

Data 6 Python Cheat Sheet

This cheat sheet has been modified from the Data 6 Python Reference and includes all of the functions and table methods that you will need for the exams.

Built-In Python Functions

Function	Input	Description/Output
<code>str(val)</code>	A value of any type (<code>int</code> , <code>float</code> , <code>NoneType</code> , etc.)	Converts <code>val</code> to a string
<code>int(num)</code> , <code>float(num)</code>	A numerical value (represented as a <code>string</code> or <code>float</code>)	Converts <code>num</code> to the respective numeric type
<code>len(seq)</code>	<code>array</code> or <code>list</code> or <code>string</code>	Returns the length of <code>seq</code>
<code>max(arr)</code> , <code>min(arr)</code> , <code>sum(arr)</code>	<code>array</code> or <code>list</code>	Returns a value in <code>arr</code> . Respectively: maximum, minimum, sum.
<code>abs(val)</code>	<code>int</code> or <code>float</code> or <code>array</code>	Returns the absolute value of <code>val</code> . If <code>val</code> is an array, return an array of absolute values.
<code>round(num)</code> , <code>round(num, ndigits)</code>	<code>int</code> or <code>float</code>	Returns <code>num</code> rounded to the nearest integer. If optional <code>ndigits</code> is provided, round to <code>ndigits</code> precision after the decimal point.
<code>print(input, ...)</code>	<code>input</code> : any inputs to print	Prints the input values, separated by spaces
<code>type(object)</code>	<code>object</code> : the object whose type is to be determined	Returns the type of <code>object</code>

NumPy Array Functions

Function	Description
<code>make_array(val1, val2, ...)</code>	Makes a NumPy array with the inputted values
<code>np.mean(arr)</code> or <code>np.average(arr)</code>	Calculates the average value of <code>arr</code>
<code>np.sum(arr)</code>	Returns the sum of the values in <code>arr</code>
<code>np.prod(arr)</code>	Returns the product of the values in <code>arr</code>
<code>np.sqrt(num)</code>	Calculates the square root of <code>num</code>
<code>np.arange(stop)</code> , <code>np.arange(start, stop)</code> , or <code>np.arange(start, stop, step)</code>	Creates an array of sequential numbers starting at <code>start</code> , going up in increments of <code>step</code> , and going up to but excluding <code>stop</code> . When <code>start</code> and/or <code>step</code> are not specified, default values are used in their place. Default <code>start</code> is 0, default <code>step</code> is 1.
<code>np.count_nonzero(arr)</code>	Returns the number of non-zero (or True) elements in <code>arr</code> .
<code>np.append(arr, item)</code>	Appends <code>item</code> to the end of <code>arr</code> . Does not modify the original array.
<code>np.cumsum(arr)</code>	Returns the cumulative sum of the elements in <code>arr</code> , where each element is the sum of all preceding elements including itself
<code>np.diff(arr)</code>	Returns a new array of size <code>len(arr)-1</code> with elements equal to the difference between adjacent elements; <code>val_2 - val_1</code> , <code>val_3 - val_2</code> , etc.
<code>np.sort(arr)</code>	Sorts an array in ascending order. This function returns nothing; array <code>arr</code> is sorted in-place.
<code>arr.item(i)</code>	[method] Returns the <code>i</code> -th item in <code>arr</code> (remember Python indices start at 0!).
<code>arr.take(indices)</code>	[method] Creates an array with only the elements of <code>arr</code> at the given indices. <code>indices</code> is either an array of indices or an integer corresponding to one index.

<code>percentile(p, arr)</code>	Returns the corresponding percentile <code>p</code> of an array <code>arr</code> .
---------------------------------	--

String Methods and Operations

Function	Description
<code>s.upper()</code>	Returns a copy of <code>s</code> where all letters are uppercase.
<code>s.lower()</code>	Returns a copy of <code>s</code> where all letters are lowercase.
<code>s.replace(old, new)</code>	Returns a copy of <code>s</code> with all occurrences of the substring <code>old</code> replaced by <code>new</code> .
<code>s.split()</code> or <code>s.split(separator)</code> or <code>s.split(separator, maxsplit)</code>	Splits <code>s</code> into a list of substrings using the specified <code>separator</code> ; returns a list of strings. If <code>separator</code> is not provided, splits at any whitespace. You can also use the optional argument <code>maxsplit</code> to limit the number of splits.
<code>s.join(iterable)</code>	Returns a string that concatenates the elements in <code>iterable</code> (usually a list or array) into a single string, with each element separated by <code>s</code> .
<code>s.strip(chars)</code>	Returns a copy of <code>s</code> with the characters in <code>chars</code> trimmed off the left and right ends of the string. If <code>chars</code> is not provided, trims off whitespace.
<code>substr in s</code>	Returns <code>True</code> if the string <code>substr</code> is contained in <code>s</code> (i.e., <code>substr</code> is a substring of <code>s</code>) and <code>False</code> otherwise. A similar operation, <code>x in lst</code> , checks if an element <code>x</code> is in an iterable (list or array).
<code>substr not in s</code>	Returns <code>True</code> if <code>substr</code> is not a substring of <code>s</code> and <code>False</code> otherwise. A similar operation, <code>x not in lst</code> , checks if an element <code>x</code> is not in an iterable (list or array).

Tables and Table Methods

Function	Description	Input	Output
<code>Table()</code>	Creates an empty table, usually to extend with data	None	Table: An empty Table
<code>tbl.with_column(name, values)</code> or <code>tbl.with_columns(n1, v1, n2, v2, ...)</code>	Adds an extra column onto <code>tbl</code> with the label <code>name</code> and <code>values</code> as the column values	1. string: name of the new column 2. array: values in the column	Table: a copy of the original table with the new column(s)
<code>tbl.column(col)</code>	Returns the values in a column in <code>tbl</code>	string or int: the column name or index	array: the values in that column
<code>tbl.num_rows</code>	Compute the number of rows in <code>tbl</code>	None	int: the number of rows in the table
<code>tbl.num_columns</code>	Compute the number of columns in <code>tbl</code>	None	int: the number of columns in the table
<code>tbl.labels</code>	Returns the labels in <code>tbl</code>	None	array: the names of each column as strings
<code>tbl.select(col1, col2, ...)</code>	Creates a copy of <code>tbl</code> only with the selected columns	string or int: column name(s) or index(es)	Table with the selected columns

<code>tbl.drop(col1, col2, ...)</code>	Creates a copy of <code>tbl</code> without the selected columns	string or int : column name(s) or index(es)	Table without the selected columns
<code>tbl.relabeled(old_label, new_label)</code>	Creates a new table, changing the column name specified by <code>old_label</code> to <code>new_label</code> , and leaves the original table unchanged	1. string : old column name 2. string : new column name	Table : a copy of the original table with the changed column name
<code>tbl.show(n)</code>	Displays the first <code>n</code> rows of <code>tbl</code> . Defaults to showing the entire table	(Optional) int : number of rows to display	None (table is displayed)
<code>tbl.sort(column_name[, descending=True])</code>	Sorts rows by <code>column_name</code> . Ascending by default unless <code>descending=True</code> is included	1. string or int : name or index of the column to sort 2. (Optional) <code>descending=True</code>	Table : a copy of the original table with the column sorted
<code>tbl.where(column, predicate)</code>	Creates a copy of <code>tbl</code> containing only the rows where the value of <code>column</code> matches the <code>predicate</code>	1. string or int : column 2. <code>are(...)</code> predicate	Table : a copy of the original table with only the rows that match the predicate
<code>tbl.take(row_indices)</code>	Creates a table with only the rows at the given indices	int or array : indices of rows to be included in the table	Table : a copy of the original table with only the rows at the given indices
<code>tbl.apply(function)</code> or <code>tbl.apply(function, col1, col2, ...)</code>	Returns an array of values resulting from applying a function to each item in a column.	1. Function: function to apply to column 2. (Optional) string or i : the column name(s) or index(es) to apply the function to	array containing an element for each value in the original column after applying the function to it
<code>tbl.group(column_or_columns, function)</code>	Groups rows in <code>tbl</code> by unique values or combinations of values in a column(s). Multiple columns must be entered as an array of strings. Values in the other columns are aggregated by count (by default) or the optional argument <code>function</code> .	1. string or array of strings : column(s) on which to group 2. (Optional) Function: function to aggregate values in cells (defaults to counting rows)	Table a new grouped table

<code>tbl.pivot(col1, col2)</code> or <code>tbl.pivot(col1, col2, values, collect)</code>	Creates a pivot table where each unique value in <code>col1</code> has its own column and each unique value in <code>col2</code> has its own row. Counts or aggregates values from a third column, collected with some function. If the values and collect arguments are not included, pivot defaults to returning counts in the cells.	1. string: name of the column in <code>tbl</code> whose unique values will make up the columns of the pivot table 2. string: name of column in <code>tbl</code> whose unique values will make up the rows of the pivot table 3. (Optional) string: name of the column in <code>tbl</code> that describes the values of cells in the pivot table 4. (Optional) Function: how the values are collected (e.g. <code>sum</code> or <code>np.mean</code>)	Table : a new pivot table
<code>tblA.join(colA, tblB)</code> or <code>tblA.join(colA, tblB, colB)</code>	Generate a table with the columns of <code>tblA</code> and <code>tblB</code> , containing rows for all values in <code>colA</code> and <code>colB</code> that appear in <code>tblA</code> and <code>tblB</code> , respectively. By default, <code>colB</code> is the same value as <code>colA</code> . <code>colA</code> and <code>colB</code> must be strings specifying column names.	1. string: name of column in <code>tblA</code> with values to join on 2. Table: the other table 3. (Optional) string: the name of the shared column in <code>tblB</code>, if column names are different between the tables	Table : a new combined table
<code>tbl.with_row(values)</code>	Adds a new row with the specified values to <code>tbl</code>	list or array : values to add as a new row	Table : a copy of the original table with the new row
<code>tbl.with_rows(list_of_rows)</code>	Adds multiple rows to <code>tbl</code> using a list of rows	list of lists or arrays : each list/array represents a new row	Table : a copy of the original table with the new rows

Visualization Functions

Function	Description	Input	Output
<code>tbl.barh(categories)</code> or <code>tbl.barh(categories, values)</code>	Displays a horizontal bar chart with bars for each category in the column categories . values specifies the column corresponding to the size of each bar, but is unnecessary if the table only has two columns. Optional argument overlay (default is <code>True</code>) specifies whether grouped bar charts should be overlaid or on separate plots.	1. string: name of the column with categories 2. (Optional) string: name of the column with values corresponding to the categories	None : draws a bar chart
<code>tbl.hist(column)</code>	Generates a histogram of the numerical values in column . Optional arguments group (to specify categorical column to group on), bins (to specify custom bins), and overlay to specify overlaid or separate histograms.	string : name of the column	None : draws a histogram

<code>tbl.plot(x_column, y_column)</code> or <code>tbl.plot(x_column)</code>	Draws a line plot consisting of one point for each row in <code>tbl</code> . If only <code>x_column</code> is specified, <code>plot</code> will plot the rest of the columns on the y-axis with different colored lines. Optional argument <code>overlay</code> (default is <code>True</code>) specifies whether multiple lines should be overlaid or on separate plots	1. string : name of the column for x-axis 2. string : name of the column for y-axis	None : draws a line plot
<code>tbl.scatter(x_column, y_column)</code>	Draws a scatter plot consisting of one point for each row in <code>tbl</code> . The optional argument <code>fit_line=True</code> can be included to draw a line of best fit through the scatter plot. The optional arguments <code>group</code> (to specify categorical column to group on) and <code>sizes</code> (to specify a numerical column for bubble sizes) can also be used to encode additional variables.	1. string : name of the column on the x-axis 2. string : name of the column on the y-axis 3. (Optional) <code>fit_line=True</code>	None : draws a scatter plot

Table.where (Boolean) Predicates

These functions can be passed in as the second argument to `tbl.where(..)` and act as a condition by which to select rows from `tbl`.

Predicate	Description
<code>are.equal_to(Z)</code>	Equal to Z (Z can be an <code>int</code> , <code>float</code> , or <code>string</code>).
<code>are.not_equal_to(Z)</code>	Not equal to Z (Z can be an <code>int</code> , <code>float</code> , or <code>string</code>).
<code>are.above(x)</code>	Greater than x.
<code>are.above_or_equal_to(x)</code>	Greater than or equal to x.
<code>are.below(x)</code>	Less than x.
<code>are.below_or_equal_to(x)</code>	Less than or equal to x.
<code>are.between(x, y)</code>	Greater than or equal to x and less than y.
<code>are.between_or_equal_to(x, y)</code>	Greater than or equal to x and less than or equal to y.
<code>are.strictly_between(x, y)</code>	Greater than x and less than y.
<code>are.contained_in(A)</code>	True if it is a substring of A (if A is a string) or an element of A (if A is an array)
<code>are.containing(S)</code>	Contains the string S.