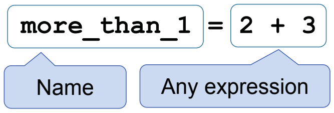
Data 8 Midterm Reference Sheet — Page 1



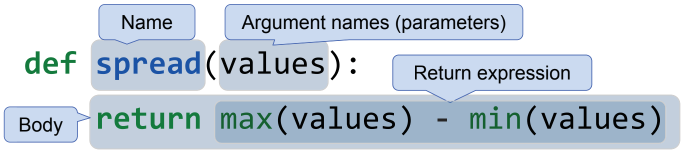
```
more_than_1 = 2 + 3
```

- Statements don't have a value; they perform an action
- An assignment statement changes the meaning of the name to the left of the = symbol
- The name is bound to a value (not an equation)

- < and > mean what you expect (less than, greater than)
- <= means "less than or equal"; likewise for >=
- == means "equal"; != means "not equal"
- Comparing strings compares their alphabetical order

Arrays - sequences that can be manipulated easily.

- All elements of an array should have the same type
- Arithmetic is applied to each element of an array individually
- Elementwise operations can be done on arrays of the same size



```
def spread(values):
    return max(values) - min(values)
```

```
for i in np.arange(12):
    print(i)
```

The body is executed **for** every item in a sequence
The body of the statement can have multiple lines
The body should do something: print, assign, hist, etc.

```
Conditional Statements
    if <if expression>:
        <if body>
    elif <elif expression 0>:
        <elif body 0>
    elif <elif expression 1>:
        <elif body 1>
    ...
    else:
        <else body>
```

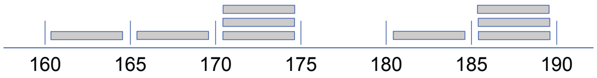
Values in Tables: Every column of a table is an array.

- **Categorical**
  - May or may not have an ordering
  - Categories are the same or different
  - Allows grouping by value (`group, groups, pivot, join`)
- **Numerical**
  - Ordered
  - Allows binning by value (`bin, hist`)

Binning is counting the number of numerical values that lie within ranges, called bins.

- Bins are defined by their lower bounds (inclusive)
- The upper bound (exclusive) is the lower bound of the next bin

163, 168, 170, 171, 173, 183, 185, 188, 189, ...

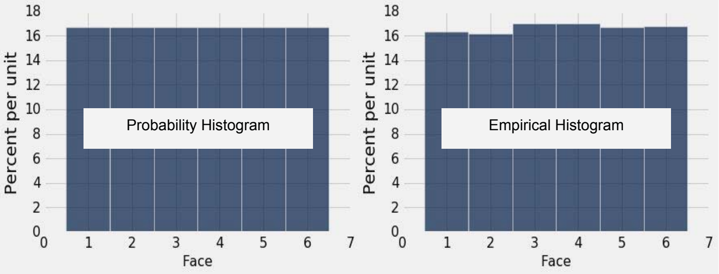


A **histogram** has two defining properties:

- The bins are contiguous (though some might be empty) and are drawn to scale
- The **area** of each bar is equal to the proportion of entries in the bin

Has total area 1 (or 100%)
Vertical axis units: Proportion / Unit on the horizontal axis

- A histogram of proportions of all possible outcomes of a *known* random process is called a *probability histogram*
- A histogram is a summary visualization of a *distribution*
- A histogram of proportions of actual outcomes generated by sampling or actual data is called an *empirical histogram*



Predicate	Description
`are.equal_to(Z)`	Equal to `Z`
`are.above(x)`	Greater than `x`
`are.above_or_equal_to(x)`	Greater than or equal to `x`
`are.below(x)`	Less than `x`
`are.below_or_equal_to(x)`	Less than or equal to `x`
`are.between(x, y)`	Greater than or equal to `x`, and less than `y`
`are.strictly_between(x, y)`	Greater than `x` and less than `y`
`are.between_or_equal_to(x, y)`	Greater than or equal to `x`, and less than or equal to `y`
`are.containing(S)`	Contains the string `S`

Data 8 Midterm Reference Sheet — Page 2

In the examples in the left column, `np` refers to the NumPy module, as usual. Everything else is a function, a method, an example of an argument to a function or method, or an example of an object we might call the method on. For example, `tbl` refers to a table, `array` refers to an array, and `num` refers to a number. `array.item(0)` is an example call for the method `item`, and in that example, `array` is the name previously given to some array.

Example function call	Value of a call to the function
`max(array)`	Maximum or minimum of a sequence
`sum(array)`	Sum of all elements in an array
`len(array)`	Length (num elements) in an array
`round(num); np.round(array)`	Round number or array of numbers to the nearest integer
`abs(num); np.abs(array)`	Take the absolute value of number or each number in an array
`np.average(array)`	The average of the values in an array

`np.arange(start, stop, step)`
`np.arange(start, stop)`
`np.arange(stop)`
	An array of numbers starting with `start`, going up in increments of `step`, and going up to but excluding `stop`. When `start` and/or `step` are left out, default values are used in their place. Default `step` is 1; default `start` is 0.

`np.count_nonzero(array)`	Count the number of non-zero elements in an array (`False` counts as zero, `True` as non-zero)
`array.item(index)`	The item in the array at some index. `array.item(0)` is the first item of `array`.
`np.append(array, item)`	A copy of the array with item appended to the end.

`np.random.choice(array, num)`
`np.random.choice(array)`
	An array of things randomly selected with replacement from an array. `num` is the number of things selected. Default `num` is 1.

`Table()`	An empty table.
`Table.read_table(filename)`	A table with data from a file.
`tbl.num_rows`	The number of rows in a table.
`tbl.num_column`	The number of columns in a table.
`tbl.labels`	A list of the column labels of a table.

`tbl.with_column(name, values)`
`tbl.with_columns(n1, v1, n2, v2…)`
	A table with an additional or replaced column or columns. `name` is a string for the name of a column, `values` is an array.

`tbl.column(column_name)`	The values of a column (an array).
`tbl.select(col1, col2, …)`	A table with only the selected columns. (Each argument is the name of a column.)
`tbl.drop(col1, col2, …)`	A table without the selected columns. (Each argument is the name of a column.)
`tbl.relabeled(old_label, new_label)`	A new table with a label changed.
`tbl.relabel(old_label, new_label)`	Change the label of a column in place. (Has no value!)
`tbl.take(row_indices)`	A table with only the rows at the given indices. `row_indices` is an array of indices.
`tbl.sort(column, descending)`	A table of rows sorted according to the values in a column. Default order is ascending.
`tbl.where(column, predicate)`	A table of the rows for which the column satisfies some predicate. See "`Table.where predicates`" below.
`tbl.apply(function, column)`	Returns an array where a function is applied to each item in a column.

`tbl.group(column, func)`
`tbl.groups(column_names_array, func)`
	Group rows by unique values in a column. Other values aggregated by count (default) or optional arg `func.`
	Group rows by unique combinations of values in some columns. Aggregate/count other values as above.

`tblA.join(colA, tblB, colB)`
`tblA.join(colA, tblB)`
	Generate a table with the columns of self and other, containing rows for all values of a column that appear in both tables. Default `colB` is `colA. colA` is a string specifying a column name, as is `colB.`

`tbl.pivot(row, col, values, collect)`
`tbl.pivot(row, col)`
	Group rows by unique values in two columns; count or aggregate values from a third column, collect with some function. Default `values` and `collect` return counts in cells.

`tbl.sample(n, with_replacement)`	Returns a new table where k rows are randomly sampled from the original table. Default is with replacement.
`tbl.scatter(x_column, y_column)`	Draws a scatter plot consisting of one point for each row of the table.

`tbl.barh(categories)`
`tbl.barh(categories, frequencies)`
	Displays a bar chart with bars for each category in a column, with height proportional to the corresponding frequency. `frequencies` argument unnecessary if table consists just of a column of categories and a column of frequencies.

`tbl.hist(column, units, bins)`	Generates a histogram of the numerical values in a column. `units` and `bins` are optional arguments, used to label the axes and group the values into intervals (bins), respectively.

Operations: addition 2+3=5; subtraction 4-2=2; division 9/2=4.5 multiplication 2*3=6; division remainder 11%3=2; exponent 2**3=8

Data Types: string 'hello'; **boolean** True, False; **int** 1, -5; **float** - 2.3, -52.52, 7.9

Arithmetic with arrays is elementwise:

```
make_array(1,2,3) ** 2 # [1, 4, 9]
```

Table.where predicates (x is a string or number)

```
are.equal_to(x) # [2, 3, 4]
are.above(x) # val > x
are.below(x) # val < x
are.between(x, y) # x <= val < y
```