YData: Introduction to Data Science

Class 07: Functions

Overview

Review of:

- Using for loops to build lists of values
- NumPy arrays computations
- Discussion of images

Functions!

If there is time: Series and Tables

Announcement: Homework 3

Homework 3has been posted!

It is due on Gradescope on Sunday February 12th at 11pm

- **Be sure to mark each question on Gradescope along with the page that has the answers!**

Notes:

- Homework might be a little longer so start early (no Q&R)
- When writing functions, useful to test code outside of the function to make sure it works, then put in into a function

Announcement: Learning Groups!

“Learning groups” are informal groups of 3-4 students where you can get together and help each other out with class related material

If you are in joining a learning group, please sign-up Wednesday at 11pm

- https://docs.google.com/forms/d/11eGjb6e96i1dVk7GK9iNyFIjnQPe40OCSxVBuFBxtsU/edit?ts=63def8a2

If you have questions, please write to our course manager

- Zihe Zheng zihe.zheng@yale.edu

Questions?

Quick review of most important concepts covered so far…

1. Using for loops to build up a list of values

2. Using Boolean indexing to extract values from an ndarray

For loop practice: Egg and wheat prices

Suppose we had the monthly prices (since 1990) of:

- eggs_prices: The cost of 12 grade A eggs
- wheat_prices: The cost of a ton of wheat
  - 1 ton = 2,000 pounds

Suppose someone bought 12 eggs and a pound of wheat each month

Using for loops and lists calculate:

1. The total amount spent since 1990
2. A list containing how much was spent each month
3. Bonus: Total amount spent on wheat if it was only purchased in months when eggs were less than $2

AND JUST LIKE THAT...
EVERYONE NOW WANTS TO BE FRIENDS WITH THE CRAZY CHICKEN LADY
digitalmomblog.com

Let’s try this in Jupyter!

Review of array computations

Review of NumPy arrays and functions

Hopefully we are comfortable with:

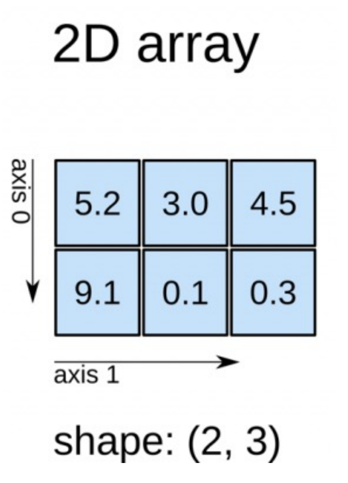
- Creating arrays and accessing elements: `np.array()`
- Getting their type and size: `.shape`, `.dtype`
- Using numeric functions: `np.sum()`, `np.mean()`, `np.diff()`
- Using broadcasting: `my_array * 2`, `my_array1 - my_array`
- Creating Boolean arrays: `my_array < 5`, `my_array == "C"`
- Using Boolean masks to get elements: `my_array[my_array < 5]`

Let's try this in Jupyter!

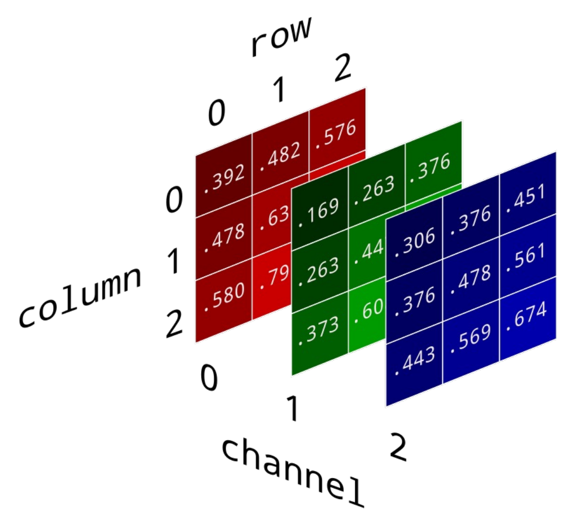
Higher dimensional arrays and images

We can make higher dimensional arrays

- my_matrix = np.array([1, 2, 3], [4, 5, 6], [7, 8, 9])
- my_matrix[0:2, 0:2]   # get a sub matrix
- np.sum(my_matrix, axis = 0)   # sum the values down rows

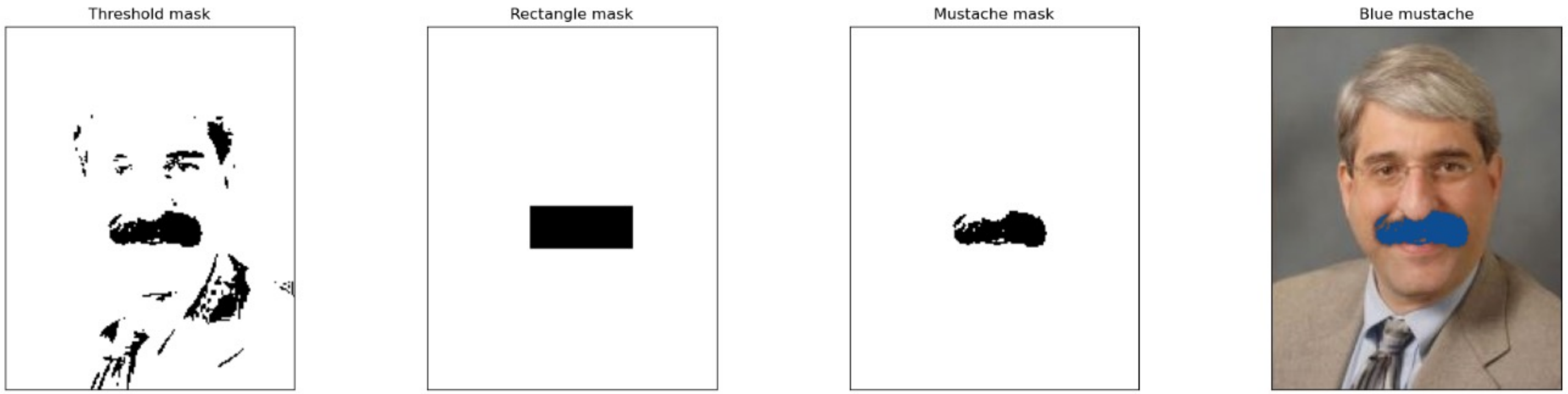


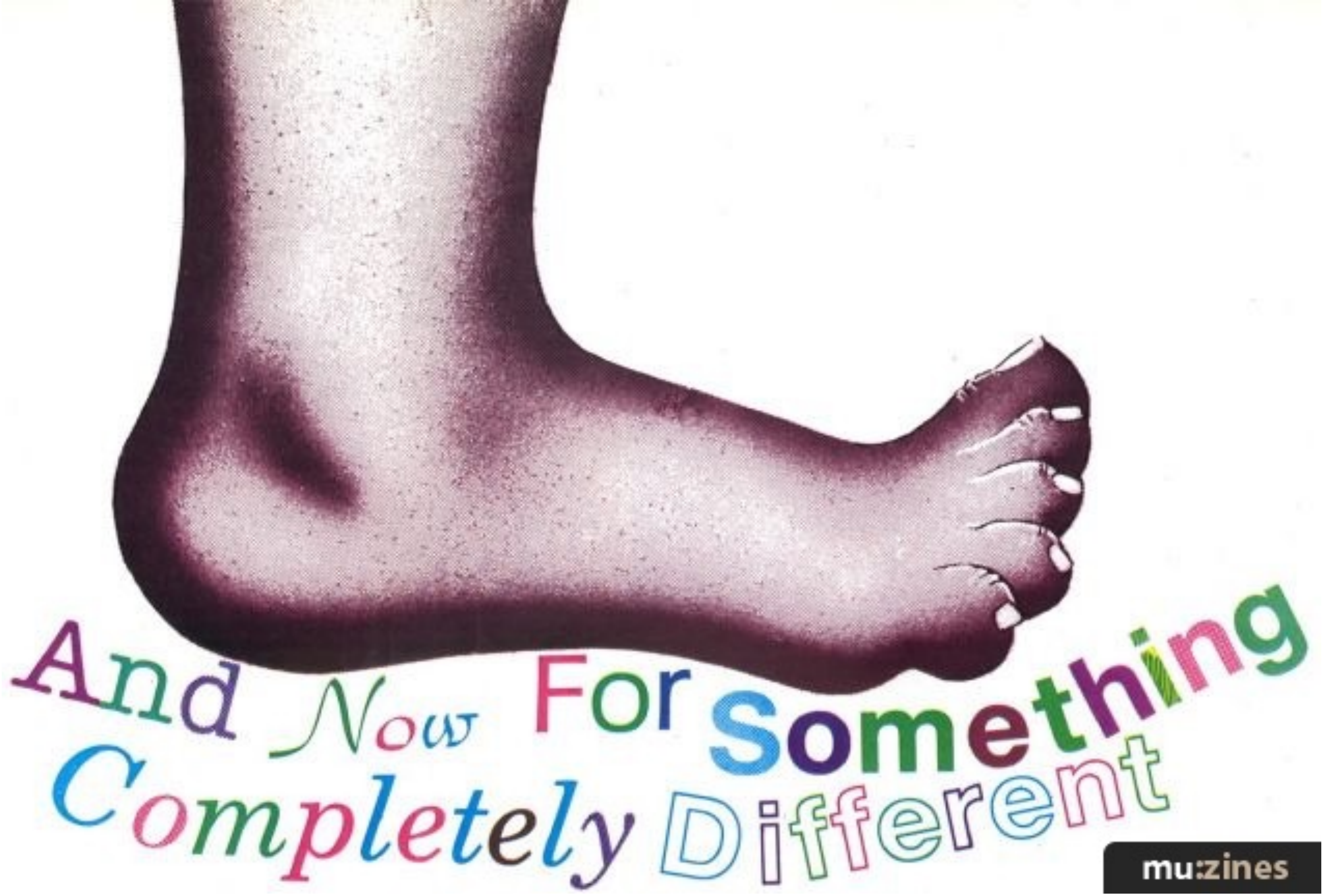
3-dimemsional numerical arrays are often used to store digital images which we can manipulate using NumPy functions



Let's take a look at some of your images!

Creating the blue 'stache




And Now For Something Completely Different

Defining functions

Writing functions

We have already used many functions that are built into Python or are imported from different modules/packages.

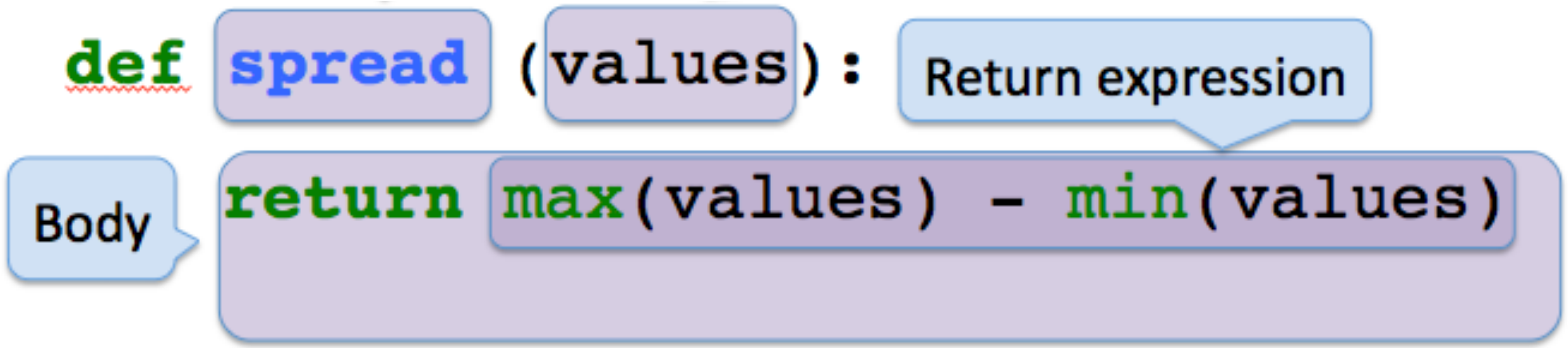
Examples…???

- sum()
- statistics.mean()
- np.diff()
- etc.

Let’s now write our own functions!

Def statements

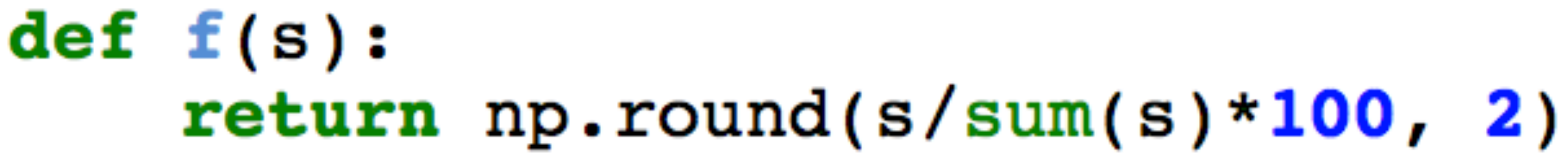
User-defined functions give names to blocks of code



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

Let's explore this in Jupyter!

Discussion questions



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

1. What does this function do?
2. What kind of input does it take?
3. What output will it give?
4. What's a reasonable name?

Let's explore this in Jupyter!

Series and Tables

Pandas: Series and DataFrames

“pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.”

There are two main data structures in pandas:

- **Series**: represent one-dimensional data
- **DataFrames**: represent data tables
  - i.e., relational data

pandas Series

pandas Series are: One-dimensional ndarray with axis labels

- (including time series)

Example: egg _prices

```
DATE
1980-01-01          0.879
1980-02-01          0.774
1980-03-01          0.812
```

Index

values

pandas Series

We can access elements by Index ***name*** using **.loc**

- egg_prices.loc["1980-01-01"]

We can access elements by Index ***number*** using **.iloc**

- egg_prices.iloc[0]

Let's explore this in Jupyter!