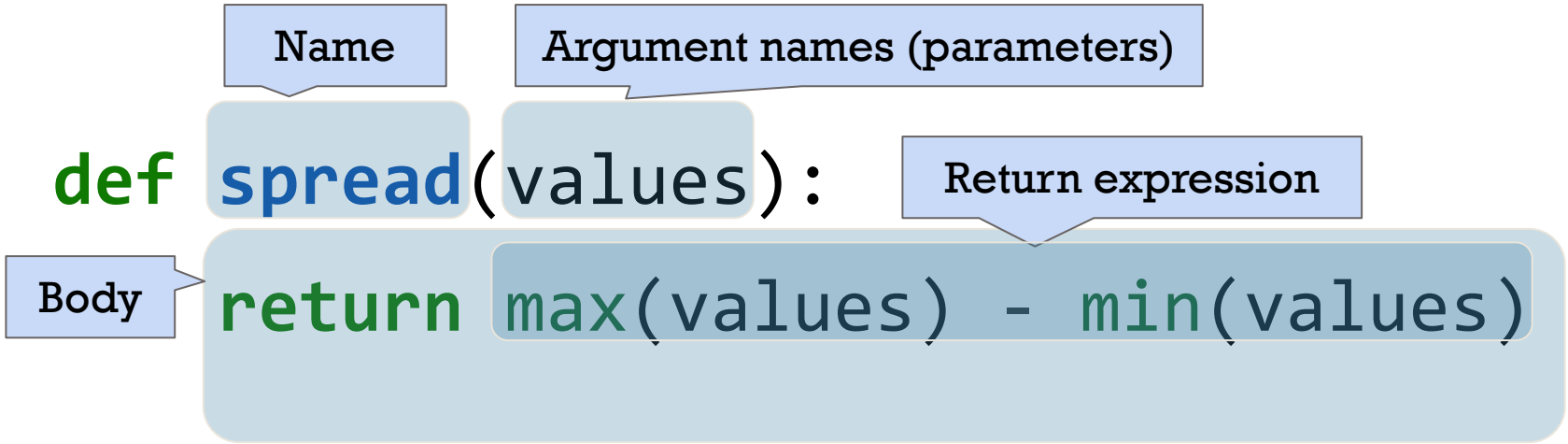
MODULE 4

Functions and Tables

DEFINING FUNCTIONS

DEF STATEMENTS

User-defined functions give names to blocks of code



(Demo – notebook 4.1)

FUNCTION DEFINITION DETAILS

Signature

- Calls to the function will look like this (with the same name and number of arguments). Example: `double(3)`.
- When you call `double`, the argument can be any expression. (The name `x` doesn't affect calls.)
- In the body of the function, `x` is the name of the argument, as if the body included the code `x = <the first argument>`.

```
# Our first function definition

def double(x):
    """ Double x """
    return 2*x
```

Documentation ("docstring")

- Text that describes what the function does.
- Can be any string, traditionally triple-quoted so it can span several lines.
- Traditionally, the first line describes what the function does, briefly.
- Subsequent lines can give more detail and examples.
- Running `double?` will show this text, just like `max?` will show the documentation for the built-in function `max`.

Body

- All the code in here runs each time you call the function.
- The special statement `return` tells Python what the value of each call to this function is: it's the value of the expression after `return`.
- For example, the value of `double(3)` is `6`. (Remember, when the argument is `3`, it's like the body starts with `x = 3`.)
- Often, the body will have multiple lines of code that build up to computing the `return`ed value. You can write any Python code here that you could write anywhere else.

Indentation

- Each line of code in the body is indented (that is, it's preceded by spaces).
- Traditionally, we use 2 or 4 spaces. They only need to be consistent.
- This tells Python that those lines are part of the body.
- The function's body ends at any unindented line.

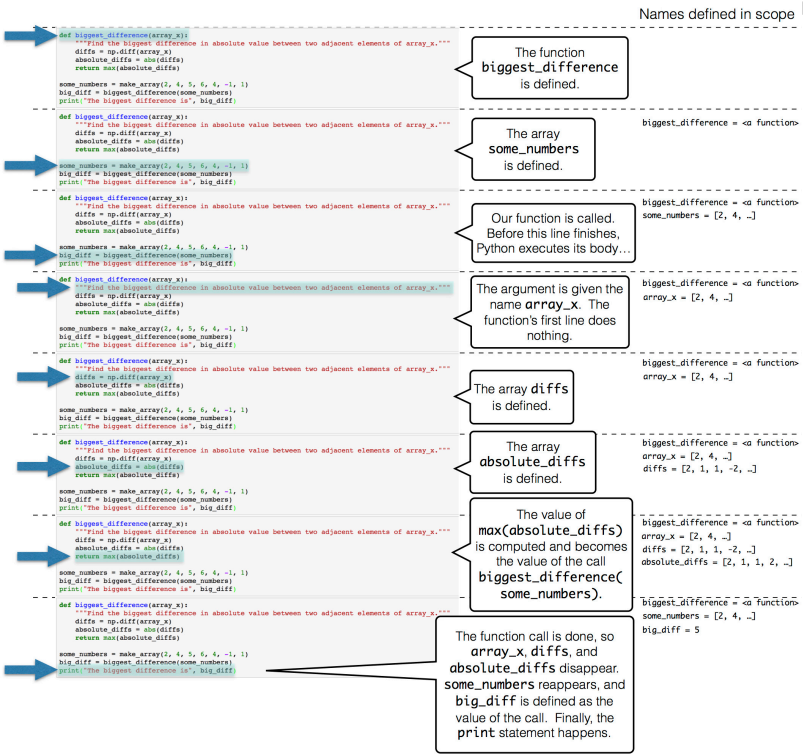
Source

DISCUSSION QUESTION

What does this function do? What kind of input does it take? What output will it give? What's a reasonable name?

```
def f(s):
    return np.round(s/sum(s) * 100, 2)
```

(Demo – notebook 4.1)



WHAT HAPPENS WHEN WE RUN THE FUNCTION?

Source

APPLY

APPLY

The **apply** method creates an array by calling a function on every element in input column(s)

- First argument: Function to apply
- Other arguments: The input column(s)

```
table_name.apply(function_name, 'column_label')
```

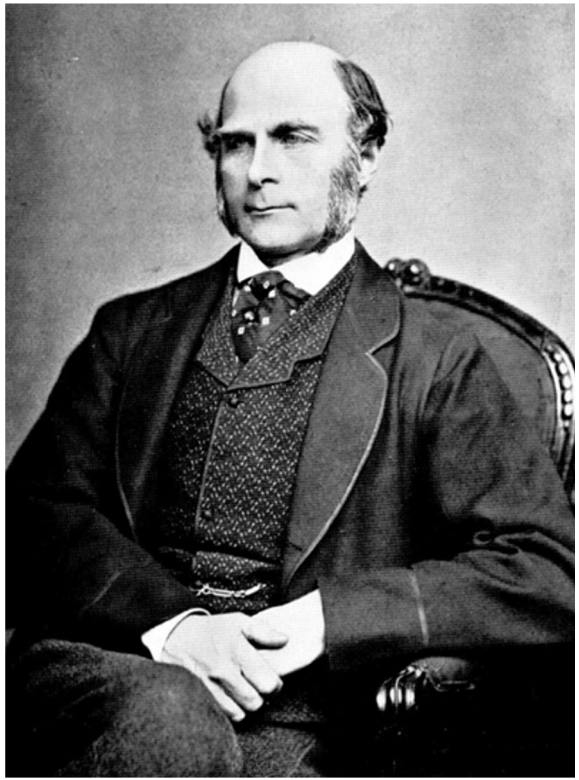
(Demo – notebook 4.1, Apply)

EXAMPLE: PREDICTION OF A CHILD'S HEIGHT

SIR FRANCIS GALTON

- 1822 - 1911 (knighted in 1909)
- A pioneer in making predictions
- Particular interest in heredity
- Charles Darwin's half-cousin

(Demo – notebook 4.1, Prediction data)



APPLY WITH MULTIPLE ARGUMENTS

APPLY

The **apply** method creates an array by calling a function on every element in one or more input columns

- First argument: Function to apply
- Other arguments: The input column(s)

```
table_name.apply(one_arg_function, 'column_label')

table_name.apply(two_arg_function,
                 'column_label_for_first_arg',
                 'column_label_for_second_arg')
```

apply called with only a function applies it to each row

(Demo – notebook 4.1,
apply with multiple columns)

GROUPING ROWS

GROUP

The **group** method aggregates all rows with the same value for a column into a single row in the result

- First argument: Which column to group by
- Second argument: (Optional) How to combine values
  - **len** — number of grouped values (default)
  - **sum** — total of all grouped values
  - **list** — list of all grouped values

(Demo – notebook 4.1, Grouping by category)

GROUPING BY TWO COLUMNS

The **group** method can also aggregate all rows that share the combination of values in multiple columns

- First argument: A list of which columns to group by
- Second argument: (Optional) How to combine values

(Demo – notebook 4.1, Grouping by category)

CHALLENGE QUESTION

Which NBA teams spent the most on their starters in 2016?

- Each team has one *starter* per position
- Assume the starter for a team & position is the player with the highest salary on that team in that position

PLAYER	POSITION	TEAM	SALARY
Paul Millsap	PF	Atlanta Hawks	18.6717
Al Horford	C	Atlanta Hawks	12
Tiago Splitter	C	Atlanta Hawks	9.75625

PIVOT TABLES

PIVOT

- **Cross-classifies** according to **two categorical** variables
- Produces a grid of counts or aggregated values
- Two required arguments:
  - First: variable that forms column labels of grid
  - Second: variable that forms row labels of grid
- Two optional arguments (include both or neither)
  - **values**='column_label_to_aggregate'
  - **collect**=function_with_which_to_aggregate

(Demo – notebook 4.1, Pivot tables)

TAKE-HOME QUESTION

Generate a table of the names of the starters for each team

TEAM	C	PF	PG	SF	SG
Atlanta Hawks	Al Horford	Paul Millsap	Jeff Teague	Thabo Sefolosha	Kyle Korver
Boston Celtics	Tyler Zeller	Jonas Jerebko	Avery Bradley	Jae Crowder	Evan Turner
Brooklyn Nets	Andrea Bargnani	Thaddeus Young	Jarrett Jack	Joe Johnson	Bojan Bogdanovic
Charlotte Hornets	Al Jefferson	Marvin Williams	Kemba Walker	Michael Kidd-Gilchrist	Nicolas Batum
Chicago Bulls	Joakim Noah	Nikola Mirotic	Derrick Rose	Doug McDermott	Jimmy Butler
Cleveland Cavaliers	Tristan Thompson	Kevin Love	Kyrie Irving	LeBron James	Iman Shumpert
Dallas Mavericks	Zaza Pachulia	David Lee	Deron Williams	Chandler Parsons	Justin Anderson
Denver Nuggets	JJ Hickson	Kenneth Faried	Jameer Nelson	Danilo Gallinari	Gary Harris
Detroit Pistons	Aron Baynes		Reggie Jackson	Stanley Johnson	Jodie Meeks
Golden State Warriors	Andrew Bogut	Draymond Green	Stephen Curry	Andre Iguodala	Klay Thompson

JOINS

JOINING TWO TABLES

```
drinks.join('Cafe', discounts, 'Location')
```

Keep all rows in the table that have a match ...

... for the value in this column ...

... somewhere in this other table's ...

... column that contains matching values.

drinks

Drink	Cafe	Price
Milk Tea	Tea One	4
Espresso	Nefeli	2
Latte	Nefeli	3
Espresso	Abe's	2

discounts

Coupon	Location
25%	Tea One
50%	Nefeli
5%	Tea One

The joined column is sorted automatically

Cafe	Drink	Price	Coupon
Nefeli	Espresso	2	50%
Nefeli	Latte	3	50%
Tea One	Milk Tea	4	25%
Tea One	Milk Tea	4	5%

(Demo – notebook 4.1, joins)

MAPS

MAPS

A table containing columns of latitude and longitude values can be used to generate a map of markers

```
_____.map_table(table, ...)
```

Either `Marker` or `Circle`

Column 0: latitudes
Column 1: longitudes
Column 2: labels
Column 3: colors
Column 4: sizes

Applies to all features:
`color='blue'`
`size=200`

Demo

COMBINING TABLE METHODS

IMPORTANT TABLE METHODS

t.select(column, ...) or **t.drop(column, ...)**
t.take([row, ...]) or **t.exclude([row, ...])**

t.sort(column, descending=False, distinct=False)
t.where(column, are.condition(...))

t.apply(function, column, ...)
t.group(column) or **t.group(column, function)**
t.group([column, ...]) or **t.group([column, ...], function)**
t.pivot(cols, rows) or t.pivot(cols, rows, vals, function)

t.join(column, other_table, other_table_column)

DISCUSSION QUESTION

Generate a table with one row per cafe that has the name and discounted price of its cheapest discounted drink

drinks

Drink	Cafe	Price
Milk Tea	Tea One	4
Espresso	Nefeli	2
Coffee	Nefeli	3
Espresso	Abe's	2

discounts

Coupon	Location
5%	Tea One
50%	Nefeli
25%	Tea One

cheapest

Cafe	Drink	Discounted Price
Nefeli	Espresso	1
Tea One	Milk Tea	3

SPRING 2016 MIDTERM, Q2(B)

(b) **(8 pt)** Each row of the `trip` table from lecture describes a single bicycle rental in the San Francisco area. Durations are integers representing times in seconds. The first three rows out of 338343 appear below.

Start	End	Duration
Ferry Building	SF Caltrain	765
San Antonio Shopping Center	Mountain View City Hall	1036
Post at Kearny	2nd at South Park	307

Write a Python expression below each of the following descriptions that computes its value. The first one is provided for you. You *may* use up to two lines and introduce variables.

- The average duration of a rental.

```
total_duration = sum(trip.column(2))
total_duration / trip.num_rows
```

- The name of the station where the most rentals ended (assume no ties).
- The number of stations for which the average duration ending at that station was more than 10 minutes.

QUESTIONS?