A5 Grading Guide

Step 1: Comment Headers (1 Point)

First you should check to make sure that the students have correctly updated the header comments for a5.py.

Step 2: Style Points (8 Points)

If the student does one of the following, apply the following deductions.

- 1 point: there are still pass statements in the code
- 1 point: there is at least 1 line that is >=80 characters
- 1 point: debugging print statements are still present. **Do not remove the given print statements in Song.play() and Loop.play()**
- 5 points: explicitly calling methods consistently (i.e. Song.play(next_song)).
    - Reduce to 3 point deduction if only uses explicit calls once or twice,
- 1 point: no specs for Mix.__add__()

Note: **max deduction is 8 points** (the deductions all add up to 9, but still only deduct max 8).

Step 3: Run Autograder

The next step is to run the autograder on students' work. This will help check correct functionality and make future steps quicker. For this assignment, there are manual grading points that you must confirm by looking through the students code. And there are functionality points. If they pass all the test cases, they get all the functionality points. Otherwise, award them partial credit using the rubric.

Step 4: fix crashes

Check each feedback file to check that the autograding isn't crashing anywhere. If a crash occurs, the student's autograder assigns 0 points for that function, but they probably deserve more from partial credit. See if any of the files mention errors like syntax error, NameError, UnboundLocalError, AttributeError, etc.

Make the most logical fix to stop the program from crashing, and deduct 5 points each time you do this. Keep track of this yourself, the autograder cannot.

One exception to this rule is the IndexError that is raised in play(), which should already be caught by the autograder, and has a smaller deduction.

Another exception if the student asserts preconditions inside the Loop class, particularly if they used type() instead of isinstance(), it could cause a crash because of the way the autograder is coded. Please comment these assert statements out, but **do not** give any deduction.

Step 5: Mix.__add__(12 points)

If .__add__() does not return a new Mix object, the autograder will crash. Apply the standard 5 point crash tax if this applies. (this includes if the student has modified the attributes of mix1 or mix2 instead of creating a new Mix object). If the student has added additional parameters to the function header this warrants a separate 5 point penalty.

No manual grading

Functionality (12 points)
Note: a student will fail the test cases if they put the other mix before the first (i.e. [other , self] instead of [self, other]), if this is their only mistake, give them full marks for this function.

A submission that passes the autograder earns all 12 points. Otherwise, partial credit can be awarded as follows:

- 6 points: correct value for title attribute (breakdown below)
    - 3 points: Uses some form of string concatenation
    - 3 points: Correct formatting of string
        - If very minor formatting error (missing a space or comma, one character), give ⅔, otherwise, give 0/3
- 6 points: correct value for content attribute (break down below)
    - 3 points: returns any 2 element list
    - 3 points: Has the correct Mix objects in the list

Step 6: Mix.songs (26 points)

For this function, you can inspect the autograder's output and see which test failed to see which deduction you should apply
Manual grading (6 points)

- 3 points: songs is called recursively on Mix objects only
    - Award 0 points if the main function isn't recursive, but the student defines a helper function that is recursive
- 3 points: there is a for loop that iterates over the contents of the Mix (a correct solution that manages to not use a for loop is also acceptable)

Functionality (20 points)
A submission that passes the autograder earns all 20 points. Otherwise, partial credit can be awarded as follows:

- 2 points: Correct with base case: Mix with only songs and no duplicate songs

- 2 points: Correct with base case: Mix with only songs with duplicate songs + keep_dups=True
- 2 points: Correct with base case: Mix with only songs with duplicate songs + keep_dups=False
- 4 points: Correct with Recursive case: Mix with songs and mixes, no duplicate songs
- 4 points: Correct with Recursive case: Mix with songs and mixes with duplicate songs + keep_dups=True
- 4 points: Correct with Recursive case: Mix with songs and mixes with duplicate songs + keep_dups=False
- 2 points: Correct with Triple nested mix with duplicate songs.

ADDED TEST CASE (handled after grade submission):
- 2 points: Correct with Triple nested mix with duplicate songs with keep_dups==True

Step 7: Loop.__init__ (17 points)

Manual grading:
This is not part of the 17 points for this function, but give this deduction if they:
- -3. Includes an extra parameter in the header

Functionality (17 points)
A submission that passes the autograder earns all 17 points. Otherwise, partial credit can be awarded as follows:
- 4 points: correct value for title attribute
- 3 points: correct value for next_pos attribute
- 3 points: correct value for slist when shuffle=False
- 4 points: correct value for slist when shuffle=True
- 3 points: there is a default argument of False for shuffle

NOTE:
This is not part of the 17 points for this function, but give this deduction if they include more attributes. This is tested in the autograder.
- -2. Includes an extra attribute (it will probably be self.shuffle)

Step 8: Loop._prompt (13 points)

We avoid double jeopardy by testing prompt() with a correct version of Loop.__init__. This means that if the student's init method was bugged, but prompt() would be correct had the init was correct, the autograder will still give them full points. As such, for manual grading, you should also assume the instance attributes were instantiated correctly.

No manual grading

Functionality (13 points)
A submission that passes the autograder earns all 13 points. Otherwise, partial credit can be awarded as follows:

- 1 points: Uses some form of string concatenation
- 3 points: Correct formatting of string
    - If very minor formatting error (missing a space or comma, one character), give ⅔, otherwise, give 0/3
- 3 points: Accesses the next song correctly
- 2 points: Accesses a song object's artist correctly (any song, as long as they use .artist correctly)
- 2 points: Accesses a song object's title correctly (any song)
- 2 points: Does not modify any attributes

Step 9: Loop.play (23 points)

Similar to prompt(), the autograder replaced the init() and prompt() method with their correct versions. In manual grading this question, assume that the attributes were instantiated correctly in init, and calls to prompt() are correct.
We have also modified song.play() to NOT open the urls (instead, nothing will happen). You need to manually check that they call song.play() was called in this method. IF your browser opens, it might be a sign that they did not use the song.play() helper function, OR they explicitly called the method (i.e. Song.play(next_song)). Both of these have deductions.

Manual grading (4 points)

- 4 points: calls song.play() anywhere or Song.play(song)
    - Give 3/4 if they do something functionally similar, but do not call the helper. Functionally identical means opening the url of the song, ~~and having some message when the song's url isn't available.~~

Functionality (19 points)

A submission that passes the autograder earns all 19 points. Otherwise, partial credit can be awarded as follows:

- 2 points: Correctly finds the next song
- 2 points: correctly increments next_pos
- 3 points: correctly checks for index out of range for next_pos (or uses % correctly)
    - Give 2/3 point if minor bug in checking (ie missing +1 or -1, using <= instead of <, etc.)
- 3 points: correctly resets next_pos
- 3 points: has correct message(i.e. "(No video available)") if the video has no URL (song.play() should take care of this)
- 2 points: Does not modify self.slist (slist should not be modified here) in play()
- 2 points: Does not modify the code given to them (outside the while loop)

TOTAL OUT OF 100 POINTS