|  |  |  |
| --- | --- | --- |
| **Function signature** | | **verify\_shipment***(input)* |
| This is the main function that calls all validation functions to validate the shipment data. It returns an array of objects. Each object contains an error associated with the input and their type, and where they exist. | | |
| Arguments: | *Input:* an array of objects each representing one line of input data | |
| Returns: | [Object{flag:String, flagmsg: String, flagname: String, flagval: Integer, line: Integer, priority: Integer}] | |
| Example: | Input : verify\_shipment([{COMMODITY\_DESCRIPTION: "Test", DOMESTIC\_CITY\_NAME: "Barre", DOMESTIC\_STATE\_ABBREV: "MA", DOMESTIC\_TRANSPORT\_MODE: "8", DOMESTIC\_ZIP\_CODE: "01005", EXPORT\_CITY\_NAME: "", EXPORT\_COUNTRY\_NAME: "", EXPORT\_TRANSPORT\_MODE: "", EXPORT\_YN: "N", HAZMAT\_CODE: "1087", SCTG\_COMMODITY\_CODE: "14992", SHIPMENT\_MONTH: "09", SHIPMENT\_VALUE: "1", SHIPMENT\_WEIGHT: "5000", SHIPMT\_DAY: "01", SHIP\_NUM: "123", TEMPERATURE\_CONTROL\_YN: "N"}])  Returns: [{"line": 1,"flagname": "miss\_NAICS","flag": "S49","flagval": 2,"flagmsg": "NAICS code is missing.", "priority": 3},{"line": 1,"flagname": "maxWeight\_threshold\_mode", "flag": "S7", "flagval": 1, "flagmsg": "Shipment weight exceeds maximum weight for modes 1, 2, 3, 8, 12, 13, 18, 21, 31 and 81.", "priority": 1},{"line": 1, "flagname": "nonNumeric\_destinationZip", "flag": "S43","flagval": 1,"flagmsg": "U.S. destination (zip) is not numeric.", "priority": 1},{"line": 1,"flagname": "miss\_exportCity","flag": "S46","flagval": 2,"flagmsg": "Foreign destination (city) is missing.","priority": 1},{"line": 1,"flagname": "miss\_exportCountry","flag": "S47","flagval": 2,"flagmsg": "Foreign destination (country) is missing.","priority": 1},{"line": 1,"flagname": "miss\_exportMode","flag": "S48","flagval": 2,"flagmsg": "Export mode of transport is missing.","priority": 1}] | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_numberOfShip** *(shipNum,nos,line)* |
| This function validates the number of shipments. If an input is present, it checks the input for the allowed characters (it is required to be numeric). Then, it checks the input to be greater than zero and less than the maximum allowable (the maximum allowed value is 100,000). Finally, the function compares the number of required and reported shipments. The function returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *shipNum: Number of reported shipments (string)* * *nos: Required number of shipments (string)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_numberOfShip *(100,120,0)*  Returns: [{"line": 0,"flagname": "miss\_NAICS","flag": "S49","flagval": 2,"flagmsg": "NAICS code is missing.", "priority": 3}] | |
| **Function signature** | | **MOS\_vs\_ATV***(ATV, MOS, estbWeight, evalres,line)* |
| This function validates the consistency of MOS and ATV values. First, it checks the validity of the values of ATV and MOS. If both values are valid, it finds the difference between the two values ‘dif’ and their ratio ‘ratio’. Next, it runs range value check for ‘dif’ and ‘ratio’ to check their consistency with the entered MOS and ATV values. If no errors are found, it runs a range value check for both ‘dif’ and ‘estbWeight’ to check the consistency of ATV and MOS. The function returns an object with two attributes, pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *ATV: ATV value (string)* * *MOS:* MOS *value (string)* * *estWeight: estimated shipment weight* * *evalres: object containing ATV and MOS results (object)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: MOS\_vs\_ATV(10, 20, 100, evalres,2)  ​ Returns: [{"line": 0,"flagname": "miss\_NAICS","flag": "S49","flagval": 2,"flagmsg": "NAICS code is missing.", "priority": 3}] | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_totShipValue***(totShipVal, totValWeek, ATV, estbWeight, evalres,line)* |
| This function validates the total shipment value with regards the validity of the input and its consistency with the ATV value. It validates the entered value first by checking its presence. If no error is found, it checks for the allowed characters (numeric input is required), it also runs range value check (minRange is 1). It also checks the validity of the entered ATV value, if it is valid, it checks for the consistency of the total shipment value with the ATV value. The function returns an object with three attributes, pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *totShipVal: total shipment value* * *totValWeek: total weekly value* * *ATV:* * *estWeight: estimated shipment weight* * *evalres:* * *line: line number* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_totShipValue(100, 2, 10, 100, evalres,2)  ​ Returns: [{"line": 0,"flagname": "miss\_NAICS","flag": "S49","flagval": 2,"flagmsg": "NAICS code is missing.", "priority": 3}] | |
| **Function signature** | | **test\_moreThan40Ship***(input,line)* |
| This function validates the input if the shipment size is more than 40 or not. It first runs presence check, if input is entered, it runs a look-up linear check to validate the existence of the input in a look-up table (lkup25). The function returns an object with two attributes, pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected. | | |
| Arguments: | * *Input: yes or no (string)* * *Line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ship\_ID("yes",1)  Returns: Object {pass: true } | |
| **Function signature** | | **test\_ship\_ID***(input,line)* |
| This function validates the shipment ID. If an input is present, it checks for allowed characters (alphanumeric input is required). It returns an object with two attributes, pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipment ID (string)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ship\_ID("shp123",1)  Returns: Object { pass: true } | |
| **Function signature** | | **test\_ship\_month***(input,line)* |
| This function validates the shipment month. If an input is present, it checks for allowed characters (numeric input is required). It also compares the input with the allowed range of values (1-12). It returns an object with three attributes, pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipment month (string)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ship\_ID("11",1)  Returns: Object { pass: true } | |
| **Function signature** | | **test\_ship\_quarter***(ship\_month, quarter, evalres,line)* |
| This function validates the shipment quarter and its consistency with the entered shipment month. First it checks the presence of the quarter input. If an input (quarter) is entered, it checks for allowed characters (a numeric input is required), it also checks the value range of the quarter (between 1 and 4). Next, it validates the shipment month value and if no error is found, it checks if the ship month matches the quarter value. The function returns an object with three attributes, pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected. | | |
| Arguments: | * *ship\_month: shipment month (string)* * *quarter: shipment quarter (string)* * *evalres:* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ship\_quarter(12,4,{pass:true},2)  Returns: Object { pass: true } | |
| **Function signature** | | **test\_ship\_day***(input,line)* |
| This function validates the shipment day. If an input is present, it checks for allowed characters (numeric input is required). It then checks the input against the range of the allowed values (1-31). It returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipment month (string)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ship\_day("28",1)  Returns: Object { pass: true } | |
| **Function signature** | | **test\_ship\_value***(input,line)* |
| This function validates the shipment value. If an input is present, it checks for the allowed characters (numeric input is required). It then performs a range check (the allowed range is greater than or equal 1). It returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipment month (string)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ship\_value("15",1)  Returns: Object { valid: true, pass: true } | |
| **Function signature** | | **test\_ship\_weight***(weight, mode, naics, evalres,line)* |
| This function validates the shipment weight and its consistency with the entered shipping mode and NAICS code. First, it checks the presence of the input. If an input is present, it checks for the allowed characters (a numeric input is required) and range (minimum value is zero). If no error is found, it validates the entered shipping mode, if the entered shipping mode is valid, it checks if the weight is within the allowed range of the entered shipping mode. Next, it validates the entered NAICS, if the NAICS is valid, it checks the validity of NAICS (lkup30) s well as the consistency of shipping modes and weight. The function returns an object with three attributes of pass, valid and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected. | | |
| Arguments: | * *weight: shipment weight (string)* * *mode: shipping mode (string)* * *naics: NAICS code (string)* * *evalres:* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ship\_weight(100, 12, 2112, evalres,1)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_sctg***(sctg, value, weight, mode, temp, naics, state, evalres,line)* |
| This function validates the entered SCTG code and its consistency with the entered shipment value, shipping mode, temperature, NAICS code, and shipping state. First, it checks the presence of the input (SCTG). If an input (SCTG) is entered, it checks the allowed characters (a numeric input is required), field length (it is required to have a length of 5), and value range of the SCTG (between 01001 and 43999). If no error is found, it checks the validity of the SCTG in table lkup1, and then validates both the shipment value and shipment weight. If no error is found, it checks the consistency of the weight and shipment value ratio with the SCTG code. If the entered NAICS is valid, it checks if the combination of SCTG and the entered NAICS is valid. If the entered shipping mode and weight are valid, it checks the consistency of them with the SCTG code. If the entered temperature is valid, it checks its consistency with the SCTG code. The function returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected. | | |
| Arguments: | * *sctg: SCTG code (string)* * *value: shipment value (string)* * *weight: shipment weight (string)* * *mode: shipping mode (string)* * *temp: temperature controlled or not (string)* * *naics: NAICS code (string)* * *state: shipping state (string)* * *evalres: object storing the validation checks performed on SCTG and NAICS (Object)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_sctg(sctg, value, weight, mode, temp, naics, state, evalres,line)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_sctg\_descr***(input,line)* |
| This function validates the description of the shipped commodity. If an input is present, it checks for allowed characters (alphanumeric input is required). It returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipped commodity description (string)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_sctg\_descr("boxes",1)  Returns: Object { pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_temp\_control***(input,line)* |
| This function validates the shipment control temperature. If an input is present, it checks the existence of the input in a lookup table (lkup25). It returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: y, n (string)* * *line: line number* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_temp\_control("y",1)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_unna***(unna, sctg, evalres,line)* |
| This function validates the UN or NA number and its consistency with the entered SCTG code. First, it checks the presence of the input (UN or NA number). If an input (UN or NA number) is entered, it checks for the allowed characters (only a numeric input is allowed) and the field length of the UN or NA number (4 digits). Then, it checks the value range of the UN or NA number, (between 0004 and 9269). If no error is found, and the SCTG code entered is valid, it checks the consistency of UN or Na number with the SCTG code. The function returns an object with three attributes, pass, valid and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected. | | |
| Arguments: | * *unna: UN or NA number (string)* * *sctg: SCTG code (string)* * *evalres: an object containing the validation results of SCTG (Object)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_unna(unna, sctg, evalres,line)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_destinationCity***(input,line)* |
| This function validates shipment destination city. If an input is present, it checks the allowed characters (alphabetic input is required) and the presence of invalid characters. It returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipment destination city (string)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_destinationCity(‘Columbus’,2)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_destinationState***(input,line)* |
| This function validates the shipment destination state. If an input is present it checks the allowed characters (alphabetic input is required), input length (two letters), and invalid characters. Finally, it checks the existence of the input in a look-up table (lkup32). It returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipment destination city (string)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_destinationState("OH",1)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_destinationZip***(city, state, zip, evalres,line)* |
| This function validates the 5-digit mailing zip code of shipment destination. If an inout is present, it checks the allowed characters (numeric) and input length (5 digits). If no errors were found, it checks the existence of the entered zip code in a lookup table. If the zip code exists, and values provided for the city and state are valid, the validity of the combination of the zip code, city and state is checked. It returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *city: destination city (string)* * *state: destination state’s two-letter abbreviation* *(string)* * *zip: destination city’s zip code (string)* * *evalres: object storing the validation results of city and state (Object)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_destinationZip(‘Athens, ‘oh’, ‘45701’, evalres,1)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_mode***(input,line)* |
| This function validates the entered transportation mode. If an input is present, it checks the allowed characters (numeric input is required) and input length (1-3). If no errors were found, it checks the existence of the shipment mode in a look-up table (lkup3). It returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: transportation mode (string)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, valid: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_mode("1",1)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_export***(input,line)* |
| This function validates value provided for export-yes-no attribute. If an input is present, it checks for the existence of the input in a look-up table (lkup25). It returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: yes or no (string)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_export("y",1)  Returns: Object { pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_exportCity***(input,line)* |
| This function validates the shipment export city name. If an input is present, it checks the allowed characters (alphabetic input is required). It returns an object with three attributes of pass, valid, and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: shipping city name (string)* * *Line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_exportCity(‘Athens’,5)  Returns: Object { pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_exportCountry***(country, city, evalres, line)* |
| This function validates the export country name of a shipment. If an input is present, it checks the allowed characters (it is required to be alphabetic). If the city name is valid, it checks the combination of the city and country names (for Canada and Mexico). If the export countries were not either Canada or Mexico, it only checks the existence of the entered export country in a look-up table. It returns an object with three attributes of pass, valid and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *country: export country name (string)* * *city: export city name (string)* * *evalres:object storing the evaluation result of export\_city (Object)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_exportCountry(‘Toronto’, ‘Canada’, evalres, 10)  Returns: Object { pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_exportMode***(exp\_mode, country, evalres, line)* |
| This function validates the entered export mode of a shipment. If an input is present, it checks the allowed characters (it is required to be numeric). If the country name is valid, it checks the existence of the entered country name in a lookup table. If no error is found, it validates the combination of export mode with the export country (truck export mode is allowed only for Canada and Mexico). It returns an object with three attributes of pass, valid and errors. Pass and valid are binary attributes indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *exp\_mode: export mode number (string)* * *country: export country name (string)* * *evalres:object storing the validation results of country name (Object)* * *line: line number (integer)* | |
| Returns: | Object { valid: Boolean, pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_exportMode(1, ‘Mexico’, evalres, 10)  Returns: Object { pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_auto\_fill\_m***(input,line)* |
| This function runs an auto-fill test on SHIPMENT\_VALUE, SHIMPENT\_WEIGHT, SCTG\_COMMODITY\_CODE, DOMESTIC\_TRANSPORT\_MODE, HAZMAT\_CODE, and DOMESTIC\_ZIP\_CODE. It returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: the input of the listed above attributes* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_auto\_fill\_m("1",1)  Returns: Object { pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_naics***(input,line)* |
| This function validates the NAICS code. If an input is present, it checks the allowed characters (numeric input is required), input length (minimum length is 2 maximum length is 6), and range of the input value (between 11 and 928120). It returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: NAICS code (string)* * *line:* line number (integer) | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_naics("11",1)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_MOS***(input,line)* |
| This function validates the MOS. If an input is present it checks the allowed characters (numeric input is required). It returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: MOS code (string)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_MOS("1",1)  Returns: Object { valid: true, pass: true } | |

|  |  |  |
| --- | --- | --- |
| **Function signature** | | **test\_ATV***(input,line)* |
| This function validates ATV. If an input is present, it checks the allowed characters (only numeric input is allowed) and range (between 0 and infinity). It returns an object with two attributes of pass, and errors. Pass is a binary attribute indicating whether the input has any errors. It is set to true if no error is found, otherwise, it is set to false. Errors is an array of objects reflecting the list of detected errors. | | |
| Arguments: | * *Input: ATV code (string)* * *line: line number (integer)* | |
| Returns: | Object {pass: Boolean, errors: [ Object {flag: String, flagmsg: string, flagname: string, flagval: int, line: int, priority: int }]} | |
| Example: | Input: test\_ATV("11",1)  Object { valid: true, pass: true } | |