Criterion	0 (Very Poor)	1 (Weak)	2 (Passable)	3 (Good)	4 (Excellent)	Mark
`Flight` **Constructor**	Unimplemented	Barely meets assignment specifications. Severe problems throughout.	Code meets many, but not all of the assignment specifications. Might miss many edge cases or boundary conditions (like off-by-one errors).	Code almost fully meets assignment specifications. There are only one or two edge cases not accounted for.	Code fully satisfies assignment specifications. All edge cases are accounted for.	
Design of `Flight` **Constructor**	Unimplemented, or severe problems throughout.	Code is overly complex and convoluted.	Solution is a bit more complex than needs be.	One or two things about the code could be simplified.	A simple, clear, elegant solution.	
`__str__`	Unimplemented	Barely meets assignment specifications. Severe problems throughout.	Code meets many, but not all of the assignment specifications. Might miss many edge cases or boundary conditions (like off-by-one errors).	Code almost fully meets assignment specifications. There are only one or two edge cases not accounted for.	Code fully satisfies assignment specifications. All edge cases are accounted for.	
Design of `__str__`	Unimplemented, or severe problems throughout.	Code is overly complex and convoluted.	Solution is a bit more complex than needs be.	One or two things about the code could be simplified.	A simple, clear, elegant solution.	
`add_passenger`	Unimplemented	Barely meets assignment specifications. Severe problems throughout.	Code meets many, but not all of the assignment specifications. Might miss many edge cases or boundary conditions (like off-by-one errors).	Code almost fully meets assignment specifications. There are only one or two edge cases not accounted for.	Code fully satisfies assignment specifications. All edge cases are accounted for.	
Design of `add_passenger`	Unimplemented, or severe problems throughout.	Code is overly complex and convoluted.	Solution is a bit more complex than needs be.	One or two things about the code could be simplified.	A simple, clear, elegant solution.	
`heaviest_passenger`	Unimplemented	Barely meets assignment specifications. Severe problems throughout.	Code meets many, but not all of the assignment specifications. Might miss many edge cases or boundary conditions (like off-by-one errors).	Code almost fully meets assignment specifications. There are only one or two edge cases not accounted for.	Code fully satisfies assignment specifications. All edge cases are accounted for.	
Design of `heaviest_passenger`	Unimplemented, or severe problems	Code is overly complex and	Solution is a bit more complex than needs be.	One or two things about the code could be	A simple, clear, elegant solution.	

Criterion	0 (Very Poor)	1 (Weak)	2 (Passable)	3 (Good)	4 (Excellent)	Mark
	throughout.	convoluted.		simplified.		
`lightest_passenger`	Unimplemented	Barely meets assignment specifications. Severe problems throughout.	Code meets many, but not all of the assignment specifications. Might miss many edge cases or boundary conditions (like off-by-one errors).	Code almost fully meets assignment specifications. There are only one or two edge cases not accounted for.	Code fully satisfies assignment specifications. All edge cases are accounted for.	
Design of `lightest_passenger`	Unimplemented, or severe problems throughout.	Code is overly complex and convoluted.	Solution is a bit more complex than needs be.	One or two things about the code could be simplified.	A simple, clear, elegant solution.	
Error Checking	Fails to catch case in both `heaviest_passenger` and `lightest_passenger` where the collection of flight passengers is empty.	-	Catches one of the cases for `heaviest_passenger` and `lightest_passenger` where the collection of flight passengers is empty.	-	Catches both cases in `heaviest_passenger` and `lightest_passenger` where the collection of flight passengers is empty.	
Style	Severe style and readability problems throughout. Variable names not meaningful.	Many serious style, formatting, or readability problems. Variable names usually not meaningful.	Code is readable, but still has several formatting or style issues. Variable names sometimes unclear.	Code only has one or two formatting or style issues. Variable names are usually meaningful.	Code consistently follows all of the formatting rules. Consistently meaningful variable names.	
Docstrings	No Docstrings.	Provides only a few Docstrings, or Docstrings don't provide much useful information.	Satisfactory number of Docstrings that provide some useful information	Good Docstrings that provide useful information.	Fantastic Docstrings that provide very useful information.	
Internal Comments	No internal comments.	Some internal comments, internal comments that don't say anything useful to the reader	Satisfactory amount of useful internal comments - though there are certainly areas that could use some more.	Good amount of useful internal comments - there may be one or two areas that might require some more documentation.	An excellent amount of useful comments. There aren't any areas that require more documentation.	