1. [NdArray And Metrix](https://docs.scipy.org/doc/numpy-1.13.0/reference/arrays.ndarray.html) (<https://docs.scipy.org/doc/numpy-1.13.0/reference/arrays.ndarray.html>) and http://www.physics.nyu.edu/pine/pymanual/html/chap3/chap3\_arrays.html
2. [Collections](http://book.pythontips.com/en/latest/collections.html) (<http://book.pythontips.com/en/latest/collections.html>)
3. [Decorators](http://book.pythontips.com/en/latest/decorators.html) (http://book.pythontips.com/en/latest/decorators.html) Decorators are a significant part of Python. In simple words: they are functions which modify the functionality of another function. They help to make our code shorter and more Pythonic.
4. [Generators](http://book.pythontips.com/en/latest/generators.html) (<http://book.pythontips.com/en/latest/generators.html>) Generators are iterators, but you can only iterate over them once. It’s because they do not store all the values in memory, they generate the values on the fly. You use them by iterating over them, either with a ‘for’ loop or by passing them to any function or construct that iterates. Most of the time generators are implemented as functions. However, they do not return a value, they yieldit.

Example: **def** **generator\_function**():

**for** i **in** range(10):

**yield** i

**for** item **in** generator\_function():

print(item)

1. [Database In Python](https://wiki.python.org/moin/DatabaseInterfaces) (<https://wiki.python.org/moin/DatabaseInterfaces>)
2. [Python Pandas](https://www.tutorialspoint.com/python_pandas/index.htm) Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. Pandas library uses most of the functionalities of NumPy. It is suggested that you go through our tutorial on NumPy before proceeding
   * 1. [Python Pandas - Home](https://www.tutorialspoint.com/python_pandas/index.htm)
     2. [Python Pandas - Introduction](https://www.tutorialspoint.com/python_pandas/python_pandas_introduction.htm)
     3. [Python Pandas - Environment Setup](https://www.tutorialspoint.com/python_pandas/python_pandas_environment_setup.htm)
     4. [Introduction to Data Structures](https://www.tutorialspoint.com/python_pandas/python_pandas_introduction_to_data_structures.htm)
     5. [Python Pandas - Series](https://www.tutorialspoint.com/python_pandas/python_pandas_series.htm)
     6. [Python Pandas - DataFrame](https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm)

A pandas DataFrame can be created using various inputs like −

1. Lists
2. dict
3. Series
4. Numpy ndarrays
5. Another DataFrame
   * 1. [Python Pandas - Panel](https://www.tutorialspoint.com/python_pandas/python_pandas_panel.htm)
     2. [Python Pandas - Basic Functionality](https://www.tutorialspoint.com/python_pandas/python_pandas_basic_functionality.htm)
     3. [Descriptive Statistics](https://www.tutorialspoint.com/python_pandas/python_pandas_descriptive_statistics.htm)
     4. [Function Application](https://www.tutorialspoint.com/python_pandas/python_pandas_function_application.htm)
     5. [Python Pandas - Reindexing](https://www.tutorialspoint.com/python_pandas/python_pandas_reindexing.htm)
     6. [Python Pandas - Iteration](https://www.tutorialspoint.com/python_pandas/python_pandas_iteration.htm)
     7. [Python Pandas - Sorting](https://www.tutorialspoint.com/python_pandas/python_pandas_sorting.htm)
     8. [Working with Text Data](https://www.tutorialspoint.com/python_pandas/python_pandas_working_with_text_data.htm)
     9. [Options & Customization](https://www.tutorialspoint.com/python_pandas/python_pandas_options_and_customization.htm)
     10. [Indexing & Selecting Data](https://www.tutorialspoint.com/python_pandas/python_pandas_indexing_and_selecting_data.htm)
     11. [Statistical Functions](https://www.tutorialspoint.com/python_pandas/python_pandas_statistical_functions.htm)
     12. [Python Pandas - Window Functions](https://www.tutorialspoint.com/python_pandas/python_pandas_window_functions.htm)
     13. [Python Pandas - Aggregations](https://www.tutorialspoint.com/python_pandas/python_pandas_aggregations.htm)
     14. [Python Pandas - Missing Data](https://www.tutorialspoint.com/python_pandas/python_pandas_missing_data.htm)
     15. [Python Pandas - GroupBy](https://www.tutorialspoint.com/python_pandas/python_pandas_groupby.htm)
     16. [Python Pandas - Merging/Joining](https://www.tutorialspoint.com/python_pandas/python_pandas_merging_joining.htm)
     17. [Python Pandas - Concatenation](https://www.tutorialspoint.com/python_pandas/python_pandas_concatenation.htm)
     18. [Python Pandas - Date Functionality](https://www.tutorialspoint.com/python_pandas/python_pandas_date_functionality.htm)
     19. [Python Pandas - Timedelta](https://www.tutorialspoint.com/python_pandas/python_pandas_timedelta.htm)
     20. [Python Pandas - Categorical Data](https://www.tutorialspoint.com/python_pandas/python_pandas_categorical_data.htm)
     21. [Python Pandas - Visualization](https://www.tutorialspoint.com/python_pandas/python_pandas_visualization.htm)
     22. [Python Pandas - IO Tools](https://www.tutorialspoint.com/python_pandas/python_pandas_io_tool.htm)
     23. [Python Pandas - Sparse Data](https://www.tutorialspoint.com/python_pandas/python_pandas_sparse_data.htm)
     24. [Python Pandas - Caveats & Gotchas](https://www.tutorialspoint.com/python_pandas/python_pandas_caveats_and_gotchas.htm)
     25. [Comparison with SQL](https://www.tutorialspoint.com/python_pandas/python_pandas_comparison_with_sql.htm)
6. [NumPy](https://www.tutorialspoint.com/numpy/index.htm) NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. It is specifically useful for algorithm developers. A basic understanding of Python is required.
   * 1. [NumPy - Home](https://www.tutorialspoint.com/numpy/index.htm)
     2. [NumPy - Introduction](https://www.tutorialspoint.com/numpy/numpy_introduction.htm)
     3. [NumPy - Environment](https://www.tutorialspoint.com/numpy/numpy_environment.htm)
     4. [NumPy - Ndarray Object](https://www.tutorialspoint.com/numpy/numpy_ndarray_object.htm)
     5. [NumPy - Data Types](https://www.tutorialspoint.com/numpy/numpy_data_types.htm)
     6. [NumPy - Array Attributes](https://www.tutorialspoint.com/numpy/numpy_array_attributes.htm)
     7. [NumPy - Array Creation Routines](https://www.tutorialspoint.com/numpy/numpy_array_creation_routines.htm)
     8. [NumPy - Array from Existing Data](https://www.tutorialspoint.com/numpy/numpy_array_from_existing_data.htm)
     9. [Array From Numerical Ranges](https://www.tutorialspoint.com/numpy/numpy_array_from_numerical_ranges.htm)
     10. [NumPy - Indexing & Slicing](https://www.tutorialspoint.com/numpy/numpy_indexing_and_slicing.htm)
     11. [NumPy - Advanced Indexing](https://www.tutorialspoint.com/numpy/numpy_advanced_indexing.htm)
     12. [NumPy - Broadcasting](https://www.tutorialspoint.com/numpy/numpy_broadcasting.htm)
     13. [NumPy - Iterating Over Array](https://www.tutorialspoint.com/numpy/numpy_iterating_over_array.htm)
     14. [NumPy - Array Manipulation](https://www.tutorialspoint.com/numpy/numpy_array_manipulation.htm)
     15. [NumPy - Binary Operators](https://www.tutorialspoint.com/numpy/numpy_binary_operators.htm)
     16. [NumPy - String Functions](https://www.tutorialspoint.com/numpy/numpy_string_functions.htm)
     17. [NumPy - Mathematical Functions](https://www.tutorialspoint.com/numpy/numpy_mathematical_functions.htm)
     18. [NumPy - Arithmetic Operations](https://www.tutorialspoint.com/numpy/numpy_arithmetic_operations.htm)
     19. [NumPy - Statistical Functions](https://www.tutorialspoint.com/numpy/numpy_statistical_functions.htm)
     20. [Sort, Search & Counting Functions](https://www.tutorialspoint.com/numpy/numpy_sort_search_counting_functions.htm)
     21. [NumPy - Byte Swapping](https://www.tutorialspoint.com/numpy/numpy_byte_swapping.htm)
     22. [NumPy - Copies & Views](https://www.tutorialspoint.com/numpy/numpy_copies_and_views.htm)
     23. [NumPy - Matrix Library](https://www.tutorialspoint.com/numpy/numpy_matrix_library.htm)
     24. [NumPy - Linear Algebra](https://www.tutorialspoint.com/numpy/numpy_linear_algebra.htm)
     25. [NumPy - Matplotlib](https://www.tutorialspoint.com/numpy/numpy_matplotlib.htm)
     26. [NumPy - Histogram Using Matplotlib](https://www.tutorialspoint.com/numpy/numpy_histogram_using_matplotlib.htm)
     27. [NumPy - I/O with NumPy](https://www.tutorialspoint.com/numpy/numpy_with_io.htm)
7. Sql Injection : <https://teamultimate.in/sql-injection-explained/>the process of using the weakness of SQL to hack the website database is called SQL Injection. SQL does a mistake too. It has a weakness/[vulnerability](https://teamultimate.in/exploit-vulnerability-payload-explained/)used by hackers too hack database of the website.  
   SQL can accept commands even from the users using the website. Well this is how SQL Injection works, we ask the database to give us a result that it is not supposed to give to a user. <https://blog.sqreen.io/preventing-sql-injections-in-python/>
8. How to call stored Procedures in python <http://www.mysqltutorial.org/calling-mysql-stored-procedures-python/>

MySQLCursor.callproc() Method

Syntax:

result\_args = cursor.callproc(proc\_name, args=())

<https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlcursor-callproc.html>

### Is any ORM feature supported by Python [PonyORM - Python ORM with beautiful query syntax](https://ponyorm.com/)

## ORM

*Libraries that implement Object-Relational Mapping or data mapping techniques.*

* Relational Databases
  + [Django Models](https://docs.djangoproject.com/en/dev/topics/db/models/) - A part of Django.
  + **SQLAlchemy** (the most advanced ORM available for the Python language [SQLAlchemy](http://www.sqlalchemy.org/) - The Python SQL Toolkit and Object Relational Mapper.
    - [awesome-sqlalchemy](https://github.com/dahlia/awesome-sqlalchemy)
  + [Orator](https://orator-orm.com/) - The Orator ORM provides a simple yet beautiful ActiveRecord implementation.
  + [Peewee](https://github.com/coleifer/peewee) - A small, expressive ORM.
  + [PonyORM](https://ponyorm.com/) - ORM that provides a generator-oriented interface to SQL.
  + [pyDAL](https://github.com/web2py/pydal/) - A pure Python Database Abstraction Layer.
  + [python-sql](https://pypi.python.org/pypi/python-sql) - Write SQL queries pythonically.
* NoSQL Databases
  + [django-mongodb-engine](https://github.com/django-nonrel/mongodb-engine) - Django MongoDB Backend.
  + [flywheel](https://github.com/stevearc/flywheel) - Object mapper for Amazon DynamoDB.
  + [hot-redis](https://github.com/stephenmcd/hot-redis) - Rich Python data types for Redis.
  + [MongoEngine](http://mongoengine.org/) - A Python Object-Document-Mapper for working with MongoDB.
  + [PynamoDB](https://github.com/pynamodb/PynamoDB) - A Pythonic interface for [Amazon DynamoDB](https://aws.amazon.com/dynamodb/).
  + [redisco](https://github.com/kiddouk/redisco) - A Python Library for Simple Models and Containers Persisted in Redis.
* Others
  + [butterdb](https://github.com/terrible-ideas/butterdb) - A Python ORM for Google Drive Spreadsheets.
  + [dataset](https://github.com/pudo/dataset) - A JSON-based database.

## **Different design pattern and architecture in Python**

* + - MVC
    - Event-Driven Programming
    - Microservice Architecture

## XML processing module in python

<https://docs.python.org/3/library/xml.html>

# [**json**](https://docs.python.org/3/library/json.html#module-json)**— JSON encoder and decoder**

https://docs.python.org/3/library/json.html

## Python Webframeworks

https://wiki.python.org/moin/WebFrameworks

## **Data Analysis**

Libraries for data analyzing.

* [Blaze](https://github.com/blaze/blaze) - NumPy and Pandas interface to Big Data.
* [Open Mining](https://github.com/mining/mining) - Business Intelligence (BI) in Pandas interface.
* [Orange](https://orange.biolab.si/) - Data mining, data visualization, analysis and machine learning through visual programming or scripts.
* [Pandas](http://pandas.pydata.org/) - A library providing high-performance, easy-to-use data structures and data analysis tools.

## **seaborn: statistical data visualization**

Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics. https://www.datacamp.com/community/tutorials/seaborn-python-tutorial

## **Seaborn vs Matplotlib**

As you have just read, Seaborn is complimentary to Matplotlib and it specifically targets statistical data visualization. But it goes even further than that: Seaborn extends Matplotlib and that’s why it can address the two biggest frustrations of working with Matplotlib. Or, as Michael Waskom says in the “[introduction to Seaborn](http://seaborn.pydata.org/introduction.html)”: “If matplotlib “tries to make easy things easy and hard things possible”, seaborn tries to make a well-defined set of hard things easy too.”

One of these hard things or frustrations had to do with the default Matplotlib parameters. Seaborn works with different parameters, which undoubtedly speaks to those users that don’t use the default looks of the Matplotlib plots.

Compare the following plots:

* [script.py](https://www.datacamp.com/community/tutorials/seaborn-python-tutorial)

# Import the necessary libraries

import matplotlib.pyplot as plt

import pandas as pd

# Initialize Figure and Axes object

fig, ax = plt.subplots()

# Load in data

tips = pd.read\_csv("https://raw.githubusercontent.com

/mwaskom/seaborn-data/master/tips.csv")

# Create violinplot

ax.violinplot(tips["total\_bill"], vert=False)

# Show the plot

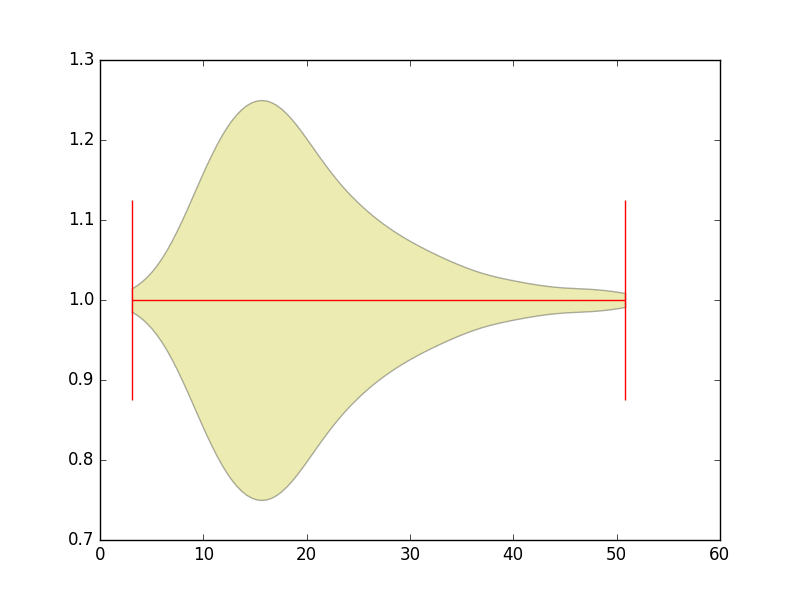
plt.show()

* [IPython Shell](https://www.datacamp.com/community/tutorials/seaborn-python-tutorial)



In [1]:

Run



* [script.py](https://www.datacamp.com/community/tutorials/seaborn-python-tutorial)



1

2

3

4

5

6

7

8

9

10

11

12

# Import the necessary libraries

import matplotlib.pyplot as plt

import seaborn as sns

# Load the data

tips = sns.load\_dataset("tips")

# Create violinplot

sns.violinplot(x = "total\_bill", data=tips)

# Show the plot

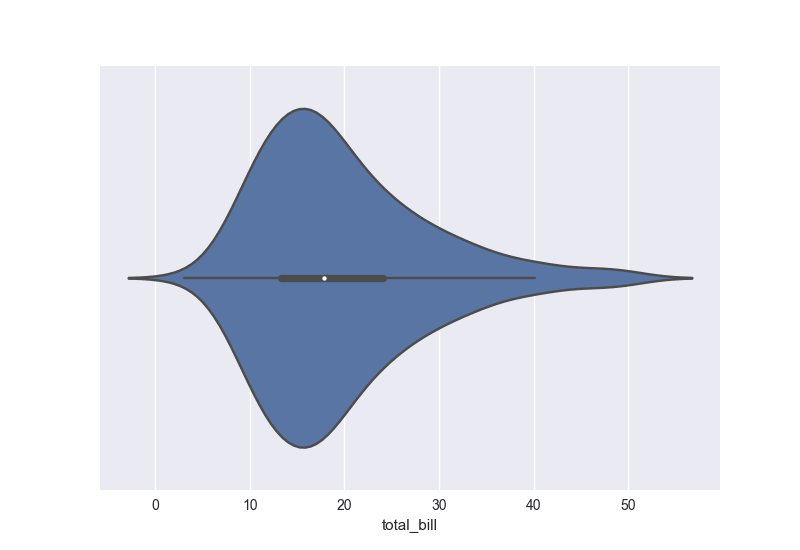
plt.show()

* [IPython Shell](https://www.datacamp.com/community/tutorials/seaborn-python-tutorial)



In [1]:

Run



The Matplotlib defaults that usually don’t speak to users are the colors, the tick marks on the upper and right axes, the style,…

The examples above also makes another frustration of users more apparent: the fact that working with DataFrames doesn’t go quite as smoothly with Matplotlib, which can be annoying if you’re doing exploratory analysis with Pandas. And that’s exactly what Seaborn addresses: the plotting functions operate on DataFrames and arrays that contain a whole dataset.

As Seaborn compliments and extends Matplotlib, the learning curve is quite gradual: if you know Matplotlib, you’ll already have most of Seaborn down.