***def explorer.*browse_rows(row_dict_list, **kwargs):**

Allows the user to browse rows of data. Start from 1 if `config_dict["start-from-one"]` is `True`. Otherwise, get a valid row integer from user and start there. Returns if user declines to continue or once last row is printed. Get desired first row from user.

***def* print_random_row(row_dict_list, **kwargs):**

Prints a random row of data. Returns when user declines to print another row. Pick a random list index (insted of using random.choice), so that we can print the row number for the user. The row number is the list index + 1. Get the random row. If the user wants to print another row, repeat the loop. Otherwise, return to main menu.

***def* print_specific_row(row_dict_list, **kwargs):**

Prints a specific row of data specified by the user. Uses `interface.get_valid_row_int` to ensure user selects valid row number. Returns when user declines to print another row. Print rows one by one til user exits. If the user wants to print another row, repeat the loop Otherwise, return to main menu.

***def* get_new_csv_data_dict(new_data_file=*None*, ask_user=*True*, extra_info=*True*, **kwargs):**

Return a new `csv_data_dict` based on a default dict. The `new_csv_dict` function variable should contain keys and default values expected by any functions called by the main menu interface. Add new entries to it when expanded program functionality will expect all CSV dictionaries to contain them by default. Optional argument extra_info controls whether summary information is printed. Optional keyword argument ask_user is passed to `get_selected_columns_list`. Dictionary with all keys expected by default functions

***def* get_selected_columns_list(row_dict_list, ask_user=*True*, **kwargs):**

Get a list of columns to include when printing CSV data. Columns are keys of the first dictionary in row_list If `ask_user` is `True`, prompt the user for each column heading. Otherwise, inlcude all columns. and returns result as dictionary compatible with main menu architecture.

def get_column_type_dict(sample_row, column_type_dict, types=[_str_, _int_, _float_, _bool_], **kwargs):

Given a dictionary of CSV row data, return a dict of column names with user-selected data types. For each column, present user with sample data. Get selected data type for each until conversion is successful. Build dict of column names and data types. Returns dict with `column_type_dict` key and resulting dict as value. Assumes sample_row is representative of all data in CSV. data could not be converted to that type.

def your_new_function(row_dict_list, columns, number_of_rows, **kwargs):

Browse rows of data. Print all data row by row. For each column, present user with data. Build dict of column names and data types. Returns dict with `column_type_dict` key and resulting dict as value. Assumes each row is representative of all data in CSV. data could not be converted to that type.