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 Quiz 2.

Built-In Python Functions

Function	Description	Input	Output
`str(val)`	Converts `val` to a string	A value of any type (**int**, **float**, **NoneType**, etc.)	The value as a **string**
`int(num)`	Converts `num` to an int	A numerical value	The value as an **int**
`float(num)`	Converts `num` to a float	A numerical value	The value as a **float**
`len(arr)`	Returns the length of `arr`	**array** or **list**	**int**: the length of the array or list
`max(arr)` or `min(arr)`	Returns the maximum or minimum value in `arr`	**array** or **list**	The maximum/minimum value the array (usually an **int**)
`sum(arr)`	Returns the sum of the values in `arr`	**array** or **list**	**int** or **float**: the sum of the values in the array
`abs(num)`	Returns the absolute value of `num`	**int** or **float**	**int** or **float**

NumPy Array Functions

Function	Description	Input	Output
`make_array(v1, v2, ...)`	Makes a NumPy array with the inputted values	A sequence of values	An **array** with those values
`arr.item(n)`	Gets the item in the array `arr` at index `n`. Remember that indices start at 0.	An **int** corresponding to	The item at index `n` in `arr`

		the index of the item	
`np.mean(arr)`	Calculates the average value of `arr`	An **array** of numbers	**float**: The average of the array
`np.arange(stop)` or `np.arange(start, stop)`	Creates an array of sequential numbers starting at `start` and going up to but excluding `stop`	**int** or **float**	**array**

Table Methods

Function	Description	Input	Output
`tbl.column(col)`	Returns the values in a column	**string** or **int**: the column name or index	**array**: the values in that column
`tbl.num_rows`, `tbl.num_columns`	Computes the number of rows or columns in `tbl`	None	**int**: the number of rows or columns in the table
`tbl.select(c1, c2, ...)`	Creates a copy of `tbl` only with the selected columns	**string** or **int**: the column name(s) or index(es) to be included in the table	**Table** with the selected columns
`tbl.sort(column_name)`	Sorts the rows of `tbl` by the values in the `column_name` column. Defaults to ascending order.	1. **string** or **int**: name or index of the column to sort 2. (Optional) `descending=True`	**Table**: copy of the table with the column sorted
`tbl.where(column, predicate)`	Creates a copy of `tbl` containing only the rows where the value of `column` matches the `predicate`.	1. **string** or **int**: column name or index 2. the value to match to	**Table**: copy of the table with only the rows that match the predicate

`tbl.take(row_indices)`	Creates a table with only the rows at the given indices.	**int** or **array**: indices of rows to be included in table	**Table**: copy of table with only the rows at the given indices
`tbl.apply(function)` or `tbl.apply(function, col1, col2, ...)`	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 **int**: column name(s) or index(es) to apply to	**array** containing an element for each value in the original column after applying the function to it
`tbl.group(column, function)`	Groups rows in `tbl` by unique values in a column. Values in the other columns are aggregated by count (by default) or the optional argument `function`.	1. **string** or **array of strings**: column(s) on which to group 2. (Optional) **Function**: function to aggregate values (defaults to counting rows)	**Table** a new groupped table
`tbl.pivot(col1, col2)` or `tbl.pivot(col1, col2, values, collect)`	Creates a pivot table where each unique value in `col1` has its own column and each unique value in `col2` has its own row. Counts or aggregates values from a third column, collected with some function (by default, aggregates by count).	1. **string**: name of column for the rows 2. **string**: name of column for the columns 3. (Optional) **string**: name of column for the values to be aggregated 4. (Optional) **Function**: how values are collected	**Table**: new pivot table
`tblA.join(colA, tblB)` or `tblA.join(colA, tblB, colB)`	Generates a table with the columns of `tblA` and `tblB`, containing rows for all values in `colA` and `colB` that match.	1. **string**: name of column in `tblA` 2. **Table**: other table 3. (Optional) **string**: name of shared column in `tblB`	**Table**: a new combined table

Visualization Functions

Function	Description	Input	Output
`tbl.barh(categories)` or `tbl.barh(categories, values)`	Displays a horizontal bar chart with bars for each category in the column `categories`	1. **string**: categories column 2. (Optional) **string**: numerical values column	None: draws a bar chart
`tbl.hist(column)`	Generates a histogram of the numerical values in `column`.	**string**: column name	None: draws a histogram
`tbl.plot(x_column, y_column)` or `tbl.plot(x_column)`	Draws a line plot consisting of one point for each row in `tbl`. If only `x_column` is specified, `plot` will plot the rest of the columns on the y-axis.	1. **string**: x-axis column 2. **string**: y-axis column	None: draws a line graph
`tbl.scatter(x_column, y_column)`	Draws a scatter plot consisting of one point for each row in `tbl`.	1. **string**: x-axis column 2. **string**: y-axis column	None: draws a scatter plot

Conditional Statements and Iteration

Syntax	Description

`if <if expression>:`
`  <if body>`
`elif <elif expression>:`
`  <elif body>`
`else:`
`  <else body>`
	Executes the code in `<if body>` only if `<if expression>` evaluates to `True`. If `<if expression>` is `False`, checks `<elif expression>` and executes code in `<elif body>` if `True`. Otherwise, executes the code in `<else body>`

`for <element> in <sequence>:` `  <for body>`	Repeats code in `<for body>` for each `<element>` in `<sequence>` (array, string, etc.), assigning `<element>` to each value in `<sequence>` one at a time
`while <boolean expression>:` `  <while_body>`	Repeats code in `<while body>` while `<boolean expression>` is `True`