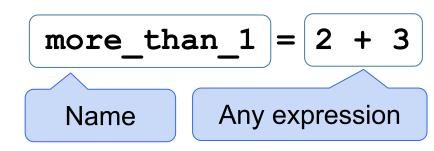
### Data 8 Midterm Reference Guide — Page 1

#### **Statements**



- 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).

### **Comparisons**

- < 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 of the same type that can be manipulated

Arithmetic and comparisons are applied to each element of an array individually

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

Elementwise operations can be done on arrays of the same size

```
o make_array(3,2) * make_array(5,4) # array([15,8])
```

#### **Defining a Function**

```
def function_name(arg1, arg2, ...):
    # Body can contain anything inside of it
    return # a value (the output of the function call)
```

### **Defining a Function with no arguments**

```
def function_name():
    # Body can contain anything inside of it
    return # a value (the output of the function call)
```

• Functions with no arguments can be called by function name()

### **For Statements**

```
total = 0
for _i in _ np.arange(12)
total = total + 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: assign, sample, print, 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>
```

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

```
Data Types: string "hello"; boolean True, False; int 1, -5; float - 2.3, -52.52, 7.9, 8.0
```

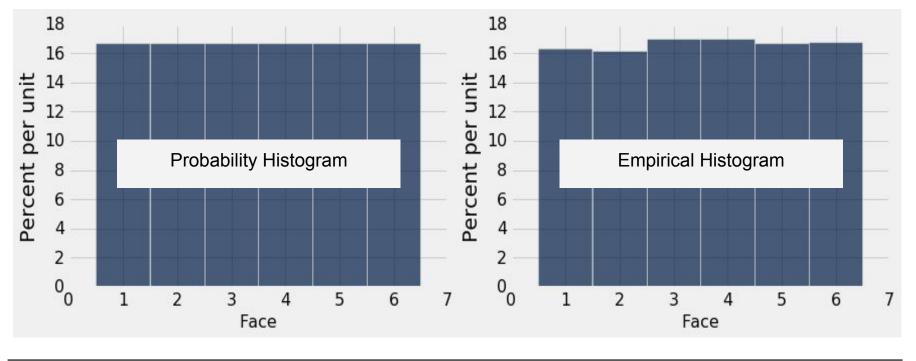
**Table.where predicates:** Any of these predicates can be negated by adding "not\_" in front of them, e.g. are.not\_equal\_to(x)

```
are.equal_to(x) # val == x
```

- are.above(x) # val > x
- are.above\_or\_equal\_to(x) # val >= x
- are.below(x) # val < x</pre>
- are.between(x, y) # x <= val < y</pre>
- are.containing(s) # contains the string s

#### A **histogram** has a few defining properties:

- The bins are continuous (though some might be empty) and are drawn to scale
- The area of each bar is equal to the percent of entries in the bin
- The total area is 100%
- The histogram on the left represents the theoretical probabilities in the distribution of the face that appears on one roll of a fair die
- The histogram on the right represents the observed distribution of the faces after rolling the die many times
- If we keep rolling, the right hand histogram is likely to look more like the one on the left



# **Calculating Probabilities**

Complement Rule: P(event does not happen) = 1 - P(event happens)

Multiplication Rule: P(two events both happen) = P(one happens) \* P(the other happens, given that the first happened)

Addition Rule: If an event can happen in ONLY one of two ways: P(event happens) = P(first way it can happen) + P(second way it can happen)

### **Simulating a Statistic:**

- Create an empty array in which to collect the simulated values
- For each repetition of the process
  - Simulate one value of the statistic
  - Append this value to the collection array
- At the end, all simulated values will be in the collection array

# Data 8 Midterm Reference Guide — 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.

<pre>max(array); min(array) </pre>	Maximum or minimum of an array
sum(array)	Sum of all elements in an array; The sum of an array of boolean values is the number of values that are True
len(array)	Length (num elements) in an array
round(num); np.round(array)	The nearest integer to a single number or each number in an array
abs(num); np.abs(array)	The absolute value of a single number or each number in an array
<pre>np.average(array), np.mean(array)</pre>	The average of the values in an array
<pre>np.arange(start, stop, step) np.arange(start, stop) np.arange(stop)</pre>	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 places. Default step is 1; default start is 0.
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. If item is another array, all of its elements are appended.
<pre>np.random.choice(array) np.random.choice(array, n)</pre>	An item selected at random from an array. If n is specified, an array of n items selected at random with replacement is returned. Default n is 1.
np.ones(n)	An array of length n which consists of all ones.
np.diff(array)	An array of length len(array)-1 which contains the difference between adjacent elements.
np.count_nonzero(array)	An integer corresponding to the number of non-zero (or True) elements in an array.
<pre>sample_proportions(sample_size, model_proportions)</pre>	An array of proportions that add up to 1. The result of sampling sample_size elements from a distribution specified by model_proportions, and keeping track of the proportion of each element sampled.
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_columns	The number of columns in a table.
tbl.labels	A list of the column labels of a table.
<pre>tbl.with_column(name, values) tbl.with_columns(n1, v1, n2, v2)</pre>	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_or_index)	An array containing the values of a column
tbl.select(col1, col2,)	A table with only the selected columns. (Each argument is the label of a column, or a column index.)
tbl.drop(col1, col2,)	A table without the dropped columns. (Each argument is the label of a column, or a column index.)
tbl.relabeled(old_label, new_label)	A new table with a label changed.
<pre>tbl.take(row_index) tbl.take(row_indices)</pre>	A table with only the row(s) at the given index or multiple indices. row_indices must be an array of indices.
tbl.sort(column_name_or_index)	A table of rows sorted according to the values in a column (specified by name/index). Default order is ascending. For descending order, use argument descending=True. For unique values, use distinct=True.
tbl.where(column, predicate)	A table of the rows for which the column satisfies some predicate. See "Table.where predicates" on Page 1.
tbl.apply(function, column)	An array of results when a function is applied to each item in a column.
tbl.group(column_or_columns)	A table with the counts of rows grouped by unique values or combinations of values in a column or columns.
tbl.group(column_or_columns, func)	A table that groups rows by unique values or combinations of values in a column or columns. The other values are aggregated by func. All column names (except the one(s) we group by) will now be `original_name func`. If a column is named 'price', and we group using the min function, our new column name will be 'price min'.
<pre>tblA.join(colA, tblB, colB) tblA.join(colA, tblB)</pre>	A table with the columns of tblA and tblB, containing rows for all values of a column that appear in both tables. Default value of colB is colA. colA is a string specifying a column name, as is colB.
<pre>tbl.pivot(col1, col2) tbl.pivot(col1, col2, vals, collect)</pre>	A pivot table where each unique value in col1 has its own column and each unique value in col2 has its own row. The cells of the grid contain row counts (two arguments) or the values from a third column, aggregated by the collect function (four arguments).
<pre>tbl.sample(n) tbl.sample(n, with_replacement)</pre>	A new table where n rows are randomly sampled from the original table. Default is with replacement. For sampling without replacement, use argument with_replacement=False. If sample size n is not specified, the default is the number of rows in the original table.
tbl.scatter(x_column, y_column)	Draws a scatter plot consisting of one point for each row of the table.
<pre>tbl.barh(categories) tbl.barh(categories, values)</pre>	Displays a bar chart with bars for each category in a column, with length proportional to the corresponding frequency. If values is not specified, overlaid bar charts of all the remaining columns are drawn.
tbl.bin(column, bins)	A table of how many values in a column fall into each bin. Bins include lower bounds & exclude upper bounds.
tbl.hist(column, unit, bins, group)	Displays a histogram of the values in a column. unit and bins are optional arguments, used to label the axes and group the values into intervals (bins), respectively. Bins include lower bounds & exclude upper bounds. If group is specified, the rows are grouped by the values in the column, and histograms for all the groups are overlaid.