Introduction to Data Science

GIRI NARASIMHAN, SCIS, FIU

NumPy: numerical computing packages

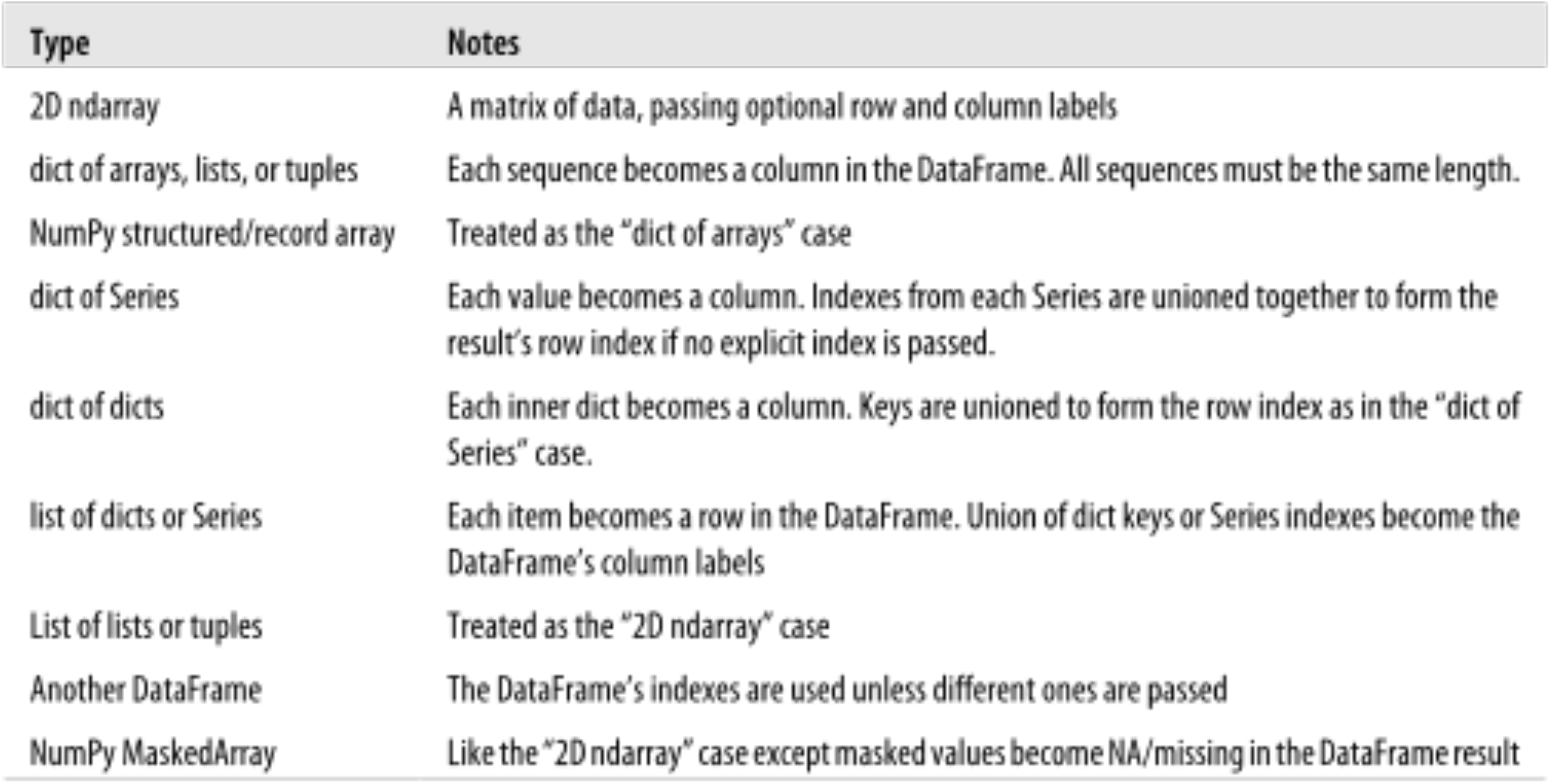
- Fast and efficient multidimensional array object ***ndarray***
- Functions for **element-wise array computations** and array operations
- Tools for **reading and writing** array-based data sets to disk
- **Linear algebra operations**, **Fourier transform**, and **random number generation**
- Tools for integrating connecting C, C++, and Fortran code to Python
- NumPy arrays are **more efficient** way of storing and manipulating data and better for passing between algorithms. Libraries in C or Fortran can operate on NumPy arrays without copying any data.

Pandas: package for structured data

- **DataFrame**: more general than R's data.frame
- Combines NumPy arrays with manipulations similar to spreadsheets and relational databases
- Sophisticated **indexing** facilities
- **Reshape**, **slice** and **dice**, **aggregations**, **subselections**, etc.
- **Time series** processing functionality

pandas DataFrames

Table 5-1. Possible data inputs to DataFrame constructor



Type	Notes
2D ndarray	A matrix of data, passing optional row and column labels
dict of arrays, lists, or tuples	Each sequence becomes a column in the DataFrame. All sequences must be the same length.
NumPy structured/record array	Treated as the "dict of arrays" case
dict of Series	Each value becomes a column. Indexes from each Series are unioned together to form the result's row index if no explicit index is passed.
dict of dicts	Each inner dict becomes a column. Keys are unioned to form the row index as in the "dict of Series" case.
list of dicts or Series	Each item becomes a row in the DataFrame. Union of dict keys or Series indexes become the DataFrame's column labels
List of lists or tuples	Treated as the "2D ndarray" case
Another DataFrame	The DataFrame's indexes are used unless different ones are passed
NumPy MaskedArray	Like the "2D ndarray" case except masked values become NA/missing in the DataFrame result

Index objects

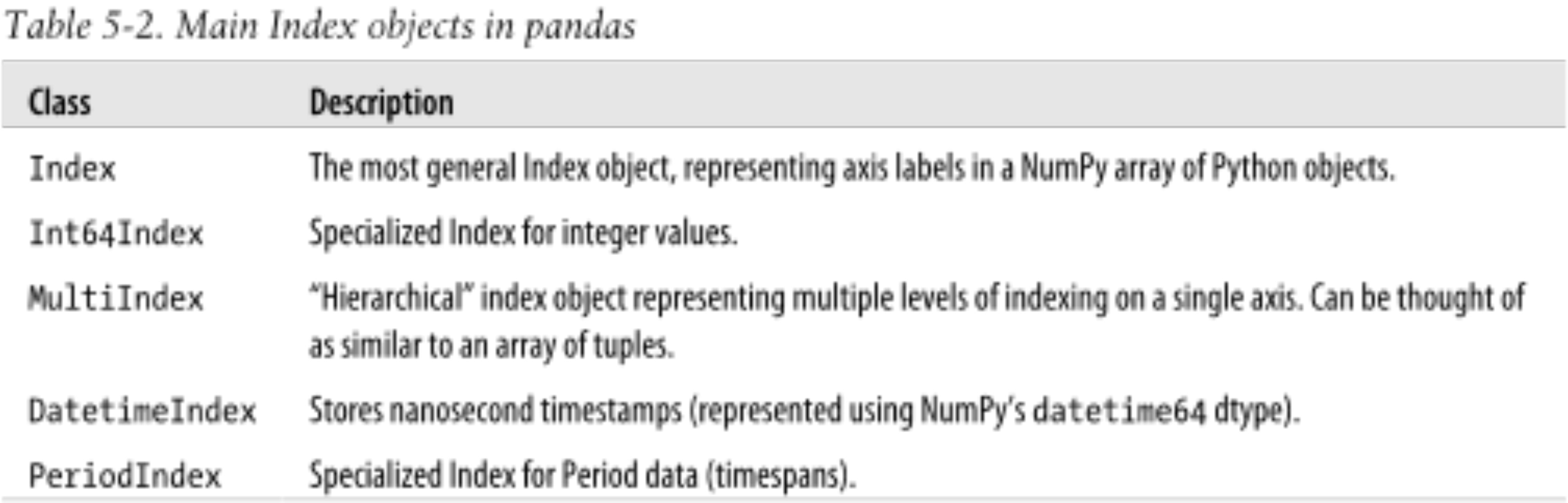


Table 5-2. Main Index objects in pandas

Class	Description
`Index`	The most general Index object, representing axis labels in a NumPy array of Python objects.
`Int64Index`	Specialized Index for integer values.
`MultiIndex`	"Hierarchical" index object representing multiple levels of indexing on a single axis. Can be thought of as similar to an array of tuples.
`DatetimeIndex`	Stores nanosecond timestamps (represented using NumPy's `datetime64` dtype).
`PeriodIndex`	Specialized Index for Period data (timespans).

More on Index

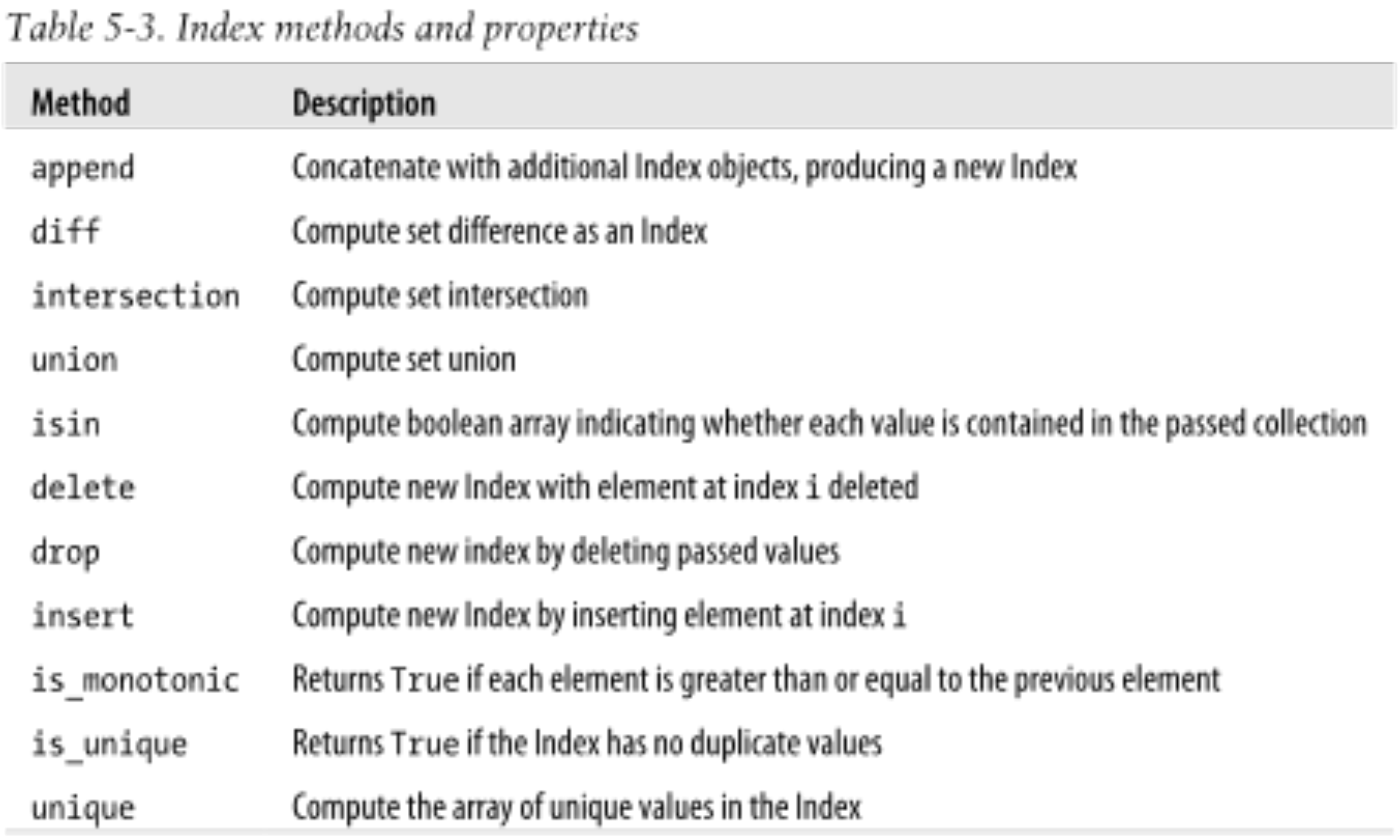


Table 5-3. Index methods and properties

Method	Description
`append`	Concatenate with additional Index objects, producing a new Index
`diff`	Compute set difference as an Index
`intersection`	Compute set intersection
`union`	Compute set union
`isin`	Compute boolean array indicating whether each value is contained in the passed collection
`delete`	Compute new Index with element at index `i` deleted
`drop`	Compute new index by deleting passed values
`insert`	Compute new Index by inserting element at index `i`
`is_monotonic`	Returns `True` if each element is greater than or equal to the previous element
`is_unique`	Returns `True` if the Index has no duplicate values
`unique`	Compute the array of unique values in the Index

SciPy: scientific computing packages

- scipy.**integrate**: numerical integration routines and differential equation solvers
- scipy.**linalg**: linear algebra, matrix decompositions extending beyond numpy.linalg.
- scipy.**optimize**: function optimizers (minimizers) and root finding algorithms
- scipy.**signal**: signal processing tools
- scipy.**sparse**: sparse matrices and sparse linear system solvers
- scipy.**special**: wrapper around SPECFUN, a Fortran library implementing many common mathematical functions, such as the gamma function
- scipy.**stats**: standard continuous and discrete probability distributions (density functions, samplers, continuous distribution functions), various statistical tests, and more descriptive statistics
- scipy.**weave**: tool for using inline C++ code to accelerate array computations

matplotlib: for visualization

- **Matplotlib:** Python library for publication-quality visualizations
- Creator: John D. Hunter, but maintained by team of developers
- Can be used in **notebooks** with ***interactive*** features; zoom in on section of plot and pan around using the toolbar in plot window.

Database *Join* (Python merge)

```
unames = ['user_id', 'gender', 'age', 'occupation', 'zip']
users = pd.read_table('data/ml-1m/users.dat', names=unames)
rnames = ['user_id', 'movie_id', 'rating', 'timestamp']
ratings = pd.read_table('data/ml-1m/ratings.dat', names=rnames)
mnames = ['movie_id', 'title', 'genres']
movies = pd.read_table('data/ml-1m/movies.dat’, names=mnames)
```

Summarization: Example

- MovieLens1M.ipynb

The DataFrame

	A	B	C	D
0	foo	one	small	1
1	foo	one	large	2
2	foo	one	large	2
3	foo	two	small	3
4	foo	two	small	3
5	bar	one	large	4
6	bar	one	small	5
7	bar	two	small	6
8	bar	two	large	7

- Rows -> Axis 0
- Columns -> Axis 1
- df[“C”]
- df.iloc[3]
- df.iloc[6][“A”]

Chain Indexing

- df.iloc[6][“A”] is an example of **chain indexing** and is considered bad Python practice

Missing Values

- Python uses NaN to indicate missing values as it reads in
- This feature can be turned off
- Missing values can be filled in with other default values
- ForwardFill and BackwardFill propagate next or previous values in table

Scales

- **Ratio** Scale: equally spaced with valid +/1; e.g. height
- **Interval** Scale: equally spaced, but zero has specific meaning; e.g. temp
- **Ordinal** Scale: ordered values, but not equally spaced; e.g. grades
- **Nominal** Scale: categorized, no order; e.g., Countries

- Can convert one to another
  - ❑ Grades could be nominal/categorical
  - ❑ Can be converted to ordinal or ratio
- Can also convert numerical values to categorical
  - ❑ Discretization
  - ❑ Histograms
- Use cut feature in pandas

Python and SQL

- SQL is a query language used to query relational databases
- SELECT operation
  - ❑ SELECT [ ] FROM [ ] WHERE [ ]

- Python notebooks allow for SQL queries to be incorporated
- query = """SELECT **fields**
  FROM **Rel**
  WHERE **conds**
  '"'’’
- df = **Rel**.query_to_pandas(query)

Google's BigQuery

- Google's serverless enterprise data warehouse with security
- Infrastructure by Google to create logical data warehouse
- Allows scalable data analytics and ML tools at good price-performance
- Uses SQL without need for database administrator
- Allows relational DB, spreadsheets, objects DB, and ODBC/JDBC drivers
- Makes it easy to join public or commercial datasets with local datasets
- Columnar & cloud storage, parallel execution, automatic optimizations
- Supports popular BI tools like Tableau, MicroStrategy, Looker, and Data Studio$^{\text{BETA}}$ out of the box

Let's try BigQuery

- BigQuery is a database that lets you use SQL to work with very large datasets.
- Open link: https://www.Kaggle.com/kernels/fork/1058477 in a new tab
- After logging in, upload the Python notebook sql2py.ipynb and run it.
- The code, loads the Chicago_crime database.
- It then shows how to convert SQL queries into python code.

Blogs

- Planetpython.org
- Dataskeptic.com