IT 209 – Homework Assignment - 2 (HA2) – Student Class Definition

In this assignment you are to write a Python program to define a class to model the characteristics of a university student. The class is part of a slightly larger program that includes code to use the class to create some student objects and test its methods. The name of the class is *Student*, and it includes the following methods and attributes:

Method Name	Purpose	Input/param	Output/returns
__init___	Initializer - sets initial attribute values. **name** is a positional attribute; the others are keyword with default values. **g_num** is generated as described in the next table.	See attribute list below	None
__str___	Displays a readable version of the Student object	None	Student data as a printable string
gpa	Calculates and returns the student's GPA. gpa = qpoints / credits	None	gpa (float)
isEnrolled	Returns whether the student is currently enrolled. ('y' = True, 'n' = False)	None	True or False
status	Computes class status based on # of credits completed (90+ = 'Senior', 60+ = 'Junior', 30+ = 'Sophomore', <30 = 'Freshman'	None	String indicating class status ('Senior', etc.)
sameStudent	Compares *self* ("this") student's g_num and name with the input Student object's – returns *True* if match, otherwise returns *False*	Student object (reference id)	True or False

Attribute	Type	Definition
totalEnrollment	int	Class level variable that counts number of Student objects created for entire class
g_num	str	6 character Student ID: "G" + <some # zeroes> + totalEnrollment at the time the student object was created – assigned by the class – range is G00001, G00002, …., G00010, G00011, …G99999
name	str	Student name – "<last name>, <first name>"
major	str	{"ACCT", "AVT", "CS", "HIST", "INFT", "MATH", "MUSI", "PHYS"}
enrolled	str	Is this a currently enrolled student? {"y", "n"}
credits	float	Number of accumulated credits to date
qpoints	float	Number of accumulated quality points to date

The class definition contains an initializer whose signature line looks similar to the following line. The parameters (in parentheses) are the input data to the method. *name* is a required parameter and must be listed first in order. The rest have defaults and can be accessed by keywords if desired.

```
def __init__(self, name, major = 'INFT', enrolled = 'y', credits = 0, qpoints = 0):
```

The code that creates a student object looks similar to this:

```
s1= Student('Fillmore, Sonia ', major = 'MATH', enrolled = 'y', credits = 90, qpoints = 315)
```

Code that prints a student object that implicitly uses the __str___method looks similar to this, with the output

immediately following:

```
print('s1 = ', s1)
s1 =  Sonia Fillmore, G00005, senior, Math, active: y, credits = 90, gpa = 3.5
```

For your submitted code, in addition to the class definition, include the following global code that tests your class. This code is provided as a separate file with the assignment. Copy and paste it into your submitted code. The code creates Student objects, prints those objects, prints additional information on them using dot-notation access to attributes, and executes methods to calculate other information about students:

```
print('\nStart of HA2 Student class demo ')

s1 = Student('David Miller', major = 'HIST',
     enrolled = 'y', credits = 0, qpoints = 0)
s2 = Student('Sonia Fillmore', major = 'MATH',
     enrolled = 'y', credits = 90, qpoints = 315)
s3 = Student('A. Einstein', major = 'PHYS',
     enrolled = 'y', credits = 0, qpoints = 0)
s4 = Student('W. A. Mozart', major = 'MUSI',
     enrolled = 'n', credits = 29, qpoints = 105)
s5 = Student('Sonia Fillmore', major = 'CS',
     enrolled = 'y', credits = 60, qpoints = 130)
s5.g_num = s2.g_num
s6 = Student('Pierre Renoir', major = 'AVT',
     enrolled = 'y', credits = 120, qpoints = 315)
print('s1 = ', s1)
print('s2 = ', s2)
print('s3 = ', s3)
print('s4 = ', s4)
print('s5 = ', s5)
print('s6 = ', s6)
print('\nTotal number of students: ', Student.totalEnrollment)
print('The gpa of ', s1.name, ' is ', s2.gpa())
print('Class standing of ', s4.name, ' is ', s4.status())
print('Class standing of ', s2.name, ' is ', s2.status())
if s1.sameStudent(s2):
   print (s1.name, ' and ', s2.name, ' are the same student')
else:
   print (s1.name, ' and ', s2.name, ' are not the same student')
if s2.sameStudent(s5):
   print (s2.name, ' and ', s5.name, ' are the same student')
else:
   print (s2.name, ' and ', s5.name, ' are not the same student')
if s1.isEnrolled():
   print (s1.name, ' is currently enrolled')
else:
   print (s1.name, ' is not currently enrolled')
if s4.isEnrolled():
   print (s4.name, ' is currently enrolled')
else:
   print (s4.name, ' is not currently enrolled')
```

```
print('\nEnd of HA2 Student class demo')
```

The above code is a testscript that exercises many of the possible combinations of attributes and method uses, but be aware that it does not test 100% of all possibilities. Your Student class might run the entire script correctly yet still have defects. This is an unfortunate reality of software testing: no matter how thorough, testing can only show the presence of defects, not their absence. Providing a testscript that tests all possibilities is therefore impossible for all but the smallest, most trivial programs. You should therefore include your own test cases to supplement the script.

Do the following when examining the output of the testscript and your own test cases:

1. Check the output of the testscript as well as other test scenarios you can think of
2. Check that Student objects are correctly created and printed using the *print* function and the return value of the *__str__()* method
3. Ensure that the g_num is correctly created and is 6 characters long with the correct sequence number and number of zeroes following the 'G' prefix (e.g. G00001, G00002, …)
4. Ensure the *sameStudent* method correctly recognizes objects that match as well as those that don't
5. Ensure the == operator does not equate two different Student objects even when their attributes match
6. Ensure that the gpa, status, and isEnrolled methods correctly compute those values

What and where to submit:

1. Submit by uploading a single file with the Python code to Blackboard
2. Submit the program using the file template provided with the assignment specification.
3. Input a short set of comments as the first lines that identify the program, its purpose, and its author.
4. Include a screen shot of the program's output using, at minimum, the display output resulting from running the testscript

How the assignment will be assessed

The Python code will be visually inspected and executed. The GTA will run it with the above script plus additional test scenarios chosen by them. The GTA will assess each of the following and assign a point value for each.

Item	Assessment Description	Max Value
Python code	A complete program is submitted (mostly a class definition), is named correctly, includes identifying comments, and includes a constructor (*__init__()* method) that correctly creates object	15
g_num	G_num is correctly generated and assigned	10
str	*__str__* correctly prints a readable version of the object data	5
gpa	Method correctly calculates the gpa	5
isEnrolled	Correctly implements the *isEnrolled* and *status* methods	5
sameStudent	Method correctly identifies other Student objects with the same g_num and name	10
Total		**50**