 70.6,
'C', 3.53, 'Pass')
[]
{'agatha chiluba_GR-0493': [{'chem': {'Quiz': 187.0, 'HW': 106.0, 'ATTND': 67.0, 'Exa
m': 83.0}}, {'phy': {'Quiz': 117.0, 'HW': 88.0, 'ATTND': 86.0, 'Exam': 61.0}}, {'bi
o': {'Quiz': 149.0, 'HW': 100.0, 'ATTND': 64.0, 'Exam': 80.0}}, {'math': {'Quiz': 13
3.0, 'HW': 79.0, 'ATTND': 55.0, 'Exam': 70.0}}, {'Subject Scores': [81.9597368421052
6, 62.40105263157895, 73.82105263157894, 64.21184210526316], 'Overall Score': 70.6,
'Grade': 'C', 'GPA': 3.53, 'Status': 'Pass'}]}
```

In [449…

```
res = avgScoreComput('agatha chiluba_GR-0493', studSubsDict['agatha chiluba_GR-0493'])
```

```
81.95973684210526
62.40105263157895
73.82105263157894
64.21184210526316
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-449-c3123df686f1> in <module>
----> 1 res = avgScoreComput('agatha chiluba_GR-0493', studSubsDict['agatha chiluba_GR-0493'])

<ipython-input-442-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [450…
```
res
```

Out[450]:
```
([81.95973684210526, 62.40105263157895, 73.82105263157894, 64.21184210526316],
 70.6,
 'C',
 3.53,
 'Pass')
```

In [451…
```
studSubsDict['agatha chiluba_GR-0493']
```

Out[451]:
```
[{'chem': {'Quiz': 187.0, 'HW': 106.0, 'ATTND': 67.0, 'Exam': 83.0}},
 {'phy': {'Quiz': 117.0, 'HW': 88.0, 'ATTND': 86.0, 'Exam': 61.0}},
 {'bio': {'Quiz': 149.0, 'HW': 100.0, 'ATTND': 64.0, 'Exam': 80.0}},
 {'math': {'Quiz': 133.0, 'HW': 79.0, 'ATTND': 55.0, 'Exam': 70.0}},
 {'Subject Scores': [81.95973684210526,
   62.40105263157895,
   73.82105263157894,
   64.21184210526316],
  'Overall Score': 70.6,
  'Grade': 'C',
  'GPA': 3.53,
  'Status': 'Pass'}]
```

In [452…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
```

```
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [453…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemangu joseph
student IDGR-0494
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam123, 171, 82, 48
set flag1
[{'chem': {'Quiz': 123.0, 'HW': 171.0, 'ATTND': 82.0, 'Exam': 48.0}}]
{}
```

In [454…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
```

```
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemangu joseph
student IDGR-0494
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam48, 139, 63, 35
set flag1
[{'chem': {'Quiz': 123.0, 'HW': 171.0, 'ATTND': 82.0, 'Exam': 48.0}}, {'phy': {'Qui
z': 48.0, 'HW': 139.0, 'ATTND': 63.0, 'Exam': 35.0}}]
{}
```

In [455…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)
```

```
    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemangu joseph
student IDGR-0494
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam127, 88, 87, 23
set flag1
[{'chem': {'Quiz': 123.0, 'HW': 171.0, 'ATTND': 82.0, 'Exam': 48.0}}, {'phy': {'Qui
z': 48.0, 'HW': 139.0, 'ATTND': 63.0, 'Exam': 35.0}}, {'bio': {'Quiz': 127.0, 'HW': 8
8.0, 'ATTND': 87.0, 'Exam': 23.0}}]
{}
```

In [456…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemangu joseph
student IDGR-0494
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam83, 33, 13, 10
set flag1
[{'chem': {'Quiz': 123.0, 'HW': 171.0, 'ATTND': 82.0, 'Exam': 48.0}}, {'phy': {'Qui
z': 48.0, 'HW': 139.0, 'ATTND': 63.0, 'Exam': 35.0}}, {'bio': {'Quiz': 127.0, 'HW': 8
8.0, 'ATTND': 87.0, 'Exam': 23.0}}, {'math': {'Quiz': 83.0, 'HW': 33.0, 'ATTND': 13.
0, 'Exam': 10.0}}]
{}
```

In [457…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemangu joseph
student IDGR-0494
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam83, 33, 13, 10
set flag2
63.811052631578946
41.47763157894737
47.765
21.635526315789477
this is the mean score of the student: 43.67
([63.811052631578946, 41.47763157894737, 47.765, 21.635526315789477], 43.67, 'F', 2.1
835, 'Fail')
[]
{'mangu joseph_GR-0494': [{'chem': {'Quiz': 123.0, 'HW': 171.0, 'ATTND': 82.0, 'Exa
m': 48.0}}, {'phy': {'Quiz': 48.0, 'HW': 139.0, 'ATTND': 63.0, 'Exam': 35.0}}, {'bi
o': {'Quiz': 127.0, 'HW': 88.0, 'ATTND': 87.0, 'Exam': 23.0}}, {'math': {'Quiz': 83.
0, 'HW': 33.0, 'ATTND': 13.0, 'Exam': 10.0}}, {'Subject Scores': [63.811052631578946,
41.47763157894737, 47.765, 21.635526315789477], 'Overall Score': 43.67, 'Grade': 'F',
'GPA': 2.1835, 'Status': 'Fail'}]}
```

In [459…
```
res = avgScoreComput('mangu joseph_GR-0494', studSubsDict['mangu joseph_GR-0494'])
```

```
63.811052631578946
41.47763157894737
47.765
21.635526315789477
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-459-0cbe08a4c6d9> in <module>
----> 1 res = avgScoreComput('mangu joseph_GR-0494', studSubsDict['mangu joseph_GR-04
94'])

<ipython-input-452-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [460…
```
res
```

Out[460]:
```
([63.811052631578946, 41.47763157894737, 47.765, 21.635526315789477],
 43.67,
 'F',
 2.1835,
 'Fail')
```

In [461…
```
studSubsDict['mangu joseph_GR-0494']
```

Out[461]:
```
[{'chem': {'Quiz': 123.0, 'HW': 171.0, 'ATTND': 82.0, 'Exam': 48.0}},
 {'phy': {'Quiz': 48.0, 'HW': 139.0, 'ATTND': 63.0, 'Exam': 35.0}},
 {'bio': {'Quiz': 127.0, 'HW': 88.0, 'ATTND': 87.0, 'Exam': 23.0}},
 {'math': {'Quiz': 83.0, 'HW': 33.0, 'ATTND': 13.0, 'Exam': 10.0}},
 {'Subject Scores': [63.811052631578946,
   41.47763157894737,
   47.765,
   21.635526315789477],
  'Overall Score': 43.67,
  'Grade': 'F',
  'GPA': 2.1835,
  'Status': 'Fail'}]
```

In [ ]:

In [462…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
```

```
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [463…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
```

```
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelonge jethro
student IDGR-0495
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam179, 160, 48, 93
set flag1
[{'chem': {'Quiz': 179.0, 'HW': 160.0, 'ATTND': 48.0, 'Exam': 93.0}}]
{}
```

In [464…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelonge jethro
student IDGR-0495
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam119, 119, 65, 70
set flag1
[{'chem': {'Quiz': 179.0, 'HW': 160.0, 'ATTND': 48.0, 'Exam': 93.0}}, {'phy': {'Qui
z': 119.0, 'HW': 119.0, 'ATTND': 65.0, 'Exam': 70.0}}]
{}
```

In [465…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelonge jethro
student IDGR-0495
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam120, 136, 92, 89
set flag1
[{'chem': {'Quiz': 179.0, 'HW': 160.0, 'ATTND': 48.0, 'Exam': 93.0}}, {'phy': {'Qui
z': 119.0, 'HW': 119.0, 'ATTND': 65.0, 'Exam': 70.0}}, {'bio': {'Quiz': 120.0, 'HW':
136.0, 'ATTND': 92.0, 'Exam': 89.0}}]
{}
```

In [466…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
```

```
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelonge jethro
student IDGR-0495
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam155, 97, 50, 81
set flag1
[{'chem': {'Quiz': 179.0, 'HW': 160.0, 'ATTND': 48.0, 'Exam': 93.0}}, {'phy': {'Qui
z': 119.0, 'HW': 119.0, 'ATTND': 65.0, 'Exam': 70.0}}, {'bio': {'Quiz': 120.0, 'HW':
136.0, 'ATTND': 92.0, 'Exam': 89.0}}, {'math': {'Quiz': 155.0, 'HW': 97.0, 'ATTND': 5
0.0, 'Exam': 81.0}}]
{}
```

In [467…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namelonge jethro
student IDGR-0495
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam155, 97, 50, 81
set flag2
86.88473684210527
66.40921052631579
79.45421052631579
73.21157894736842
this is the mean score of the student: 76.49
([86.88473684210527, 66.40921052631579, 79.45421052631579, 73.21157894736842], 76.49,
'B', 3.8244999999999996, 'Pass')
[]
{'longe jethro_GR-0495': [{'chem': {'Quiz': 179.0, 'HW': 160.0, 'ATTND': 48.0, 'Exa
m': 93.0}}, {'phy': {'Quiz': 119.0, 'HW': 119.0, 'ATTND': 65.0, 'Exam': 70.0}}, {'bi
o': {'Quiz': 120.0, 'HW': 136.0, 'ATTND': 92.0, 'Exam': 89.0}}, {'math': {'Quiz': 15
5.0, 'HW': 97.0, 'ATTND': 50.0, 'Exam': 81.0}}, {'Subject Scores': [86.8847368421052
7, 66.40921052631579, 79.45421052631579, 73.21157894736842], 'Overall Score': 76.49,
'Grade': 'B', 'GPA': 3.8244999999999996, 'Status': 'Pass'}]}
```

In [468…

```
res = avgScoreComput('longe jethro_GR-0495', studSubsDict['longe jethro_GR-0495'])
```

```
86.88473684210527
66.40921052631579
79.45421052631579
73.21157894736842
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-468-345b09f0fb24> in <module>
----> 1 res = avgScoreComput('longe jethro_GR-0495', studSubsDict['longe jethro_GR-04
95'])

<ipython-input-462-d4e3a58591dd> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [469…

```
res
```

Out[469]:
```
([86.88473684210527, 66.40921052631579, 79.45421052631579, 73.21157894736842],
 76.49,
 'B',
 3.8244999999999996,
 'Pass')
```

In [470…
```
studSubsDict['longe jethro_GR-0495']
```

Out[470]:
```
[{'chem': {'Quiz': 179.0, 'HW': 160.0, 'ATTND': 48.0, 'Exam': 93.0}},
 {'phy': {'Quiz': 119.0, 'HW': 119.0, 'ATTND': 65.0, 'Exam': 70.0}},
 {'bio': {'Quiz': 120.0, 'HW': 136.0, 'ATTND': 92.0, 'Exam': 89.0}},
 {'math': {'Quiz': 155.0, 'HW': 97.0, 'ATTND': 50.0, 'Exam': 81.0}},
 {'Subject Scores': [86.88473684210527,
   66.40921052631579,
   79.45421052631579,
   73.21157894736842],
  'Overall Score': 76.49,
  'Grade': 'B',
  'GPA': 3.8244999999999996,
  'Status': 'Pass'}]
```

In [ ]:

In [478…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
```

```
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [480…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
```

```
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameflorence giwa
student IDGR-0496
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam98, 172, 30, 94
set flag1
[{'chem': {'Quiz': 98.0, 'HW': 172.0, 'ATTND': 30.0, 'Exam': 94.0}}]
{}
```

In [481…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)
```

```
    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameflorence giwa
student IDGR-0496
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam122, 20, 24, 68
set flag1
[{'chem': {'Quiz': 98.0, 'HW': 172.0, 'ATTND': 30.0, 'Exam': 94.0}}, {'phy': {'Quiz':
122.0, 'HW': 20.0, 'ATTND': 24.0, 'Exam': 68.0}}]
{}
```

In [482…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameflorence giwa
student IDGR-0496
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam42, 19, 36, 84
set flag1
[{'chem': {'Quiz': 98.0, 'HW': 172.0, 'ATTND': 30.0, 'Exam': 94.0}}, {'phy': {'Quiz':
122.0, 'HW': 20.0, 'ATTND': 24.0, 'Exam': 68.0}}, {'bio': {'Quiz': 42.0, 'HW': 19.0,
'ATTND': 36.0, 'Exam': 84.0}}]
{}
```

In [483…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameflorence giwa
student IDGR-0496
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam84, 157, 18, 46
set flag1
[{'chem': {'Quiz': 98.0, 'HW': 172.0, 'ATTND': 30.0, 'Exam': 94.0}}, {'phy': {'Quiz':
122.0, 'HW': 20.0, 'ATTND': 24.0, 'Exam': 68.0}}, {'bio': {'Quiz': 42.0, 'HW': 19.0,
'ATTND': 36.0, 'Exam': 84.0}}, {'math': {'Quiz': 84.0, 'HW': 157.0, 'ATTND': 18.0, 'E
xam': 46.0}}]
{}
```

In [484…

```
# automating inputs

name = input('student name')
```

```
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student nameflorence giwa
student IDGR-0496
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam84, 157, 18, 46
set flag2
73.22263157894736
53.0821052631579
48.751578947368415
47.68789473684211
this is the mean score of the student: 55.69
([73.22263157894736, 53.0821052631579, 48.751578947368415, 47.68789473684211], 55.69,
'D', 2.7844999999999995, 'Pass')
[]
{'florence giwa_GR-0496': [{'chem': {'Quiz': 98.0, 'HW': 172.0, 'ATTND': 30.0, 'Exa
m': 94.0}}, {'phy': {'Quiz': 122.0, 'HW': 20.0, 'ATTND': 24.0, 'Exam': 68.0}}, {'bi
o': {'Quiz': 42.0, 'HW': 19.0, 'ATTND': 36.0, 'Exam': 84.0}}, {'math': {'Quiz': 84.0,
'HW': 157.0, 'ATTND': 18.0, 'Exam': 46.0}}, {'Subject Scores': [73.22263157894736, 5
3.0821052631579, 48.751578947368415, 47.68789473684211], 'Overall Score': 55.69, 'Gra
de': 'D', 'GPA': 2.7844999999999995, 'Status': 'Pass'}]}
```

In [485…

```
res = avgScoreComput('florence giwa_GR-0496', studSubsDict['florence giwa_GR-0496'])
```

```
73.22263157894736
53.0821052631579
48.751578947368415
47.68789473684211
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-485-6b4b8315cabd> in <module>
----> 1 res = avgScoreComput('florence giwa_GR-0496', studSubsDict['florence giwa_GR-0496'])

<ipython-input-478-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [486…
```
res
```

Out[486]:
```
([73.22263157894736, 53.0821052631579, 48.751578947368415, 47.68789473684211],
 55.69,
 'D',
 2.7844999999999995,
 'Pass')
```

In [487…
```
studSubsDict['florence giwa_GR-0496']
```

Out[487]:
```
[{'chem': {'Quiz': 98.0, 'HW': 172.0, 'ATTND': 30.0, 'Exam': 94.0}},
 {'phy': {'Quiz': 122.0, 'HW': 20.0, 'ATTND': 24.0, 'Exam': 68.0}},
 {'bio': {'Quiz': 42.0, 'HW': 19.0, 'ATTND': 36.0, 'Exam': 84.0}},
 {'math': {'Quiz': 84.0, 'HW': 157.0, 'ATTND': 18.0, 'Exam': 46.0}},
 {'Subject Scores': [73.22263157894736,
   53.0821052631579,
   48.751578947368415,
   47.68789473684211],
  'Overall Score': 55.69,
  'Grade': 'D',
  'GPA': 2.7844999999999995,
  'Status': 'Pass'}]
```

In [ ]:
```
```

In [488…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict
```

```
def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
```

```
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [489…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevetiva lucent
student IDGR-0497
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam17, 60, 48, 36
set flag1
[{'chem': {'Quiz': 17.0, 'HW': 60.0, 'ATTND': 48.0, 'Exam': 36.0}}]
{}
```

In [490…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
```

```
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevetiva lucent
student IDGR-0497
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam114, 109, 84, 31
set flag1
[{'chem': {'Quiz': 17.0, 'HW': 60.0, 'ATTND': 48.0, 'Exam': 36.0}}, {'phy': {'Quiz':
114.0, 'HW': 109.0, 'ATTND': 84.0, 'Exam': 31.0}}]
{}
```

In [491…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status
```

```
    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevetiva lucent
student IDGR-0497
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam177, 70, 48, 34
set flag1
[{'chem': {'Quiz': 17.0, 'HW': 60.0, 'ATTND': 48.0, 'Exam': 36.0}}, {'phy': {'Quiz':
114.0, 'HW': 109.0, 'ATTND': 84.0, 'Exam': 31.0}}, {'bio': {'Quiz': 177.0, 'HW': 70.
0, 'ATTND': 48.0, 'Exam': 34.0}}]
{}
```

In [492…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevetiva lucent
student IDGR-0497
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam60, 102, 73, 87
set flag1
[{'chem': {'Quiz': 17.0, 'HW': 60.0, 'ATTND': 48.0, 'Exam': 36.0}}, {'phy': {'Quiz':
114.0, 'HW': 109.0, 'ATTND': 84.0, 'Exam': 31.0}}, {'bio': {'Quiz': 177.0, 'HW': 70.
0, 'ATTND': 48.0, 'Exam': 34.0}}, {'math': {'Quiz': 60.0, 'HW': 102.0, 'ATTND': 73.0,
'Exam': 87.0}}]
{}
```

In [493…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevetiva lucent
student IDGR-0497
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam60, 102, 73, 87
set flag2
28.901052631578946
50.43526315789474
54.09368421052632
64.06131578947368
this is the mean score of the student: 49.37
([28.901052631578946, 50.43526315789474, 54.09368421052632, 64.06131578947368], 49.3
7, 'F', 2.4684999999999997, 'Fail')
[]
{'vetiva lucent_GR-0497': [{'chem': {'Quiz': 17.0, 'HW': 60.0, 'ATTND': 48.0, 'Exam':
36.0}}, {'phy': {'Quiz': 114.0, 'HW': 109.0, 'ATTND': 84.0, 'Exam': 31.0}}, {'bio':
{'Quiz': 177.0, 'HW': 70.0, 'ATTND': 48.0, 'Exam': 34.0}}, {'math': {'Quiz': 60.0, 'H
W': 102.0, 'ATTND': 73.0, 'Exam': 87.0}}, {'Subject Scores': [28.901052631578946, 50.
43526315789474, 54.09368421052632, 64.06131578947368], 'Overall Score': 49.37, 'Grad
e': 'F', 'GPA': 2.4684999999999997, 'Status': 'Fail'}]}
```

In [494…]
```
res = avgScoreComput('vetiva lucent_GR-0497', studSubsDict['vetiva lucent_GR-0497'])
```

```
28.901052631578946
50.43526315789474
54.09368421052632
64.06131578947368
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-494-d9d94847cea7> in <module>
----> 1 res = avgScoreComput('vetiva lucent_GR-0497', studSubsDict['vetiva lucent_GR-
0497'])

<ipython-input-488-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [495…]
```
res
```

Out[495]:
```
([28.901052631578946, 50.43526315789474, 54.09368421052632, 64.06131578947368],
 49.37,
 'F',
 2.4684999999999997,
 'Fail')
```

In [496…]
```
studSubsDict['vetiva lucent_GR-0497']
```

Out[496]:
```
[{'chem': {'Quiz': 17.0, 'HW': 60.0, 'ATTND': 48.0, 'Exam': 36.0}},
 {'phy': {'Quiz': 114.0, 'HW': 109.0, 'ATTND': 84.0, 'Exam': 31.0}},
 {'bio': {'Quiz': 177.0, 'HW': 70.0, 'ATTND': 48.0, 'Exam': 34.0}},
 {'math': {'Quiz': 60.0, 'HW': 102.0, 'ATTND': 73.0, 'Exam': 87.0}},
 {'Subject Scores': [28.901052631578946,
   50.43526315789474,
   54.09368421052632,
   64.06131578947368],
  'Overall Score': 49.37,
  'Grade': 'F',
  'GPA': 2.4684999999999997,
  'Status': 'Fail'}]
```

In [ ]:

In [497…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
```

```
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [498…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
```

```
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemelody braimoh
student IDGR-0498
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam67, 177, 68, 39
set flag1
[{'chem': {'Quiz': 67.0, 'HW': 177.0, 'ATTND': 68.0, 'Exam': 39.0}}]
{}
```

In [499…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemelody braimoh
student IDGR-0498
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam145, 65, 43, 14
set flag1
[{'chem': {'Quiz': 67.0, 'HW': 177.0, 'ATTND': 68.0, 'Exam': 39.0}}, {'phy': {'Quiz':
145.0, 'HW': 65.0, 'ATTND': 43.0, 'Exam': 14.0}}]
{}
```

In [500…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemelody braimoh
student IDGR-0498
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam162, 125, 38, 36
set flag1
[{'chem': {'Quiz': 67.0, 'HW': 177.0, 'ATTND': 68.0, 'Exam': 39.0}}, {'phy': {'Quiz':
145.0, 'HW': 65.0, 'ATTND': 43.0, 'Exam': 14.0}}, {'bio': {'Quiz': 162.0, 'HW': 125.
0, 'ATTND': 38.0, 'Exam': 36.0}}]
{}
```

In [501…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
```

```
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemelody braimoh
student IDGR-0498
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam104, 58, 36, 89
set flag1
[{'chem': {'Quiz': 67.0, 'HW': 177.0, 'ATTND': 68.0, 'Exam': 39.0}}, {'phy': {'Quiz':
145.0, 'HW': 65.0, 'ATTND': 43.0, 'Exam': 14.0}}, {'bio': {'Quiz': 162.0, 'HW': 125.
0, 'ATTND': 38.0, 'Exam': 36.0}}, {'math': {'Quiz': 104.0, 'HW': 58.0, 'ATTND': 36.0,
'Exam': 89.0}}]
{}
```

In [502…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemelody braimoh
student IDGR-0498
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam104, 58, 36, 89
set flag2
49.822631578947366
39.42131578947369
55.67736842105263
63.77
this is the mean score of the student: 52.17
([49.822631578947366, 39.42131578947369, 55.67736842105263, 63.77], 52.17, 'E', 2.608
5000000000003, 'Retake')
[]
{'melody braimoh_GR-0498': [{'chem': {'Quiz': 67.0, 'HW': 177.0, 'ATTND': 68.0, 'Exa
m': 39.0}}, {'phy': {'Quiz': 145.0, 'HW': 65.0, 'ATTND': 43.0, 'Exam': 14.0}}, {'bi
o': {'Quiz': 162.0, 'HW': 125.0, 'ATTND': 38.0, 'Exam': 36.0}}, {'math': {'Quiz': 10
4.0, 'HW': 58.0, 'ATTND': 36.0, 'Exam': 89.0}}, {'Subject Scores': [49.82263157894736
6, 39.42131578947369, 55.67736842105263, 63.77], 'Overall Score': 52.17, 'Grade':
'E', 'GPA': 2.6085000000000003, 'Status': 'Retake'}]}
```

In [503…

```
res = avgScoreComput('melody braimoh_GR-0498', studSubsDict['melody braimoh_GR-0498'])
```

```
49.822631578947366
39.42131578947369
55.67736842105263
63.77
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-503-b71a11a97580> in <module>
----> 1 res = avgScoreComput('melody braimoh_GR-0498', studSubsDict['melody braimoh_G
R-0498'])

<ipython-input-497-a9f59b08a929> in avgScoreComput(studname, sDictKeys)
     37         #print(subjScores.keys())
     38         #print(subjScores['Quiz'])
---> 39         for sub in list(subjScores.keys()):
     40             #print(sub)
     41             if sub=='Quiz':

AttributeError: 'list' object has no attribute 'keys'
```

In [504…

```
res
```

Out[504]:
```
([49.822631578947366, 39.42131578947369, 55.67736842105263, 63.77],
 52.17,
 'E',
 2.6085000000000003,
 'Retake')
```

In [505…
```
studSubsDict['melody braimoh_GR-0498']
```

Out[505]:
```
[{'chem': {'Quiz': 67.0, 'HW': 177.0, 'ATTND': 68.0, 'Exam': 39.0}},
 {'phy': {'Quiz': 145.0, 'HW': 65.0, 'ATTND': 43.0, 'Exam': 14.0}},
 {'bio': {'Quiz': 162.0, 'HW': 125.0, 'ATTND': 38.0, 'Exam': 36.0}},
 {'math': {'Quiz': 104.0, 'HW': 58.0, 'ATTND': 36.0, 'Exam': 89.0}},
 {'Subject Scores': [49.822631578947366,
   39.42131578947369,
   55.67736842105263,
   63.77],
  'Overall Score': 52.17,
  'Grade': 'E',
  'GPA': 2.6085000000000003,
  'Status': 'Retake'}]
```

In [ ]:
```
```

In [506…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
```

```
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [507…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
```

```
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevictor ihab
student IDGR-0499
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam177, 14, 83, 75
set flag1
[{'chem': {'Quiz': 177.0, 'HW': 14.0, 'ATTND': 83.0, 'Exam': 75.0}}]
{}
```

In [508…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)
```

```
    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevictor ihab
student IDGR-0499
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam135, 6, 50, 33
set flag1
[{'chem': {'Quiz': 177.0, 'HW': 14.0, 'ATTND': 83.0, 'Exam': 75.0}}, {'phy': {'Quiz':
135.0, 'HW': 6.0, 'ATTND': 50.0, 'Exam': 33.0}}]
{}
```

In [509…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevictor ihab
student IDGR-0499
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam155, 90, 51, 35
set flag1
[{'chem': {'Quiz': 177.0, 'HW': 14.0, 'ATTND': 83.0, 'Exam': 75.0}}, {'phy': {'Quiz':
135.0, 'HW': 6.0, 'ATTND': 50.0, 'Exam': 33.0}}, {'bio': {'Quiz': 155.0, 'HW': 90.0,
'ATTND': 51.0, 'Exam': 35.0}}]
{}
```

In [510…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevictor ihab
student IDGR-0499
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam43, 44, 63, 48
set flag1
[{'chem': {'Quiz': 177.0, 'HW': 14.0, 'ATTND': 83.0, 'Exam': 75.0}}, {'phy': {'Quiz':
135.0, 'HW': 6.0, 'ATTND': 50.0, 'Exam': 33.0}}, {'bio': {'Quiz': 155.0, 'HW': 90.0,
'ATTND': 51.0, 'Exam': 35.0}}, {'math': {'Quiz': 43.0, 'HW': 44.0, 'ATTND': 63.0, 'Ex
am': 48.0}}]
{}
```

In [511…

```
# automating inputs

name = input('student name')
```

```
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namevictor ihab
student IDGR-0499
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam43, 44, 63, 48
set flag2
71.67763157894737
42.229473684210525
53.00394736842105
38.77815789473684
this is the mean score of the student: 51.42
([71.67763157894737, 42.229473684210525, 53.00394736842105, 38.77815789473684], 51.4
2, 'E', 2.5709999999999997, 'Retake')
[]
{'victor ihab_GR-0499': [{'chem': {'Quiz': 177.0, 'HW': 14.0, 'ATTND': 83.0, 'Exam':
75.0}}, {'phy': {'Quiz': 135.0, 'HW': 6.0, 'ATTND': 50.0, 'Exam': 33.0}}, {'bio': {'Q
uiz': 155.0, 'HW': 90.0, 'ATTND': 51.0, 'Exam': 35.0}}, {'math': {'Quiz': 43.0, 'HW':
44.0, 'ATTND': 63.0, 'Exam': 48.0}}, {'Subject Scores': [71.67763157894737, 42.229473
684210525, 53.00394736842105, 38.77815789473684], 'Overall Score': 51.42, 'Grade':
'E', 'GPA': 2.5709999999999997, 'Status': 'Retake'}]}
```

In [512…

```
res
```

Out[512]:

```
([71.67763157894737, 42.229473684210525, 53.00394736842105, 38.77815789473684],
 51.42,
 'E',
 2.5709999999999997,
 'Retake')
```

In [513…
```
studSubsDict['victor ihab_GR-0499']
```

Out[513]:
```
[{'chem': {'Quiz': 177.0, 'HW': 14.0, 'ATTND': 83.0, 'Exam': 75.0}},
 {'phy': {'Quiz': 135.0, 'HW': 6.0, 'ATTND': 50.0, 'Exam': 33.0}},
 {'bio': {'Quiz': 155.0, 'HW': 90.0, 'ATTND': 51.0, 'Exam': 35.0}},
 {'math': {'Quiz': 43.0, 'HW': 44.0, 'ATTND': 63.0, 'Exam': 48.0}},
 {'Subject Scores': [71.67763157894737,
   42.229473684210525,
   53.00394736842105,
   38.77815789473684],
  'Overall Score': 51.42,
  'Grade': 'E',
  'GPA': 2.5709999999999997,
  'Status': 'Retake'}]
```

In [ ]:
```
```

In [514…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
```

```
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [515…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemimi trucker
student IDGR-0500
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam148, 182, 80, 46
set flag1
[{'chem': {'Quiz': 148.0, 'HW': 182.0, 'ATTND': 80.0, 'Exam': 46.0}}]
{}
```

In [516…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemimi trucker
student IDGR-0500
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam34, 104, 73, 23
set flag1
[{'chem': {'Quiz': 148.0, 'HW': 182.0, 'ATTND': 80.0, 'Exam': 46.0}}, {'phy': {'Qui
z': 34.0, 'HW': 104.0, 'ATTND': 73.0, 'Exam': 23.0}}]
{}
```

In [517…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemimi trucker
student IDGR-0500
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam141, 146, 41, 80
set flag1
[{'chem': {'Quiz': 148.0, 'HW': 182.0, 'ATTND': 80.0, 'Exam': 46.0}}, {'phy': {'Qui
z': 34.0, 'HW': 104.0, 'ATTND': 73.0, 'Exam': 23.0}}, {'bio': {'Quiz': 141.0, 'HW': 1
46.0, 'ATTND': 41.0, 'Exam': 80.0}}]
{}
```

In [518…

```python
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
```

```
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemimi trucker
student IDGR-0500
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam61, 136, 44, 53
set flag1
[{'chem': {'Quiz': 148.0, 'HW': 182.0, 'ATTND': 80.0, 'Exam': 46.0}}, {'phy': {'Qui
z': 34.0, 'HW': 104.0, 'ATTND': 73.0, 'Exam': 23.0}}, {'bio': {'Quiz': 141.0, 'HW': 1
46.0, 'ATTND': 41.0, 'Exam': 80.0}}, {'math': {'Quiz': 61.0, 'HW': 136.0, 'ATTND': 4
4.0, 'Exam': 53.0}}]
{}
```

In [519…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemimi trucker
student IDGR-0500
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam61, 136, 44, 53
set flag2
67.51684210526315
32.593947368421055
73.31447368421053
48.65842105263158
this is the mean score of the student: 55.52
([67.51684210526315, 32.593947368421055, 73.31447368421053, 48.65842105263158], 55.5
2, 'D', 2.7760000000000002, 'Pass')
[]
{'mimi trucker_GR-0500': [{'chem': {'Quiz': 148.0, 'HW': 182.0, 'ATTND': 80.0, 'Exa
m': 46.0}}, {'phy': {'Quiz': 34.0, 'HW': 104.0, 'ATTND': 73.0, 'Exam': 23.0}}, {'bi
o': {'Quiz': 141.0, 'HW': 146.0, 'ATTND': 41.0, 'Exam': 80.0}}, {'math': {'Quiz': 61.
0, 'HW': 136.0, 'ATTND': 44.0, 'Exam': 53.0}}, {'Subject Scores': [67.51684210526315,
32.593947368421055, 73.31447368421053, 48.65842105263158], 'Overall Score': 55.52, 'G
rade': 'D', 'GPA': 2.7760000000000002, 'Status': 'Pass'}]}
```

In [520…

```
res
```

Out[520]:

```
([67.51684210526315, 32.593947368421055, 73.31447368421053, 48.65842105263158],
 55.52,
 'D',
 2.7760000000000002,
 'Pass')
```

In [522…

```
studSubsDict['mimi trucker_GR-0500']
```

Out[522]:

```
[{'chem': {'Quiz': 148.0, 'HW': 182.0, 'ATTND': 80.0, 'Exam': 46.0}},
 {'phy': {'Quiz': 34.0, 'HW': 104.0, 'ATTND': 73.0, 'Exam': 23.0}},
 {'bio': {'Quiz': 141.0, 'HW': 146.0, 'ATTND': 41.0, 'Exam': 80.0}},
 {'math': {'Quiz': 61.0, 'HW': 136.0, 'ATTND': 44.0, 'Exam': 53.0}},
 {'Subject Scores': [67.51684210526315,
   32.593947368421055,
   73.31447368421053,
   48.65842105263158],
  'Overall Score': 55.52,
  'Grade': 'D',
  'GPA': 2.7760000000000002,
  'Status': 'Pass'}]
```

In [ ]:

In [523…

```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
```

```
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [524…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []
```

```
print(perfList)
print(studSubsDict)
```

```
student namemaguel peter
student IDGR-0501
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam23, 95, 63, 27
set flag1
[{'chem': {'Quiz': 23.0, 'HW': 95.0, 'ATTND': 63.0, 'Exam': 27.0}}]
{}
```

In [525…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemaguel peter
student IDGR-0501
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam81, 143, 80, 16
set flag1
[{'chem': {'Quiz': 23.0, 'HW': 95.0, 'ATTND': 63.0, 'Exam': 27.0}}, {'phy': {'Quiz':
81.0, 'HW': 143.0, 'ATTND': 80.0, 'Exam': 16.0}}]
{}
```

In [526…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
```

```
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemaguel peter
student IDGR-0501
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam183, 43, 28, 25
set flag1
[{'chem': {'Quiz': 23.0, 'HW': 95.0, 'ATTND': 63.0, 'Exam': 27.0}}, {'phy': {'Quiz':
81.0, 'HW': 143.0, 'ATTND': 80.0, 'Exam': 16.0}}, {'bio': {'Quiz': 183.0, 'HW': 43.0,
'ATTND': 28.0, 'Exam': 25.0}}]
{}
```

In [527…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemaguel peter
student IDGR-0501
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam33, 79, 95, 62
set flag1
[{'chem': {'Quiz': 23.0, 'HW': 95.0, 'ATTND': 63.0, 'Exam': 27.0}}, {'phy': {'Quiz':
81.0, 'HW': 143.0, 'ATTND': 80.0, 'Exam': 16.0}}, {'bio': {'Quiz': 183.0, 'HW': 43.0,
'ATTND': 28.0, 'Exam': 25.0}}, {'math': {'Quiz': 33.0, 'HW': 79.0, 'ATTND': 95.0, 'Ex
am': 62.0}}]
{}
```

In [528…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
```

```
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namemaguel peter
student IDGR-0501
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam33, 79, 95, 62
set flag2
30.616578947368424
40.95894736842106
46.53947368421053
49.982368421052634
this is the mean score of the student: 42.02
([30.616578947368424, 40.95894736842106, 46.53947368421053, 49.982368421052634], 42.0
2, 'F', 2.101, 'Fail')
[]
{'maguel peter_GR-0501': [{'chem': {'Quiz': 23.0, 'HW': 95.0, 'ATTND': 63.0, 'Exam':
27.0}}, {'phy': {'Quiz': 81.0, 'HW': 143.0, 'ATTND': 80.0, 'Exam': 16.0}}, {'bio':
{'Quiz': 183.0, 'HW': 43.0, 'ATTND': 28.0, 'Exam': 25.0}}, {'math': {'Quiz': 33.0, 'H
W': 79.0, 'ATTND': 95.0, 'Exam': 62.0}}, {'Subject Scores': [30.616578947368424, 40.9
5894736842106, 46.53947368421053, 49.982368421052634], 'Overall Score': 42.02, 'Grad
e': 'F', 'GPA': 2.101, 'Status': 'Fail'}]}
```

In [529…
```
res
```

Out[529]:
```
([30.616578947368424,
  40.95894736842106,
  46.53947368421053,
  49.982368421052634],
 42.02,
 'F',
 2.101,
 'Fail')
```

In [530…
```
studSubsDict['maguel peter_GR-0501']
```

Out[530]:
```
[{'chem': {'Quiz': 23.0, 'HW': 95.0, 'ATTND': 63.0, 'Exam': 27.0}},
 {'phy': {'Quiz': 81.0, 'HW': 143.0, 'ATTND': 80.0, 'Exam': 16.0}},
 {'bio': {'Quiz': 183.0, 'HW': 43.0, 'ATTND': 28.0, 'Exam': 25.0}},
 {'math': {'Quiz': 33.0, 'HW': 79.0, 'ATTND': 95.0, 'Exam': 62.0}},
 {'Subject Scores': [30.616578947368424,
   40.95894736842106,
   46.53947368421053,
   49.982368421052634],
  'Overall Score': 42.02,
  'Grade': 'F',
  'GPA': 2.101,
  'Status': 'Fail'}]
```

In [531…
```
import statistics

perfList = []
studSubsDict = dict()

def scoreAggregator(name, ID, subject, assessTypeList, scoreList):

    tempDict = dict()
    #asTList = assessTypeList.split(',')
    sList = scoreList.split(',')
```

```
    sList = [float(x) for x in sList]

    resDict= dict(zip(assessTypeList,sList ))
    tempDict[subject] = resDict



    return tempDict



def avgScoreComput(studname, sDictKeys):
    studName = studname
    qContrib = ''
    hContrib = ''
    aContrib = ''
    eContrib = ''
    ScoreList = []
    grade = ''
    gpa = ''
    status = ''

    for rec in list(sDictKeys):
        subject = list(rec.keys())[0]
        subjScores = list(rec.values())[0]
        #print(subject, subjScores)
        #print(subjScores.keys())
        #print(subjScores['Quiz'])
        for sub in list(subjScores.keys()):
            #print(sub)
            if sub=='Quiz':
                qContrib = (float(subjScores[sub])/190)*30
                #print(qContrib)
            elif sub == 'HW':
                hContrib = (float(subjScores[sub])/190)*15

            elif sub == 'ATTND':
                aContrib = (float(subjScores[sub])/96)*12


            elif sub == 'Exam':
                eContrib = (float(subjScores[sub])/100)*43
            else:
                print('assessment type unkown')
        avgScore =   qContrib +hContrib +aContrib + eContrib
        ScoreList.append(avgScore)
        print(avgScore)
        Score = statistics.mean(ScoreList)
        Score = round(Score, 2)

    print(f'this is the mean score of the student: {Score}')
    gpa = (Score/100)*5
    if Score > 89.9:
        grade = 'A'
    elif Score > 74.9:
        grade = 'B'
    elif Score > 64.9:
        grade = 'C'
    elif Score > 54.9:
        grade = 'D'
```

```
    elif Score > 49.9:
        grade = 'E'
    else:
        grade = 'F'

    if grade in ['A','B','C','D']:
        status = 'Pass'
    elif grade in ['E']:
        status = 'Retake'
    else:
        status = 'Fail'
    return ScoreList, Score, grade, gpa, status
```

In [532…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namewellington zuba
student IDGR-0502
subjectchem
enter scores in the following order; Quiz, HW, ATTND, Exam182, 130, 92, 47
set flag1
[{'chem': {'Quiz': 182.0, 'HW': 130.0, 'ATTND': 92.0, 'Exam': 47.0}}]
{}
```

In [533…

```
# automating inputs

name = input('student name')
```

```
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namewellington zuba
student IDGR-0502
subjectphy
enter scores in the following order; Quiz, HW, ATTND, Exam71, 28, 88, 19
set flag1
[{'chem': {'Quiz': 182.0, 'HW': 130.0, 'ATTND': 92.0, 'Exam': 47.0}}, {'phy': {'Qui
z': 71.0, 'HW': 28.0, 'ATTND': 88.0, 'Exam': 19.0}}]
{}
```

In [534…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)
```

```
    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namewellington zuba
student IDGR-0502
subjectbio
enter scores in the following order; Quiz, HW, ATTND, Exam102, 70, 19, 38
set flag1
[{'chem': {'Quiz': 182.0, 'HW': 130.0, 'ATTND': 92.0, 'Exam': 47.0}}, {'phy': {'Qui
z': 71.0, 'HW': 28.0, 'ATTND': 88.0, 'Exam': 19.0}}, {'bio': {'Quiz': 102.0, 'HW': 7
0.0, 'ATTND': 19.0, 'Exam': 38.0}}]
{}
```

In [535…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []
```

```
print(perfList)
print(studSubsDict)
```

```
student namewellington zuba
student IDGR-0502
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam60, 77, 94, 31
set flag1
[{'chem': {'Quiz': 182.0, 'HW': 130.0, 'ATTND': 92.0, 'Exam': 47.0}}, {'phy': {'Qui
z': 71.0, 'HW': 28.0, 'ATTND': 88.0, 'Exam': 19.0}}, {'bio': {'Quiz': 102.0, 'HW': 7
0.0, 'ATTND': 19.0, 'Exam': 38.0}}, {'math': {'Quiz': 60.0, 'HW': 77.0, 'ATTND': 94.
0, 'Exam': 31.0}}]
{}
```

In [536…

```
# automating inputs

name = input('student name')
ID = str(input('student ID'))
subject = input('subject')
assessTypeList = ['Quiz', 'HW', 'ATTND', 'Exam'] #input('assessment type List')
scoresList = input('enter scores in the following order; Quiz, HW, ATTND, Exam')
flag = int(input('set flag'))
 # when set to 1, code aggregates subjectwise score. when set to 2 it compust full rec

if flag == 1:
    tempDict = scoreAggregator(name, ID, subject, assessTypeList, scoresList)
    perfList.append(tempDict)
    #print(tempDict)
    #print(perfList)
else:
    name_ID = name + '_'+ str(ID)
    res = avgScoreComput(name_ID, perfList)

    print(res)
    scorecard = dict()
    ScoreList, Score, grade, gpa, status = res
    scorecard['Subject Scores'] = ScoreList
    scorecard['Overall Score'] = Score
    scorecard['Grade'] = grade
    scorecard['GPA'] = gpa
    scorecard['Status'] = status

    perfList.append(scorecard)

    studSubsDict[name_ID]=perfList
    perfList = []

print(perfList)
print(studSubsDict)
```

```
student namewellington zuba
student IDGR-0502
subjectmath
enter scores in the following order; Quiz, HW, ATTND, Exam60, 77, 94, 31
set flag2
70.71
32.59105263157895
40.34657894736842
40.63263157894737
this is the mean score of the student: 46.07
([70.71, 32.59105263157895, 40.34657894736842, 40.63263157894737], 46.07, 'F', 2.303
5, 'Fail')
[]
{'wellington zuba_GR-0502': [{'chem': {'Quiz': 182.0, 'HW': 130.0, 'ATTND': 92.0, 'Ex
am': 47.0}}, {'phy': {'Quiz': 71.0, 'HW': 28.0, 'ATTND': 88.0, 'Exam': 19.0}}, {'bi
o': {'Quiz': 102.0, 'HW': 70.0, 'ATTND': 19.0, 'Exam': 38.0}}, {'math': {'Quiz': 60.
0, 'HW': 77.0, 'ATTND': 94.0, 'Exam': 31.0}}, {'Subject Scores': [70.71, 32.591052631
57895, 40.34657894736842, 40.63263157894737], 'Overall Score': 46.07, 'Grade': 'F',
'GPA': 2.3035, 'Status': 'Fail'}]}
```

In [537…
```
res
```

Out[537]:
```
([70.71, 32.59105263157895, 40.34657894736842, 40.63263157894737],
 46.07,
 'F',
 2.3035,
 'Fail')
```

In [539…
```
studSubsDict['wellington zuba_GR-0502']
```

Out[539]:
```
[{'chem': {'Quiz': 182.0, 'HW': 130.0, 'ATTND': 92.0, 'Exam': 47.0}},
 {'phy': {'Quiz': 71.0, 'HW': 28.0, 'ATTND': 88.0, 'Exam': 19.0}},
 {'bio': {'Quiz': 102.0, 'HW': 70.0, 'ATTND': 19.0, 'Exam': 38.0}},
 {'math': {'Quiz': 60.0, 'HW': 77.0, 'ATTND': 94.0, 'Exam': 31.0}},
 {'Subject Scores': [70.71,
   32.59105263157895,
   40.34657894736842,
   40.63263157894737],
  'Overall Score': 46.07,
  'Grade': 'F',
  'GPA': 2.3035,
  'Status': 'Fail'}]
```

In [ ]:
```
```

In [541…
```
# Function that saves a file to disk

def fileSaver(dataframe):
    pass


def dataFrameCreator(listset, colNamesSet):
    pass
```

In [543…
```
#All float scores of subject scores and average score rounded off to 2d.p
#while GPA rounded up to 1d.p
```

In [546…
```
import pandas as pd
```

```
studName = ['bett james', 'namukolo abrams', 'vera abutu', 'kwame doga', 'lukman ahmad
         'luke brant', 'james kenyata', 'ngugi tionga', 'okoro eze', 'agatha chiluba',
        'longe jethro', 'florence giwa', 'veiva lucent', 'melody braimoh', 'victor iha
        'maguel peter', 'wellington zuba']

studID = ['GR-0483', 'GR-0484', 'GR-0485', 'GR-0486', 'GR-0487', 'GR-0488', 'GR-0489',
         'GR-0491', 'GR-0492', 'GR-0493', 'GR-0494', 'GR-0495', 'GR-0496', 'GR-0497',
         'GR-0499', 'GR-0500', 'GR-0501', 'GR-0502']

chemQuizScore = [127, 141, 51, 26, 29, 99, 171, 51, 155, 147, 187, 123, 179, 98, 17, 6
bioQuizScore = [184, 177, 112, 133, 21, 90, 117, 188, 17, 61, 149, 127, 120, 42, 177,
phyQuizScore = [52, 21, 61, 62, 177, 0, 151, 41, 12, 0, 117, 48, 119, 122, 114, 145, 1
mathQuizScore = [133, 136, 74, 86, 50, 109, 121, 120, 51, 79, 133, 83, 155, 84, 60, 10

chemHwScore = [135, 82, 102, 166, 180, 125, 129, 103, 111, 153, 106, 171, 160, 172, 60
bioHwScore = [186, 170, 55, 108, 123, 47, 47, 156, 136, 66, 100, 88, 136, 19, 70, 125,
phyHwScore = [142, 12, 105, 59, 11, 119, 63, 31, 77, 97, 88, 139, 119, 20, 109, 65, 6,
mathHwScore = [97, 180, 50, 181, 18, 144, 16, 188, 145, 149, 79, 33, 97, 157, 102, 58,

chemAttndScore = [17, 95, 96, 75, 38, 26, 91, 39, 14, 31, 67, 82, 48, 30, 48, 68, 83,
bioAttndScore = [58, 57, 71, 52, 47, 88, 51, 13, 42, 58, 64, 87, 92, 36, 48, 38, 51, 4
phyAttndScore = [29, 43, 83, 94, 38, 37, 93, 90, 70, 54, 86, 63, 65, 24, 84, 43, 50, 7
mathAttndScore = [34, 48, 93, 66, 34, 14, 22, 25, 36, 71, 55, 13, 50, 18, 73, 36, 63,

chemExamScore = [46, 52, 24, 73, 80, 83, 70, 98, 96, 48, 83, 48, 93, 94, 36, 39, 75, 4
bioExamScore = [97, 41, 51, 68, 90, 18, 25, 72, 27, 50, 80, 23, 89, 84, 34, 36, 35, 80
phyExamScore = [73, 31, 98, 65, 51, 44, 19, 18, 43, 75, 61, 35, 70, 68, 31, 14, 33, 23
mathExamScore = [45, 31, 94, 67, 68, 65, 41, 52, 22, 76, 70, 10, 81, 46, 87, 89, 48, 5

chemAvgScore = [52.26, 62.97, 38.43, 57.98, 57.94, 64.44, 78.66, 63.20, 76.27, 59.80,
                86.88, 73.22, 28.90, 49.82, 71.68, 67.52, 30.62, 70.71]
bioAvgScore = [92.70, 66.12, 52.83,  65.27, 57.60, 36.66, 39.31, 74.59, 30.28, 43.59,
                79.45, 48.75, 54.09, 55.68,  53.00, 73.31, 46.54, 40.35]
phyAvgScore = [54.44, 22.97, 70.44, 54.15, 55.50, 32.94,  48.61, 27.91, 35.21, 46.66,
               66.41, 53.08, 50.44, 39.42, 42.23, 32.59, 40.96, 32.59]
mathAvgScore = [52.26, 55.01, 67.68, 64.93, 42.81, 58.28, 40.75, 59.27, 33.46, 65.79,
                 73.21, 47.69, 64.06, 63.77, 38.78, 48.66, 49.98, 40.63]
overAllScore = [63.00, 51.77, 57.34, 60.58, 53.46, 48.08, 51.83, 56.24, 43.81, 53.96,
                76.49, 55.69, 49.37, 52.17, 51.42, 55.52, 42.02, 46.07]

GPA = [3.2, 2.6, 2.9, 3.0, 2.7, 2.4, 2.6, 2.8, 2.2, 2.7, 3.5, 2.2, 3.8, 2.8, 2.5, 2.6,

gradeType = ['D', 'E', 'D', 'D', 'E', 'F', 'E', 'D', 'F', 'E', 'C', 'F', 'B', 'D', 'F'

gradeStatus = [' Pass', 'Retake', ' Pass', ' Pass', 'Retake', 'Fail', 'Retake', 'Pass'
               'Fail', 'Pass', 'Pass', ' Fail', 'Retake', 'Retake', 'Pass', 'Fail', 'Fa


def fileSaver(dataframe):
    pass


def dataFrameCreator(listset, colNamesSet):
    dataTable = pd.DataFrame()

    dataTable[colNamesSet[0]] = listset[0]
    dataTable[colNamesSet[1]] =  listset[1]
    dataTable[colNamesSet[2]] =  listset[2]
    dataTable[colNamesSet[3]] =  listset[3]
    dataTable[colNamesSet[4]] =  listset[4]
```

```
    dataTable[colNamesSet[5]] = listset[5]
    dataTable[colNamesSet[6]] =  listset[6]
    dataTable[colNamesSet[7]] =  listset[7]
    dataTable[colNamesSet[8]] =  listset[8]
    dataTable[colNamesSet[9]] =  listset[9]
    dataTable[colNamesSet[10]] = listset[10]
    dataTable[colNamesSet[11]] =  listset[11]
    dataTable[colNamesSet[12]] =  listset[12]
    dataTable[colNamesSet[13]] =  listset[13]
    dataTable[colNamesSet[14]] =  listset[14]
    dataTable[colNamesSet[15]] = listset[15]
    dataTable[colNamesSet[16]] =  listset[16]
    dataTable[colNamesSet[17]] = listset[17]
    dataTable[colNamesSet[18]] =  listset[18]
    dataTable[colNamesSet[19]] =  listset[19]
    dataTable[colNamesSet[20]] = listset[20]
    dataTable[colNamesSet[21]] =  listset[21]
    dataTable[colNamesSet[22]] =  listset[22]
    dataTable[colNamesSet[23]] =  listset[23]
    dataTable[colNamesSet[24]] =  listset[24]
    dataTable[colNamesSet[25]] =  listset[25]


    print(dataTable)


# pass the lists and the column names to our dataFrameCreator function
tableVals = [studName, studID, chemQuizScore, bioQuizScore, phyQuizScore, mathQuizScor
             phyHwScore, mathHwScore, chemAttndScore, bioAttndScore, phyAttndScore, ma
             bioExamScore, phyExamScore, mathExamScore, chemAvgScore, bioAvgScore, phy
             mathAvgScore, overAllScore, GPA, gradeType, gradeStatus]

colNames = ['Name', 'ID', 'Chem Quiz', 'Bio Quiz', 'Phy Quiz', 'Math Quiz', 'Chem HW',
            'Math HW', 'Chem Attnd', 'Bio Attnd', 'Phy Attnd', 'Math Attnd', 'Chem Exa
            'Phy Exam', 'Math Exam', 'Chem Avg', 'Bio Avg', 'Phy Avg', 'Math Avg', 'Av

dataFrameCreator(tableVals, colNames)  # call function to create table
```

	Name	ID	Chem Quiz	Bio Quiz	Phy Quiz	Math Quiz	\
0	bett james	GR-0483	127	184	52	133	
1	namukolo abrams	GR-0484	141	177	21	136	
2	vera abutu	GR-0485	51	112	61	74	
3	kwame doga	GR-0486	26	133	62	86	
4	lukman ahmad	GR-0487	29	21	177	50	
5	akin torey	GR-0488	99	90	0	109	
6	luke brant	GR-0489	171	117	151	121	
7	james kenyata	GR-0490	51	188	41	120	
8	ngugi tionga	GR-0491	155	17	12	51	
9	okoro eze	GR-0492	147	61	0	79	
10	agatha chiluba	GR-0493	187	149	117	133	
11	mangu joseph	GR-0494	123	127	48	83	
12	longe jethro	GR-0495	179	120	119	155	
13	florence giwa	GR-0496	98	42	122	84	
14	veiva lucent	GR-0497	17	177	114	60	
15	melody braimoh	GR-0498	67	162	145	104	
16	victor ihab	GR-0499	177	155	135	43	
17	mim trucker	GR-0500	148	141	34	61	
18	maguel peter	GR-0501	23	183	81	33	
19	wellington zuba	GR-0502	182	102	71	60	

	Chem HW	Bio HW	Phy HW	Math HW	...	Phy Exam	Math Exam	Chem Avg	\
0	135	186	142	97	...	73	45	52.26	
1	82	170	12	180	...	31	31	62.97	
2	102	55	105	50	...	98	94	38.43	
3	166	108	59	181	...	65	67	57.98	
4	180	123	11	18	...	51	68	57.94	
5	125	47	119	144	...	44	65	64.44	
6	129	47	63	16	...	19	41	78.66	
7	103	156	31	188	...	18	52	63.20	
8	111	136	77	145	...	43	22	76.27	
9	153	66	97	149	...	75	76	59.80	
10	106	100	88	79	...	61	70	81.96	
11	171	88	139	33	...	35	10	63.81	
12	160	136	119	97	...	70	81	86.88	
13	172	19	20	157	...	68	46	73.22	
14	60	70	109	102	...	31	87	28.90	
15	177	125	65	58	...	14	89	49.82	
16	14	90	6	44	...	33	48	71.68	
17	182	146	104	136	...	23	53	67.52	
18	95	43	143	79	...	16	62	30.62	
19	130	70	28	77	...	19	31	70.71	

	Bio Avg	Phy Avg	Math Avg	Avg Score	GPA	Grade	Status
0	92.70	54.44	52.26	63.00	3.2	D	Pass
1	66.12	22.97	55.01	51.77	2.6	E	Retake
2	52.83	70.44	67.68	57.34	2.9	D	Pass
3	65.27	54.15	64.93	60.58	3.0	D	Pass
4	57.60	55.50	42.81	53.46	2.7	E	Retake
5	36.66	32.94	58.28	48.08	2.4	F	Fail
6	39.31	48.61	40.75	51.83	2.6	E	Retake
7	74.59	27.91	59.27	56.24	2.8	D	Pass
8	30.28	35.21	33.46	43.81	2.2	F	Fail
9	43.59	46.66	65.79	53.96	2.7	E	Retake
10	73.82	62.40	64.21	70.60	3.5	C	Pass
11	47.77	41.48	21.64	43.67	2.2	F	Fail
12	79.45	66.41	73.21	76.49	3.8	B	Pass
13	48.75	53.08	47.69	55.69	2.8	D	Pass
14	54.09	50.44	64.06	49.37	2.5	F	Fail

```
15    55.68    39.42    63.77    52.17  2.6     E   Retake
16    53.00    42.23    38.78    51.42  2.6     E   Retake
17    73.31    32.59    48.66    55.52  2.8     D     Pass
18    46.54    40.96    49.98    42.02  2.1     F     Fail
19    40.35    32.59    40.63    46.07  2.3     F     Fail

[20 rows x 26 columns]
```

In [549…

```
# creating an datatable

import pandas as pd

dataTable = pd.DataFrame()
dataTable['Name'] = studName
dataTable['ID'] = studID
dataTable['Chem Quiz'] = chemQuizScore
dataTable['Bio Quiz'] = bioQuizScore
dataTable['Phy Quiz'] = phyQuizScore
dataTable['Math Quiz'] = mathQuizScore
dataTable['Chem HW'] = chemHwScore
dataTable['Bio HW'] = bioHwScore
dataTable['Phy HW'] = phyHwScore
dataTable['Math HW'] = mathHwScore
dataTable['Chem Attnd'] = chemAttndScore
dataTable['Bio Attnd'] = bioAttndScore
dataTable['Phy Attnd'] = phyAttndScore
dataTable['Math Attnd'] = mathAttndScore
dataTable['Chem Exam'] = chemExamScore
dataTable['Bio Exam'] = bioExamScore
dataTable['Phy Exam'] = phyExamScore
dataTable['Math Exam'] = mathExamScore
dataTable['Chem Avg'] = chemAvgScore
dataTable['Bio Avg'] = bioAvgScore
dataTable['Phy Avg'] = phyAvgScore
dataTable['Math Avg'] = mathAvgScore
dataTable['Avg Score'] = overAllScore
dataTable['GPA'] = GPA
dataTable['Grade'] = gradeType
dataTable['Status'] = gradeStatus

dataTable
```

Out[549]:

	Name	ID	Chem Quiz	Bio Quiz	Phy Quiz	Math Quiz	Chem HW	Bio HW	Phy HW	Math HW	...	Phy Exam	Math Exam	Chem Avg	
0	bett james	GR-0483	127	184	52	133	135	186	142	97	...	73	45	52.26	9
1	namukolo abrams	GR-0484	141	177	21	136	82	170	12	180	...	31	31	62.97	6
2	vera abutu	GR-0485	51	112	61	74	102	55	105	50	...	98	94	38.43	5
3	kwame doga	GR-0486	26	133	62	86	166	108	59	181	...	65	67	57.98	6
4	lukman ahmad	GR-0487	29	21	177	50	180	123	11	18	...	51	68	57.94	5
5	akin torey	GR-0488	99	90	0	109	125	47	119	144	...	44	65	64.44	3
6	luke brant	GR-0489	171	117	151	121	129	47	63	16	...	19	41	78.66	3
7	james kenyata	GR-0490	51	188	41	120	103	156	31	188	...	18	52	63.20	7
8	ngugi tionga	GR-0491	155	17	12	51	111	136	77	145	...	43	22	76.27	3
9	okoro eze	GR-0492	147	61	0	79	153	66	97	149	...	75	76	59.80	4
10	agatha chiluba	GR-0493	187	149	117	133	106	100	88	79	...	61	70	81.96	7
11	mangu joseph	GR-0494	123	127	48	83	171	88	139	33	...	35	10	63.81	4
12	longe jethro	GR-0495	179	120	119	155	160	136	119	97	...	70	81	86.88	7
13	florence giwa	GR-0496	98	42	122	84	172	19	20	157	...	68	46	73.22	4
14	veiva lucent	GR-0497	17	177	114	60	60	70	109	102	...	31	87	28.90	5
15	melody braimoh	GR-0498	67	162	145	104	177	125	65	58	...	14	89	49.82	5
16	victor ihab	GR-0499	177	155	135	43	14	90	6	44	...	33	48	71.68	5
17	mim trucker	GR-0500	148	141	34	61	182	146	104	136	...	23	53	67.52	7
18	maguel peter	GR-0501	23	183	81	33	95	43	143	79	...	16	62	30.62	4
19	wellington zuba	GR-0502	182	102	71	60	130	70	28	77	...	19	31	70.71	4

20 rows × 26 columns

In [550…
```
dataTable.to_excel('c:/filestore1/philoTable.xls')
```

```
<ipython-input-550-8f3a1dd87783>:1: FutureWarning: As the xlwt package is no longer m
aintained, the xlwt engine will be removed in a future version of pandas. This is the
only engine in pandas that supports writing in the xls format. Install openpyxl and w
rite to an xlsx file instead. You can set the option io.excel.xls.writer to 'xlwt' to
silence this warning. While this option is deprecated and will also raise a warning,
it can be globally set and the warning suppressed.
  dataTable.to_excel('c:/filestore1/philoTable.xls')
```

In [551…
```
import pandas as pd

studName = ['bett james', 'namukolo abrams', 'vera abutu', 'kwame doga', 'lukman ahmad
         'luke brant', 'james kenyata', 'ngugi tionga', 'okoro eze', 'agatha chiluba',
        'longe jethro', 'florence giwa', 'veiva lucent', 'melody braimoh', 'victor iha
        'maguel peter', 'wellington zuba']

studID = ['GR-0483', 'GR-0484', 'GR-0485', 'GR-0486', 'GR-0487', 'GR-0488', 'GR-0489',
         'GR-0491', 'GR-0492', 'GR-0493', 'GR-0494', 'GR-0495', 'GR-0496', 'GR-0497',
         'GR-0499', 'GR-0500', 'GR-0501', 'GR-0502']

chemQuizScore = [127, 141, 51, 26, 29, 99, 171, 51, 155, 147, 187, 123, 179, 98, 17, 6
bioQuizScore = [184, 177, 112, 133, 21, 90, 117, 188, 17, 61, 149, 127, 120, 42, 177,
phyQuizScore = [52, 21, 61, 62, 177, 0, 151, 41, 12, 0, 117, 48, 119, 122, 114, 145, 1
mathQuizScore = [133, 136, 74, 86, 50, 109, 121, 120, 51, 79, 133, 83, 155, 84, 60, 10

chemHwScore = [135, 82, 102, 166, 180, 125, 129, 103, 111, 153, 106, 171, 160, 172, 60
bioHwScore = [186, 170, 55, 108, 123, 47, 47, 156, 136, 66, 100, 88, 136, 19, 70, 125,
phyHwScore = [142, 12, 105, 59, 11, 119, 63, 31, 77, 97, 88, 139, 119, 20, 109, 65, 6,
mathHwScore = [97, 180, 50, 181, 18, 144, 16, 188, 145, 149, 79, 33, 97, 157, 102, 58,

chemAttndScore = [17, 95, 96, 75, 38, 26, 91, 39, 14, 31, 67, 82, 48, 30, 48, 68, 83,
bioAttndScore = [58, 57, 71, 52, 47, 88, 51, 13, 42, 58, 64, 87, 92, 36, 48, 38, 51, 4
phyAttndScore = [29, 43, 83, 94, 38, 37, 93, 90, 70, 54, 86, 63, 65, 24, 84, 43, 50, 7
mathAttndScore = [34, 48, 93, 66, 34, 14, 22, 25, 36, 71, 55, 13, 50, 18, 73, 36, 63,

chemExamScore = [46, 52, 24, 73, 80, 83, 70, 98, 96, 48, 83, 48, 93, 94, 36, 39, 75, 4
bioExamScore = [97, 41, 51, 68, 90, 18, 25, 72, 27, 50, 80, 23, 89, 84, 34, 36, 35, 80
phyExamScore = [73, 31, 98, 65, 51, 44, 19, 18, 43, 75, 61, 35, 70, 68, 31, 14, 33, 23
mathExamScore = [45, 31, 94, 67, 68, 65, 41, 52, 22, 76, 70, 10, 81, 46, 87, 89, 48, 5

chemAvgScore = [52.26, 62.97, 38.43, 57.98, 57.94, 64.44, 78.66, 63.20, 76.27, 59.80,
                86.88, 73.22, 28.90, 49.82, 71.68, 67.52, 30.62, 70.71]
bioAvgScore = [92.70, 66.12, 52.83,  65.27, 57.60, 36.66, 39.31, 74.59, 30.28, 43.59,
                79.45, 48.75, 54.09, 55.68,  53.00, 73.31, 46.54, 40.35]
phyAvgScore = [54.44, 22.97, 70.44, 54.15, 55.50, 32.94,  48.61, 27.91, 35.21, 46.66,
               66.41, 53.08, 50.44, 39.42, 42.23, 32.59, 40.96, 32.59]
mathAvgScore = [52.26, 55.01, 67.68, 64.93, 42.81, 58.28, 40.75, 59.27, 33.46, 65.79,
                 73.21, 47.69, 64.06, 63.77, 38.78, 48.66, 49.98, 40.63]
overAllScore = [63.00, 51.77, 57.34, 60.58, 53.46, 48.08, 51.83, 56.24, 43.81, 53.96,
                76.49, 55.69, 49.37, 52.17, 51.42, 55.52, 42.02, 46.07]

GPA = [3.2, 2.6, 2.9, 3.0, 2.7, 2.4, 2.6, 2.8, 2.2, 2.7, 3.5, 2.2, 3.8, 2.8, 2.5, 2.6,

gradeType = ['D', 'E', 'D', 'D', 'E', 'F', 'E', 'D', 'F', 'E', 'C', 'F', 'B', 'D', 'F'

gradeStatus = [' Pass', 'Retake', ' Pass', ' Pass', 'Retake', 'Fail', 'Retake', 'Pass'
               'Fail', 'Pass', 'Pass', ' Fail', 'Retake', 'Retake', 'Pass', 'Fail', 'Fa
```

```
# Function that saves a file to disk

def fileSaver(dataframe):
    print('Starting saving of file')
    '''to save a file to disk, we need a name for the file,
        nd a directory or path to the folder where ot will be dsaved'''
    saveName = 'dataTable.xls'
    savePath = 'c:/filestore1/'

    # Saving the file
    #SYNTAX
    # nameOfTable.to_excel(FolderPath + fileNAME)
    dataframe.to_excel(savePath+saveName)
    print('finished saving file')

def dataFrameCreator(listset, colNamesSet):
    dataTable = pd.DataFrame()

    dataTable[colNamesSet[0]] = listset[0]
    dataTable[colNamesSet[1]] =  listset[1]
    dataTable[colNamesSet[2]] =  listset[2]
    dataTable[colNamesSet[3]] =  listset[3]
    dataTable[colNamesSet[4]] =  listset[4]
    dataTable[colNamesSet[5]] = listset[5]
    dataTable[colNamesSet[6]] =  listset[6]
    dataTable[colNamesSet[7]] =  listset[7]
    dataTable[colNamesSet[8]] =  listset[8]
    dataTable[colNamesSet[9]] =  listset[9]
    dataTable[colNamesSet[10]] = listset[10]
    dataTable[colNamesSet[11]] =  listset[11]
    dataTable[colNamesSet[12]] =  listset[12]
    dataTable[colNamesSet[13]] =  listset[13]
    dataTable[colNamesSet[14]] =  listset[14]
    dataTable[colNamesSet[15]] = listset[15]
    dataTable[colNamesSet[16]] =  listset[16]
    dataTable[colNamesSet[17]] = listset[17]
    dataTable[colNamesSet[18]] =  listset[18]
    dataTable[colNamesSet[19]] =  listset[19]
    dataTable[colNamesSet[20]] = listset[20]
    dataTable[colNamesSet[21]] =  listset[21]
    dataTable[colNamesSet[22]] =  listset[22]
    dataTable[colNamesSet[23]] =  listset[23]
    dataTable[colNamesSet[24]] =  listset[24]
    dataTable[colNamesSet[25]] =  listset[25]

    print(dataTable)
    print('calling function to save file')
    fileSaver(dataTable)



# pass the lists and the column names to our dataFrameCreator function
tableVals = [studName, studID, chemQuizScore, bioQuizScore, phyQuizScore, mathQuizScor
             phyHwScore, mathHwScore, chemAttndScore, bioAttndScore, phyAttndScore, ma
             bioExamScore, phyExamScore, mathExamScore, chemAvgScore, bioAvgScore, phy
             mathAvgScore, overAllScore, GPA, gradeType, gradeStatus]

colNames = ['Name', 'ID', 'Chem Quiz', 'Bio Quiz', 'Phy Quiz', 'Math Quiz', 'Chem HW',
```

```
            'Math HW', 'Chem Attnd', 'Bio Attnd', 'Phy Attnd', 'Math Attnd', 'Chem Exa
            'Phy Exam', 'Math Exam', 'Chem Avg', 'Bio Avg', 'Phy Avg', 'Math Avg', 'Av

dataFrameCreator(tableVals, colNames)
```

	Name	ID	Chem Quiz	Bio Quiz	Phy Quiz	Math Quiz	\
0	bett james	GR-0483	127	184	52	133	
1	namukolo abrams	GR-0484	141	177	21	136	
2	vera abutu	GR-0485	51	112	61	74	
3	kwame doga	GR-0486	26	133	62	86	
4	lukman ahmad	GR-0487	29	21	177	50	
5	akin torey	GR-0488	99	90	0	109	
6	luke brant	GR-0489	171	117	151	121	
7	james kenyata	GR-0490	51	188	41	120	
8	ngugi tionga	GR-0491	155	17	12	51	
9	okoro eze	GR-0492	147	61	0	79	
10	agatha chiluba	GR-0493	187	149	117	133	
11	mangu joseph	GR-0494	123	127	48	83	
12	longe jethro	GR-0495	179	120	119	155	
13	florence giwa	GR-0496	98	42	122	84	
14	veiva lucent	GR-0497	17	177	114	60	
15	melody braimoh	GR-0498	67	162	145	104	
16	victor ihab	GR-0499	177	155	135	43	
17	mim trucker	GR-0500	148	141	34	61	
18	maguel peter	GR-0501	23	183	81	33	
19	wellington zuba	GR-0502	182	102	71	60	

	Chem HW	Bio HW	Phy HW	Math HW	...	Phy Exam	Math Exam	Chem Avg	\
0	135	186	142	97	...	73	45	52.26	
1	82	170	12	180	...	31	31	62.97	
2	102	55	105	50	...	98	94	38.43	
3	166	108	59	181	...	65	67	57.98	
4	180	123	11	18	...	51	68	57.94	
5	125	47	119	144	...	44	65	64.44	
6	129	47	63	16	...	19	41	78.66	
7	103	156	31	188	...	18	52	63.20	
8	111	136	77	145	...	43	22	76.27	
9	153	66	97	149	...	75	76	59.80	
10	106	100	88	79	...	61	70	81.96	
11	171	88	139	33	...	35	10	63.81	
12	160	136	119	97	...	70	81	86.88	
13	172	19	20	157	...	68	46	73.22	
14	60	70	109	102	...	31	87	28.90	
15	177	125	65	58	...	14	89	49.82	
16	14	90	6	44	...	33	48	71.68	
17	182	146	104	136	...	23	53	67.52	
18	95	43	143	79	...	16	62	30.62	
19	130	70	28	77	...	19	31	70.71	

	Bio Avg	Phy Avg	Math Avg	Avg Score	GPA	Grade	Status
0	92.70	54.44	52.26	63.00	3.2	D	Pass
1	66.12	22.97	55.01	51.77	2.6	E	Retake
2	52.83	70.44	67.68	57.34	2.9	D	Pass
3	65.27	54.15	64.93	60.58	3.0	D	Pass
4	57.60	55.50	42.81	53.46	2.7	E	Retake
5	36.66	32.94	58.28	48.08	2.4	F	Fail
6	39.31	48.61	40.75	51.83	2.6	E	Retake
7	74.59	27.91	59.27	56.24	2.8	D	Pass
8	30.28	35.21	33.46	43.81	2.2	F	Fail
9	43.59	46.66	65.79	53.96	2.7	E	Retake
10	73.82	62.40	64.21	70.60	3.5	C	Pass
11	47.77	41.48	21.64	43.67	2.2	F	Fail
12	79.45	66.41	73.21	76.49	3.8	B	Pass
13	48.75	53.08	47.69	55.69	2.8	D	Pass
14	54.09	50.44	64.06	49.37	2.5	F	Fail

```
15    55.68    39.42    63.77    52.17  2.6     E   Retake
16    53.00    42.23    38.78    51.42  2.6     E   Retake
17    73.31    32.59    48.66    55.52  2.8     D     Pass
18    46.54    40.96    49.98    42.02  2.1     F     Fail
19    40.35    32.59    40.63    46.07  2.3     F     Fail

[20 rows x 26 columns]
calling function to save file
Starting saving of file
finished saving file
```

```
<ipython-input-551-9c02749e7d4d>:63: FutureWarning: As the xlwt package is no longer
maintained, the xlwt engine will be removed in a future version of pandas. This is th
e only engine in pandas that supports writing in the xls format. Install openpyxl and
write to an xlsx file instead. You can set the option io.excel.xls.writer to 'xlwt' t
o silence this warning. While this option is deprecated and will also raise a warnin
g, it can be globally set and the warning suppressed.
  dataframe.to_excel(savePath+saveName)
```

In [ ]:

In [552…

```
# Reading back a file from disk

# read_csv()  to read a csv file from our disk
# read_excel() to read an excel file from disk

'''
to read a file from disk, we need the path where the file is saved
we also need the file name and its extension
'''

filename = 'dataTable.xls'
dirPath = 'c:/filestore1/'

file = pd.read_excel(dirPath+filename)

file
```

Out[552]:

	Unnamed: 0	Name	ID	Chem Quiz	Bio Quiz	Phy Quiz	Math Quiz	Chem HW	Bio HW	Phy HW	...	Phy Exam	Math Exam	Chem Av
0	0	bett james	GR-0483	127	184	52	133	135	186	142	...	73	45	52.2
1	1	namukolo abrams	GR-0484	141	177	21	136	82	170	12	...	31	31	62.9
2	2	vera abutu	GR-0485	51	112	61	74	102	55	105	...	98	94	38.4
3	3	kwame doga	GR-0486	26	133	62	86	166	108	59	...	65	67	57.9
4	4	lukman ahmad	GR-0487	29	21	177	50	180	123	11	...	51	68	57.9
5	5	akin torey	GR-0488	99	90	0	109	125	47	119	...	44	65	64.4
6	6	luke brant	GR-0489	171	117	151	121	129	47	63	...	19	41	78.6
7	7	james kenyata	GR-0490	51	188	41	120	103	156	31	...	18	52	63.2
8	8	ngugi tionga	GR-0491	155	17	12	51	111	136	77	...	43	22	76.2
9	9	okoro eze	GR-0492	147	61	0	79	153	66	97	...	75	76	59.8
10	10	agatha chiluba	GR-0493	187	149	117	133	106	100	88	...	61	70	81.9
11	11	mangu joseph	GR-0494	123	127	48	83	171	88	139	...	35	10	63.8
12	12	longe jethro	GR-0495	179	120	119	155	160	136	119	...	70	81	86.8
13	13	florence giwa	GR-0496	98	42	122	84	172	19	20	...	68	46	73.2
14	14	veiva lucent	GR-0497	17	177	114	60	60	70	109	...	31	87	28.9
15	15	melody braimoh	GR-0498	67	162	145	104	177	125	65	...	14	89	49.8
16	16	victor ihab	GR-0499	177	155	135	43	14	90	6	...	33	48	71.6
17	17	mim trucker	GR-0500	148	141	34	61	182	146	104	...	23	53	67.5
18	18	maguel peter	GR-0501	23	183	81	33	95	43	143	...	16	62	30.6
19	19	wellington zuba	GR-0502	182	102	71	60	130	70	28	...	19	31	70.7

20 rows × 27 columns

In [ ]:

In [553…
```
def dataFrameCreator(param1, param2):
    dataTable = pd.DataFrame()

    dataTable[param2[0]] = param1[0]
    dataTable[param2[1]] =  param1[1]
    dataTable[param2[2]] =  param1[2]
    dataTable[param2[3]] =  param1[3]
    dataTable[param2[4]] =  param1[4]
    dataTable[param2[5]] = param1[5]
    dataTable[param2[6]] =  param1[6]
    dataTable[param2[7]] =  param1[7]
    dataTable[param2[8]] =  param1[8]
    dataTable[param2[9]] =  param1[9]
    dataTable[param2[10]] = param1[10]
    dataTable[param2[11]] =  param1[11]
    dataTable[param2[12]] =  param1[12]
    dataTable[param2[13]] =  param1[13]
    dataTable[param2[14]] =  param1[14]
    dataTable[param2[15]] = param1[15]
    dataTable[param2[16]] =  param1[16]
    dataTable[param2[17]] =  param1[17]
    dataTable[param2[18]] =  param1[18]
    dataTable[param2[19]] =  param1[19]
    dataTable[param2[20]] = param1[20]
    dataTable[param2[21]] =  param1[21]
    dataTable[param2[22]] =  param1[22]
    dataTable[param2[23]] =  param1[23]
    dataTable[param2[24]] =  param1[24]
    dataTable[param2[25]] =  param1[25]


    print(dataTable)
    print('calling function to save file')


# pass the lists and the column names to our dataFrameCreator function
recVals = [studName, studID, chemQuizScore, bioQuizScore, phyQuizScore, mathQuizScore,
             phyHwScore, mathHwScore, chemAttndScore, bioAttndScore, phyAttndScore, ma
             bioExamScore, phyExamScore, mathExamScore, chemAvgScore, bioAvgScore, phy
             mathAvgScore, overAllScore, GPA, gradeType, gradeStatus]

cols = ['Name', 'ID', 'Chem Quiz', 'Bio Quiz', 'Phy Quiz', 'Math Quiz', 'Chem HW', 'Bi
            'Math HW', 'Chem Attnd', 'Bio Attnd', 'Phy Attnd', 'Math Attnd', 'Chem Exa
            'Phy Exam', 'Math Exam', 'Chem Avg', 'Bio Avg', 'Phy Avg', 'Math Avg', 'Av

dataFrameCreator(recVals, cols)
```

	Name	ID	Chem Quiz	Bio Quiz	Phy Quiz	Math Quiz	\
0	bett james	GR-0483	127	184	52	133	
1	namukolo abrams	GR-0484	141	177	21	136	
2	vera abutu	GR-0485	51	112	61	74	
3	kwame doga	GR-0486	26	133	62	86	
4	lukman ahmad	GR-0487	29	21	177	50	
5	akin torey	GR-0488	99	90	0	109	
6	luke brant	GR-0489	171	117	151	121	
7	james kenyata	GR-0490	51	188	41	120	
8	ngugi tionga	GR-0491	155	17	12	51	
9	okoro eze	GR-0492	147	61	0	79	
10	agatha chiluba	GR-0493	187	149	117	133	
11	mangu joseph	GR-0494	123	127	48	83	
12	longe jethro	GR-0495	179	120	119	155	
13	florence giwa	GR-0496	98	42	122	84	
14	veiva lucent	GR-0497	17	177	114	60	
15	melody braimoh	GR-0498	67	162	145	104	
16	victor ihab	GR-0499	177	155	135	43	
17	mim trucker	GR-0500	148	141	34	61	
18	maguel peter	GR-0501	23	183	81	33	
19	wellington zuba	GR-0502	182	102	71	60	

	Chem HW	Bio HW	Phy HW	Math HW	...	Phy Exam	Math Exam	Chem Avg	\
0	135	186	142	97	...	73	45	52.26	
1	82	170	12	180	...	31	31	62.97	
2	102	55	105	50	...	98	94	38.43	
3	166	108	59	181	...	65	67	57.98	
4	180	123	11	18	...	51	68	57.94	
5	125	47	119	144	...	44	65	64.44	
6	129	47	63	16	...	19	41	78.66	
7	103	156	31	188	...	18	52	63.20	
8	111	136	77	145	...	43	22	76.27	
9	153	66	97	149	...	75	76	59.80	
10	106	100	88	79	...	61	70	81.96	
11	171	88	139	33	...	35	10	63.81	
12	160	136	119	97	...	70	81	86.88	
13	172	19	20	157	...	68	46	73.22	
14	60	70	109	102	...	31	87	28.90	
15	177	125	65	58	...	14	89	49.82	
16	14	90	6	44	...	33	48	71.68	
17	182	146	104	136	...	23	53	67.52	
18	95	43	143	79	...	16	62	30.62	
19	130	70	28	77	...	19	31	70.71	

	Bio Avg	Phy Avg	Math Avg	Avg Score	GPA	Grade	Status
0	92.70	54.44	52.26	63.00	3.2	D	Pass
1	66.12	22.97	55.01	51.77	2.6	E	Retake
2	52.83	70.44	67.68	57.34	2.9	D	Pass
3	65.27	54.15	64.93	60.58	3.0	D	Pass
4	57.60	55.50	42.81	53.46	2.7	E	Retake
5	36.66	32.94	58.28	48.08	2.4	F	Fail
6	39.31	48.61	40.75	51.83	2.6	E	Retake
7	74.59	27.91	59.27	56.24	2.8	D	Pass
8	30.28	35.21	33.46	43.81	2.2	F	Fail
9	43.59	46.66	65.79	53.96	2.7	E	Retake
10	73.82	62.40	64.21	70.60	3.5	C	Pass
11	47.77	41.48	21.64	43.67	2.2	F	Fail
12	79.45	66.41	73.21	76.49	3.8	B	Pass
13	48.75	53.08	47.69	55.69	2.8	D	Pass
14	54.09	50.44	64.06	49.37	2.5	F	Fail

```
15    55.68    39.42    63.77    52.17  2.6      E   Retake
16    53.00    42.23    38.78    51.42  2.6      E   Retake
17    73.31    32.59    48.66    55.52  2.8      D     Pass
18    46.54    40.96    49.98    42.02  2.1      F     Fail
19    40.35    32.59    40.63    46.07  2.3      F     Fail

[20 rows x 26 columns]
calling function to save file
```

In [554…

```
for row in recVals:
    print(len(row))
```

```
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
20
```

In [ ]: