**Interacting with APIs: Mean Lyrics Calculator - Test Log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description of test** | **Test data to be used (if required)** | **Expected outcome** | **Actual outcome** | **Comments and intended actions** |
| **(test\_connections())**  **Testing API connection by checking response code (200 indicates response received)** | **artist\_info\_connection = requests.get(uri\_artist\_search\_normal)**  **artist\_works\_connection = requests.get(uri\_artist\_works\_normal)**  **artist\_recordings\_connection = requests.get(uri\_artist\_recordings)**  **song\_lyrics\_connection = requests.get(uri\_song\_lyrics)** | **assert artist\_info\_connection.status\_code == 200**  **assert artist\_works\_connection.status\_code == 200**  **assert artist\_recordings\_connection.status\_code == 200**  **assert song\_lyrics\_connection.status\_code == 200** | **artist\_info\_connection == 502 (bad gateway)**  **assert artist\_works\_connection.status\_code == 200**  **assert artist\_recordings\_connection.status\_code == 200**  **assert song\_lyrics\_connection.status\_code == 200** | **Check internet connection and try test again** |
| **(test\_connections())**  **Testing API connection by checking response code (200 indicates response received)** | **artist\_info\_connection**  **artist\_works\_connection**  **artist\_recordings\_connection**  **song\_lyrics\_connection** | **assert artist\_info\_connection.status\_code == 200**  **assert artist\_works\_connection.status\_code == 200**  **assert artist\_recordings\_connection.status\_code == 200**  **assert song\_lyrics\_connection.status\_code == 200** | **assert artist\_info\_connection.status\_code == 200**  **assert artist\_works\_connection.status\_code == 200**  **assert artist\_recordings\_connection.status\_code == 200**  **assert song\_lyrics\_connection.status\_code == 200** | **Successfully retrieving information from server** |
| **(test\_url\_encoder())**  **Testing whether url encoder converts input to string datatype** | **url\_encoder(url)** | **assert isinstance(interacting\_with\_apis.url\_encoder(url), str) == True** | **assert isinstance(interacting\_with\_apis.url\_encoder(url), str) == True** | **URL Encoder successfully creating URL string** |
| **(test\_uri\_search())**  **Testing whether artist search returns string datatype for normal, abnormal and extreme data input** | **uri\_artist\_search\_normal = “abba”**  **uri\_artist\_search\_abnormal = 1**  **uri\_artist\_search\_extreme = “”** | **assert isinstance(uri\_artist\_search\_normal, str)**  **== True**  **assert isinstance(uri\_artist\_search\_abnormal, str)**  **== True**  **assert isinstance(uri\_artist\_search\_extreme, str)**  **== True** | **assert isinstance(uri\_artist\_search\_normal, str)**  **== True**  **assert isinstance(uri\_artist\_search\_abnormal, str)**  **== TypeError: can only concatenate str (not "int") to str**  **assert isinstance(uri\_artist\_search\_extreme, str)**  **== True** | **Cast abnormal data mock input to string as url encoder will successfully perform this process** |
| **(test\_uri\_search())**  **Testing whether artist search returns string datatype for normal, abnormal and extreme data input** | **uri\_artist\_search\_normal = “abba”**  **uri\_artist\_search\_abnormal = 1**  **uri\_artist\_search\_extreme = “”** | **assert isinstance(uri\_artist\_search\_normal, str)**  **== True**  **assert isinstance(uri\_artist\_search\_abnormal, str)**  **== True**  **assert isinstance(uri\_artist\_search\_extreme, str)**  **== True** | **assert isinstance(uri\_artist\_search\_normal, str)**  **== True**  **assert isinstance(uri\_artist\_search\_abnormal, str)**  **== True**  **assert isinstance(uri\_artist\_search\_extreme, str)**  **== True** | **URI search successfully concatenates input and converts to string datatype** |
| **(test\_uri\_search())**  **Testing whether artist search raises exception for normal, abnormal and extreme data input** | **uri\_artist\_search\_normal = “abba”**  **uri\_artist\_search\_abnormal = 1**  **uri\_artist\_search\_extreme = “”** | **with pytest.raises(Exception):**  **requests.get(uri\_artist\_search\_normal)**  **requests.get(uri\_artist\_search\_abnormal)**  **requests.get(uri\_artist\_search\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **requests.get(uri\_artist\_search\_normal)**  **requests.get(uri\_artist\_search\_abnormal)**  **requests.get(uri\_artist\_search\_extreme)**  **assert False** | **No exception raised for URI artist search** |
| **(test\_artist\_num\_choice())**  **Testing whether artist number choice raises exception for normal, abnormal and extreme data input** | **option\_name\_normal = "artist"**  **option\_name\_abnormal = 1**  **option\_name\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.artist\_num\_choice(option\_name\_normal)**  **interacting\_with\_apis.artist\_num\_choice(option\_name\_abnormal)**  **interacting\_with\_apis.artist\_num\_choice(option\_name\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **interacting\_with\_apis.artist\_num\_choice(option\_name\_normal)**  **interacting\_with\_apis.artist\_num\_choice(option\_name\_abnormal)**  **interacting\_with\_apis.artist\_num\_choice(option\_name\_extreme)**  **assert False** | **No exception raised for artist number choice** |
| **(test\_analysed\_songs\_num())**  **Testing whether user input for number of songs to average across raises exception for normal, abnormal and extreme data input** | **analysed\_songs\_normal = 1**  **analysed\_songs\_abnormal = "artist"**  **analysed\_songs\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.analysed\_songs\_num(analysed\_songs\_normal)**  **interacting\_with\_apis.analysed\_songs\_num(analysed\_songs\_abnormal)**  **interacting\_with\_apis.analysed\_songs\_num(analysed\_songs\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **interacting\_with\_apis.analysed\_songs\_num(analysed\_songs\_normal)**  **interacting\_with\_apis.analysed\_songs\_num(analysed\_songs\_abnormal)**  **interacting\_with\_apis.analysed\_songs\_num(analysed\_songs\_extreme)**  **assert False** | **No exception raised for analysed songs number** |
| **(test\_get\_artist\_info())**  **Testing whether retrieving artist info raises exception for normal, abnormal and extreme data input** | **artists\_name\_normal = "abba"**  **artists\_name\_abnormal = '1'**  **artists\_name\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_normal)**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_abnormal)**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_normal)**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_extreme)**  **assert False**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_abnormal) == TypeError: can only concatenate str (not "int") to str** | **Enter abnormal data mock input as string as url encoder will successfully perform this process** |
| **(test\_get\_artist\_info())**  **Testing whether retrieving artist info raises exception for normal, abnormal and extreme data input** | **artists\_name\_normal = "abba"**  **artists\_name\_abnormal = '1'**  **artists\_name\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_normal)**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_abnormal)**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_normal)**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_abnormal)**  **interacting\_with\_apis.get\_artist\_info(artists\_name\_extreme)**  **assert False** | **No exception raised for get artist info** |
| **(test\_get\_artist\_works())**  **Testing whether retrieving artist works raises exception for normal, abnormal and extreme data input** | **uri\_artist\_works\_normal**  **uri\_artist\_works\_abnormal**  **uri\_artist\_works\_extreme** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_works()**  **requests.get(uri\_artist\_works\_normal)**  **requests.get(uri\_artist\_works\_abnormal)**  **requests.get(uri\_artist\_works\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_works()**  **requests.get(uri\_artist\_works\_normal)**  **requests.get(uri\_artist\_works\_abnormal)**  **requests.get(uri\_artist\_works\_extreme)**  **assert False** | **No exception raised for get artist works** |
| **(test\_get\_artist\_recordings())**  **Testing whether retrieving artist recordings raises exception for normal, abnormal and extreme data input** | **song\_id\_normal = "6bb16b15-5e54-3c22-879c-b46fcab750d6"**  **song\_id\_abnormal = 1**  **song\_id\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_recordings(song\_id\_normal)**  **interacting\_with\_apis.get\_artist\_recordings(song\_id\_abnormal)**  **interacting\_with\_apis.get\_artist\_recordings(song\_id\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_artist\_recordings(song\_id\_normal)**  **interacting\_with\_apis.get\_artist\_recordings(song\_id\_abnormal)**  **interacting\_with\_apis.get\_artist\_recordings(song\_id\_extreme)**  **assert False** | **No exception raised for get artist recordings** |
| **(test\_get\_song\_lyrics())**  **Testing whether retrieving song lyrics raises exception for normal, abnormal and extreme data input** | **artists\_name\_normal = "abba"**  **artists\_name\_abnormal = '1'**  **artists\_name\_extreme = ""**  **song\_name\_normal = "waterloo"**  **song\_name\_abnormal = 1**  **song\_name\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_normal, song\_name\_normal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_abnormal, song\_name\_abnormal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_extreme, song\_name\_extreme)**  **assert False** | **AssertionError: Exception Raised** | **Added error handling features to main program loop**  **(try statement at line 230, exceptions at lines 329-370)** |
| **(test\_get\_song\_lyrics())**  **Testing whether retrieving song lyrics raises exception for normal, abnormal and extreme data input** | **artists\_name\_normal = "abba"**  **artists\_name\_abnormal = '1'**  **artists\_name\_extreme = ""**  **song\_name\_normal = "waterloo"**  **song\_name\_abnormal = 1**  **song\_name\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_normal, song\_name\_normal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_abnormal, song\_name\_abnormal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_extreme, song\_name\_extreme)**  **assert False** | **artist = get\_artist\_info(artist\_name) == IndentationError: unexpected indent** | **Removed extra indent from line 185 (artist\_name)** |
| **(test\_get\_song\_lyrics())**  **Testing whether retrieving song lyrics raises exception for normal, abnormal and extreme data input** | **artists\_name\_normal = "abba"**  **artists\_name\_abnormal = '1'**  **artists\_name\_extreme = ""**  **song\_name\_normal = "waterloo"**  **song\_name\_abnormal = 1**  **song\_name\_extreme = ""** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_normal, song\_name\_normal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_abnormal, song\_name\_abnormal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_extreme, song\_name\_extreme)**  **assert False** | **with pytest.raises(Exception):**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_normal, song\_name\_normal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_abnormal, song\_name\_abnormal)**  **interacting\_with\_apis.get\_song\_lyrics(artists\_name\_extreme, song\_name\_extreme)**  **assert False** | **No exception raised for get song lyrics** |