

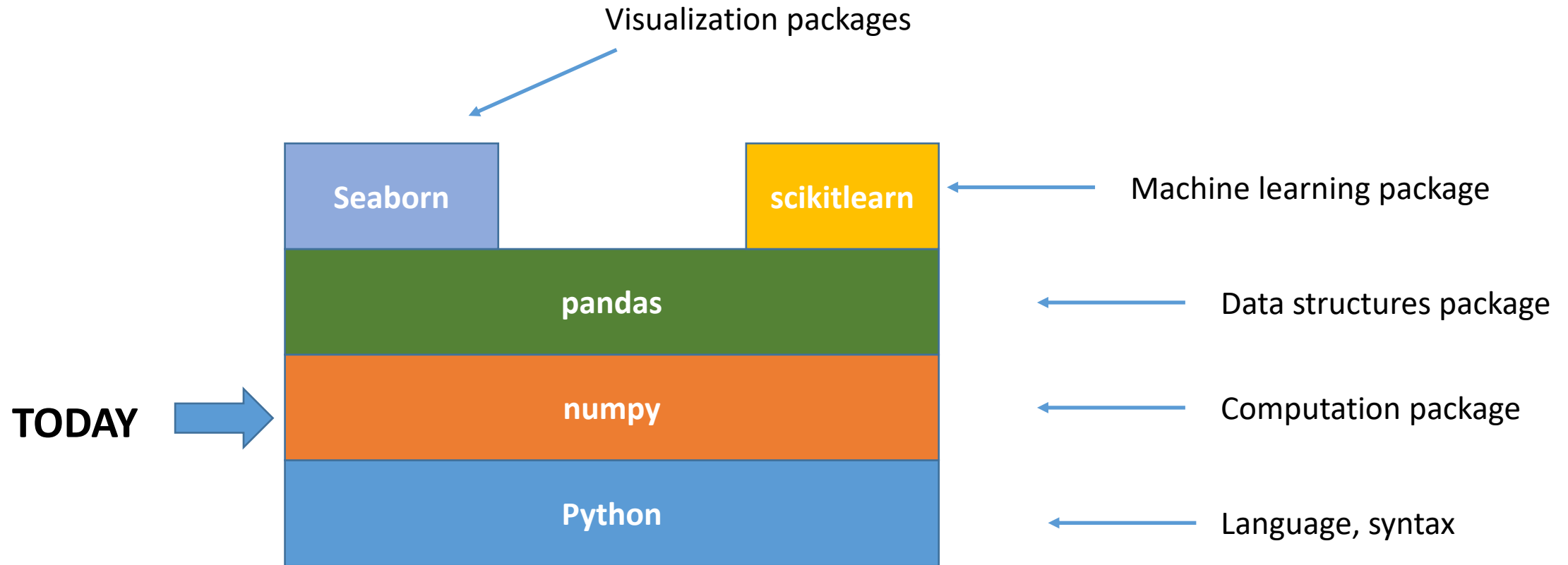
Announcements:

- **Quiz 1 - Python:** Due today! Due at 2:59pm PST
- **HW 1 - Sentiment Analysis on Twitter:** due on Wed, 4/17
- **Python Codecademy Tutorial:** due 1/28

Pandas.Series

module 2

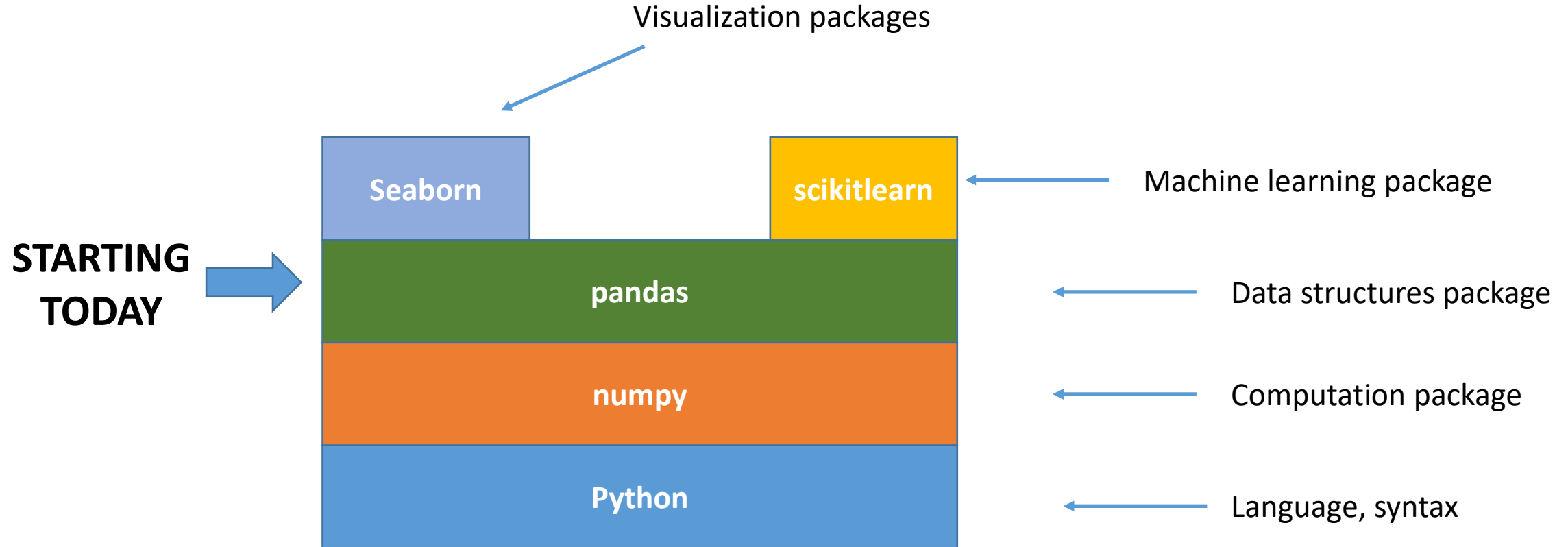
This course



Numpy – In a slide

- It is a module that provides:
 - ***ndarray***: a wrapper for Python arrays
 - ***dtype***: a wrapper for Python types
- Pandas is built on top of *numpy*. Therefore:
 - Occasionally, pandas functions return arrays – they will be *ndarray*
 - Data Types in pandas (int, float, date, ...) are *dtype*
- Today, we will see just a bit of *numpy*
- We will see more in the machine learning part

This course



Pandas– Overview

- It is the module that provides data structures and methods for data manipulation. Examples
 - Load a table from a csv file
 - Find the rows that satisfy a condition
 - Create a new calculated column
 - Group by
 - Merge tables
 - Remove null values
 - Replace YES with 1 and NO with 0
 - Many more ...

Set up

- Download the following files into the same directory:
 - *Module 02 -- pandas-Series template.ipynb*
 - *students.csv*
- In Jupyter Notebook, OPEN
Module 02 -- pandas-Series template.ipynb

Today's data set

- students.csv:
 - One student per row
 - Columns: *hw1* (the grade received on hw1), *hw2* (the grade received on hw2), and *program*.

	hw1	hw2	program
Name			
Dorian	10.0	10.0	MSIS
Jeannine	6.0	7.0	MSIS
Illuminada	2.0	NaN	MBA
Luci	7.0	7.0	MSIS
Jenny	8.0	NaN	NaN

Loading the data set

```
import pandas as pd
```

```
# print all the outputs in a cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

The two main classes in pandas are *DataFrame* and *Series*. In a nutshell, a *DataFrame* is a table and a *Series* is a column. This lecture illustrates the details of the class *Series*. First, we load the data set in *students.csv* and store it in a *DataFrame* called *df*.

```
df = pd.read_csv('students.csv', index_col=0)
```

The method **head** returns the top 5 rows of the *DataFrame*. This *DataFrame* has one student per row and three columns: *hw1* (the grade received on hw1), *hw2* (the grade received on hw2), and *program*.

```
df.head()
```

	hw1	hw2	program
Name			
Dorian	10.0	10.0	MSIS
Jeannine	6.0	7.0	MSIS
Illuminada	2.0	NaN	MBA
Luci	7.0	7.0	MSIS
Jenny	8.0	NaN	NaN

Series

hw1 Series

In this lecture, we will mostly focus only on the column *hw1*. Let's make a Series of hw1 scores.

```
hw1 = df['hw1']
```

A Series is a one-dimensional array of data (**values**) and an associated array of data labels (**index**). The **index** is the student name and the **value** is the score in hw1.

```
hw1
```

```
Name
Dorian      10.0
Jeannine     6.0
Iluminada    2.0
Luci         7.0
Jenny        8.0
Demetria     2.0
Michael      6.0
Garland      9.0
Shelby       1.0
Mercy        5.0
John         NaN
Name: hw1, dtype: float64
```

The length of hw1

```
len(hw1)
```

11

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

index and values

Return the index (as an index object) and the values

Example:

```
hw1.index
```

```
Index([u'Dorian', u'Jeannine', u'Illuminada', u'Luci', u'Jenny', u'Demetria',  
       u'Michael', u'Garland', u'Shelby', u'Mercy', u'John'],  
      dtype='object', name=u'Name')
```

```
hw1.values
```

```
array([ 10.,  6.,  2.,  7.,  8.,  2.,  6.,  9.,  1.,  5., nan])
```

ndarray

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

describe

The method **describe** reports summary statistics of the Series values.

```
hw1.describe()
```

```
count      10.000000
mean        5.600000
std         3.098387
min         1.000000
25%         2.750000
50%         6.000000
75%         7.750000
max         10.000000
Name: hw1, dtype: float64
```

10 non-null values

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Aggregate Functions

Mean, sum, abs, cumsum, etc

Example:

```
hw1.mean()
```

```
5.5999999999999996
```

The median grade

```
hw1.median()
```

```
6.0
```

The minimum and maximum grade among all students

```
hw1.min()
```

```
1.0
```

```
hw1.max()
```

```
10.0
```

POSITIONAL INDEX

INDEX

VALUES

0
1
2
3
4
5
6
7
8
9
10

Dorian
Jeannine
Illuminada
Luci
Jenny
Demetria
Michael
Garland
Shelby
Mercy
John

10.0
6.0
2.0
7.0
8.0
2.0
6.0
9.0
1.0
5.0
NaN

Name: hw1, dtype: float64

iloc (using a single integer)

Using one index value

Access the 4-th value. It returns one value.


```
hw1.iloc[3]
```

7.0

POSITIONAL INDEX

INDEX

VALUES



0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

iloc (using a slice of integers)

POSITIONAL INDEX

INDEX

VALUES

Using slices

Retrieve all elements from the 3rd (included) to the 7th (excluded). It returns a Series.

```
hw1.iloc[2:6]
```

```
Name
Illuminada    2.0
Luci          7.0
Jenny         8.0
Demetria      2.0
Name: hw1, dtype: float64
```

Caution! It returns a view, not a copy

POSITIONAL INDEX	INDEX	VALUES
0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Index-based selection: a single label value

Find Luci's hw1 grade.

```
hw1['Luci']
```

7.0

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Index-based selection: a slice of index labels

Find the grades from Luci's to Michael's

```
hw1['Luci':'Michael']
```

Name	
Luci	7.0
Jenny	8.0
Demetria	2.0
Michael	6.0

Name: hw1, dtype: float64

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Index-based selection: a sequence of Booleans

POSITIONAL INDEX

```
v = [True, False, True, True, False, True, False, True, True, False, False]
```

```
hw1[v]
```

Name

Dorian 10.0

Iluminada 2.0

Luci 7.0

Demetria 2.0

Garland 9.0

Shelby 1.0

Name: hw1, dtype: float64

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Boolean Selection (1/2)

POSITIONAL INDEX

Creates a filter (or mask) that selects those rows whose value satisfy a condition

Example:

Problem: Find the students whose grade is greater than or equal to 6

First, create a boolean Series

```
hw1 >= 6
```

Name	
Dorian	True
Jeannine	True
Iluminada	False
Luci	True
Jenny	True
Demetria	False
Michael	True
Garland	True
Shelby	False
Mercy	False
John	False

Name: hw1, dtype: bool

← All rows whose value is >= 6

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Boolean Selection (2/2)

Creates a filter (or mask) that selects those rows whose value satisfy a condition

Example:

Second, select only those students who have a "True" in the boolean Series above

```
hw1[hw1 >= 6]
```

```
Name
Dorian      10.0
Jeannine     6.0
Luci         7.0
Jenny        8.0
Michael      6.0
Garland      9.0
Name: hw1, dtype: float64
```

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Problems in class

1. What's Michael's hw1 score?
2. Select the “last” student of the Series (i.e., the one reported last). Make sure to retrieve both the name and the grade.
3. Compute the average hw1 grade among those students whose grade is less than or equal to 6
4. (together) Select those students whose hw1 score is less than 5 or greater than 9

rank()

Ranks each row based on the value

Example:

```
hw1.rank()

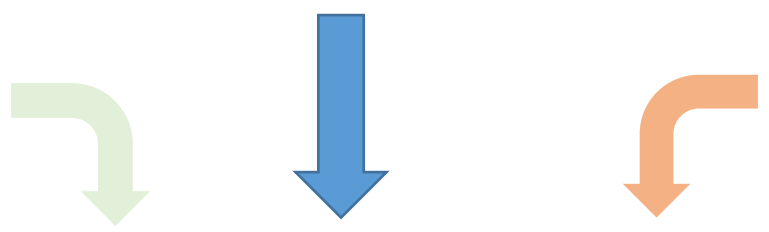
Name
Dorian      10.0
Jeannine     5.5
Iluminada    2.5
Luci         7.0
Jenny        8.0
Demetria     2.5
Michael      5.5
Garland      9.0
Shelby       1.0
Mercy        4.0
John         NaN
Name: hw1, dtype: float64
```

Notice any problem?

POSITIONAL INDEX

INDEX

VALUES



The diagram illustrates the components of the rank() function. A green arrow labeled 'POSITIONAL INDEX' points to a column of indices from 0 to 10. A blue arrow labeled 'INDEX' points to a column of names. An orange arrow labeled 'VALUES' points to a column of numerical values. The names and values are aligned with their respective indices.

0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

idxmax() and idxmin()

Find the index of the row with maximum and minimum values

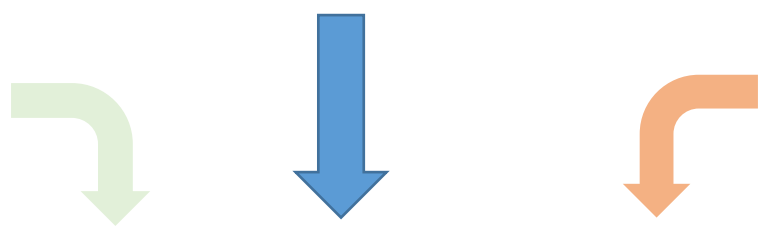
Example:

```
hw1.idxmax()  
'Dorian'  
  
hw1.idxmin()  
'Shelby'
```

POSITIONAL INDEX

INDEX

VALUES



0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

sort_values()

Sort by values

Example:

```
hw1.sort_values()
```

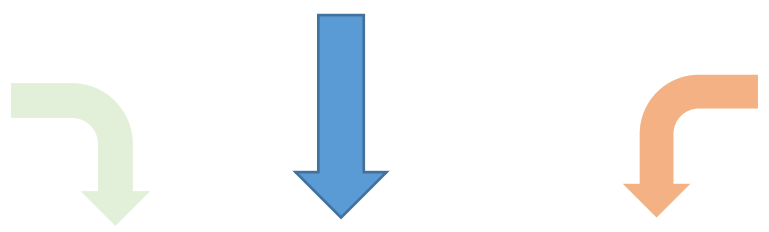
Name	
Shelby	1.0
Iluminada	2.0
Demetria	2.0
Mercy	5.0
Jeannine	6.0
Michael	6.0
Luci	7.0
Jenny	8.0
Garland	9.0
Dorian	10.0
John	NaN

Name: hw1, dtype: float64

POSITIONAL INDEX

INDEX

VALUES



0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

sort_index()

Sort by index

Example:

```
hw1.sort_index()
```

Name	
Demetria	2.0
Dorian	10.0
Garland	9.0
Illuminada	2.0
Jeannine	6.0
Jenny	8.0
John	NaN
Luci	7.0
Mercy	5.0
Michael	6.0
Shelby	1.0

Name: hw1, dtype: float64

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

nlargest(n) and nsmallest(n)

Finds the n items with largest or smallest value

Example:

```
hw1.nlargest(3)

Name
Dorian      10.0
Garland      9.0
Jenny        8.0
Name: hw1, dtype: float64

hw1.nsmallest(3)

Name
Shelby       1.0
Illuminada   2.0
Demetria     2.0
Name: hw1, dtype: float64
```

POSITIONAL INDEX

INDEX

VALUES

The diagram illustrates the relationship between three columns of data. A green arrow labeled 'POSITIONAL INDEX' points to a column of indices from 0 to 10. A blue arrow labeled 'INDEX' points to a column of names. An orange arrow labeled 'VALUES' points to a column of numerical values. The data is as follows:

0	Dorian	10.0
1	Jeannine	6.0
2	Illuminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

head(n) and tail(n)

Returns the first (or last) rows according to the positional index

Example:

```
hw1.head(3)

Name
Dorian      10.0
Jeannine     6.0
Iluminada    2.0
Name: hw1, dtype: float64

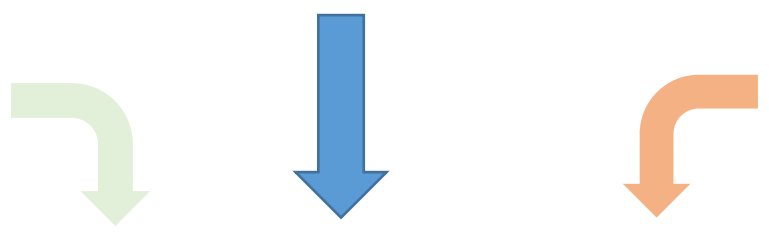
hw1.tail(3)

Name
Shelby      1.0
Mercy       5.0
John        NaN
Name: hw1, dtype: float64
```

POSITIONAL INDEX

INDEX

VALUES



The diagram illustrates the relationship between the positional index, the index, and the values of a pandas Series. A green arrow labeled 'POSITIONAL INDEX' points to a column of indices from 0 to 10. A blue arrow labeled 'INDEX' points to a column of names. An orange arrow labeled 'VALUES' points to a column of numerical values. The data is as follows:

0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Problems in class

1. Explore the parameters of the method "rank" to solve this question. Find the rank of each student (1=best, 10=worst) and deal with ties in the way that makes most sense to you.
2. Who got the 4th highest grade?
3. Retrieve the row of the person who comes last in alphabetical order.
4. Among those students whose name starts with 'J', who got the highest grade?

Operations between a scalar and a Series

Performed element-by-element

Example:

hw1 + 5	
Name	
Dorian	15.0
Jeannine	11.0
Iluminada	7.0
Luci	12.0
Jenny	13.0
Demetria	7.0
Michael	11.0
Garland	14.0
Shelby	6.0
Mercy	10.0
John	NaN
Name: hw1, dtype: float64	

hw1 * 2	
Name	
Dorian	20.0
Jeannine	12.0
Iluminada	4.0
Luci	14.0
Jenny	16.0
Demetria	4.0
Michael	12.0
Garland	18.0
Shelby	2.0
Mercy	10.0
John	NaN
Name: hw1, dtype: float64	

POSITIONAL INDEX

INDEX

VALUES

	INDEX	VALUES
0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN
Name: hw1, dtype: float64		

astype: Convert the values of a Series

Sometimes it is useful to convert a series to another type. For instance, convert a numeric series into a series of strings (`np.str`) or convert a series of text into dates (`np.datetime64`). Here is how to convert a Series of floats to a Series of string.

```
import numpy as np

hw1.astype(np.str)
```

Name	
Dorian	10.0
Jeannine	6.0
Iluminada	2.0
Luci	7.0
Jenny	8.0
Demetria	2.0
Michael	6.0
Garland	9.0
Shelby	1.0
Mercy	5.0
John	nan

Name: hw1, dtype: object

```
hw1.astype(np.str) + 'aaa'
```

Name	
Dorian	10.0aaa
Jeannine	6.0aaa
Iluminada	2.0aaa
Luci	7.0aaa
Jenny	8.0aaa
Demetria	2.0aaa
Michael	6.0aaa
Garland	9.0aaa
Shelby	1.0aaa
Mercy	5.0aaa
John	nanaaa

Name: hw1, dtype: object

POSITIONAL INDEX

INDEX

VALUES

0	Dorian	10.0
1	Jeannine	6.0
2	Iluminada	2.0
3	Luci	7.0
4	Jenny	8.0
5	Demetria	2.0
6	Michael	6.0
7	Garland	9.0
8	Shelby	1.0
9	Mercy	5.0
10	John	NaN

Name: hw1, dtype: float64

Operations between two Series

Performed between elements with the same index.

May result in NaN

Example:

Students have taken hw2. But some of them have dropped out right after taking hw1

hw1	
Name	
Dorian	10.0
Jeannine	6.0
Iluminada	2.0
Luci	7.0
Jenny	8.0
Demetria	2.0
Michael	6.0
Garland	9.0
Shelby	1.0
Mercy	5.0
John	NaN
Name: hw1, dtype: float64	

hw2	
Name	
Dorian	10.0
Jeannine	7.0
Iluminada	NaN
Luci	7.0
Jenny	NaN
Demetria	4.0
Michael	10.0
Garland	1.0
Shelby	10.0
Mercy	6.0
John	10.0
Name: hw2, dtype: float64	

Compute the average between the two hws

(hw1 + hw2) / 2	
Name	
Dorian	10.0
Jeannine	6.5
Iluminada	NaN
Luci	7.0
Jenny	NaN
Demetria	3.0
Michael	8.0
Garland	5.0
Shelby	5.5
Mercy	5.5
John	NaN
dtype: float64	

Problems in class

1. The average grade of hw1 is too low. We want to normalize it to 8. To this end, increase everyone's grade so that the new average is 8. Note that some students' grade might become greater than 10 – don't worry about it.
2. Compute the average grade of each student between hw1 and hw2. Which student has the average closest to 6.7?

(solutions on notebook)

Tech Note: double bracket vs single bracket

- What's the differences between `hw1['Michael']` and `hw1[['Michael']]` ?
 - `type(hw1['Michael'])` is a `numpy.float64`
 - `type(hw1[['Michael']])` is a `pandas.core.series.Series`

```
hw1[ 'Michael' ]
```

```
6.0
```

```
type(hw1[ 'Michael' ])
```

```
numpy.float64
```

```
hw1[ [ 'Michael' ] ]
```

```
Name
```

```
Michael      6.0
```

```
Name: hw1, dtype: float64
```

```
type(hw1[ [ 'Michael' ] ])
```

```
pandas.core.series.Series
```

Tech Note: double bracket vs single bracket – cont

- What's the differences between `df['hw1']` and `df[['hw1']]` ?
 - `type(df['hw1'])` is a `pandas.core.series.Series`
 - `type(df[['hw1']])` is a `pandas.core.frame.DataFrame`

```
df['hw1']
```

```
Name
Dorian      10.0
Jeannine     6.0
Iluminada    2.0
Luci         7.0
Jenny        8.0
Demetria     2.0
Michael      6.0
Garland      9.0
Shelby       1.0
Mercy        5.0
John         NaN
Name: hw1, dtype: float64
```

```
type(df['hw1'])
```

```
pandas.core.series.Series
```

```
df[['hw1']]
```

	hw1
Name	
Dorian	10.0
Jeannine	6.0
Iluminada	2.0
Luci	7.0
Jenny	8.0
Demetria	2.0
Michael	6.0
Garland	9.0
Shelby	1.0
Mercy	5.0
John	NaN

```
type(df[['hw1']])
```

```
pandas.core.frame.DataFrame
```