Selecting List Elements

Import libraries
>>> import numpy
>>> import numpy as np

Selective import
>>> from ... Operations

Selecting Numpy Array Elements

Index starts at 0

Numpy Array Operations

Numpy Array Functions

Install Python

ANAconda

Leading open data science platform
powered by Python

Free IDE that is included
with Anaconda

Create and share documents with live code, visualizations, text, ...
Kernels provide computation and communication with front-end interfaces like the notebooks. There are three main kernels:

- **IPython**: Restart kernel and run all cells.
- **IRkernel**: Restart kernel and run all cells.
- **IJulia**: Connect back to a remote notebook.

Installing Jupyter Notebook will automatically install the IPython kernel.

Widgets

Notebook widgets provide the ability to visualize and control changes in your data, often as a control like a slider, textbox, etc. You can use them to build interactive GUIs for your notebooks or to synchronize stateful and stateless information between Python and JavaScript.

Download serialized state of all widget models in use.

Embed current widgets.

Save notebook with interactive widgets.

Command Mode:

- Jupyter Notebook
- MyJupyterNotebook
- Last Checkpoint: a few seconds ago (unsaved changes)

File

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>jupyter</td>
<td>MyJupyterNotebook</td>
<td>Last Checkpoint: a few seconds ago (unsaved changes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Edit Mode:

- 1. Save and checkpoint
- 2. Insert cell below
- 3. Cut cell
- 4. Copy cell(s)
- 5. Paste cell(s) below
- 6. Move cell up
- 7. Move cell down
- 8. Run current cell
- 9. Interrupt kernel
- 10. Restart kernel
- 11. Display characteristics
- 12. Open command palette
- 13. Current kernel
- 14. Kernel status
- 15. Log out from notebook server

Edit Cells

- Cut currently selected cells to clipboard
- Paste cells from clipboard above current cell
- Paste cells from clipboard on top of current cell
- Revert "Delete Cells" invocation
- Merge current cell with the one above
- Move current cell up
- Adjust metadata underlying the current notebook
- Remove cell attachments
- Paste attachments of current cell

Insert Cells

- Add new cell above the current one
- Add new cell below the current one

View Cells

- Toggle display of Jupyter logo and filename
- Toggle display of toolbar
- Toggle display of cell action icons: None, Edit metadata, Raw cell format, Slideshow, Attachments, Tags

Execute Cells

- Run selected cell(s)
- Run current cells down and create a new one below
- Run all cells
- Run all cells below the current cell
- Toggle, toggle scrolling and clear all output
- Toggle, toggle scrolling and clear current outputs

Asking For Help

- Walk through a UI tour
- Edit the built-in keyboard shortcuts
- Description of markdown available in notebook
- Python help topics
- NumPy help topics
- Matplotlib help topics
- Pandas help topics
- List of built-in keyboard shortcuts
- Notebook help topics
- Information on unofficial Jupyter Notebook extensions
- IPython help topics
- SciPy help topics
- SymPy help topics
- About Jupyter Notebook
### NumPy Basics

**Creating Arrays**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>np.array()</code></td>
<td>Create an array from a list of elements</td>
</tr>
<tr>
<td><code>np.zeros()</code></td>
<td>Create an array of a specified shape filled with zeros</td>
</tr>
<tr>
<td><code>np.empty()</code></td>
<td>Create an empty array of a specified shape</td>
</tr>
<tr>
<td><code>np.random.random()</code></td>
<td>Create an array of random numbers</td>
</tr>
<tr>
<td><code>np.linspace()</code></td>
<td>Create an array of evenly spaced values within a specified range</td>
</tr>
<tr>
<td><code>np.arange()</code></td>
<td>Create an array of evenly spaced values within a specified range</td>
</tr>
<tr>
<td><code>np.ones()</code></td>
<td>Create an array of ones of a specified shape</td>
</tr>
<tr>
<td><code>np.zeros()</code></td>
<td>Create an array of zeros of a specified shape</td>
</tr>
</tbody>
</table>

**Initial Placeholders**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>np.zeros(3,4)</code></td>
<td>Create a 3x4 array of zeros</td>
</tr>
<tr>
<td><code>np.ones(3,4)</code></td>
<td>Create a 3x4 array of ones</td>
</tr>
<tr>
<td><code>np.empty((3,2))</code></td>
<td>Create a 3x2 empty array</td>
</tr>
<tr>
<td><code>np.random.random((2,2))</code></td>
<td>Create a 2x2 array of random numbers</td>
</tr>
<tr>
<td><code>np.linspace(0,2,9)</code></td>
<td>Create an array of 9 evenly spaced values between 0 and 2</td>
</tr>
<tr>
<td><code>d = np.arange(10,25,5)</code></td>
<td>Create an array of integers from 10 to 25 with a step of 5</td>
</tr>
</tbody>
</table>

**Data Types**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>np.int8()</code></td>
<td>Convert to 8-bit integer type</td>
</tr>
<tr>
<td><code>np.int16()</code></td>
<td>Convert to 16-bit integer type</td>
</tr>
<tr>
<td><code>np.int32()</code></td>
<td>Convert to 32-bit integer type</td>
</tr>
<tr>
<td><code>np.int64()</code></td>
<td>Convert to 64-bit integer type</td>
</tr>
<tr>
<td><code>np.float16()</code></td>
<td>Convert to 16-bit floating point type</td>
</tr>
<tr>
<td><code>np.float32()</code></td>
<td>Convert to 32-bit floating point type</td>
</tr>
<tr>
<td><code>np.float64()</code></td>
<td>Convert to 64-bit floating point type</td>
</tr>
</tbody>
</table>

**Array Mathematics**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a + b</code></td>
<td>Add arrays element-wise</td>
</tr>
<tr>
<td><code>a * b</code></td>
<td>Multiply arrays element-wise</td>
</tr>
<tr>
<td><code>a / b</code></td>
<td>Divide arrays element-wise</td>
</tr>
<tr>
<td><code>a % b</code></td>
<td>Modulo operation (element-wise)</td>
</tr>
<tr>
<td><code>a ** b</code></td>
<td>Exponentiation (element-wise)</td>
</tr>
<tr>
<td><code>np.sin(a)</code></td>
<td>Compute sine of each element</td>
</tr>
<tr>
<td><code>np.cos(a)</code></td>
<td>Compute cosine of each element</td>
</tr>
<tr>
<td><code>np.tan(a)</code></td>
<td>Compute tangent of each element</td>
</tr>
<tr>
<td><code>np.exp(a)</code></td>
<td>Compute exponential of each element</td>
</tr>
<tr>
<td><code>np.log(a)</code></td>
<td>Compute natural logarithm of each element</td>
</tr>
</tbody>
</table>

**Inspecting Your Array**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>np.shape()</code></td>
<td>Return the shape of the array</td>
</tr>
<tr>
<td><code>len(a)</code></td>
<td>Return the length of the array</td>
</tr>
<tr>
<td><code>np.size(a)</code></td>
<td>Return the total number of elements in the array</td>
</tr>
<tr>
<td><code>np.dtype(a)</code></td>
<td>Return the data type of the array</td>
</tr>
<tr>
<td><code>a.ndim</code></td>
<td>Return the number of dimensions of the array</td>
</tr>
<tr>
<td><code>a.dtype</code></td>
<td>Return the data type of the array elements</td>
</tr>
<tr>
<td><code>a.dtype.name</code></td>
<td>Return the name of the data type of the array elements</td>
</tr>
</tbody>
</table>

**Array Manipulation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>np.transpose(a)</code></td>
<td>Transpose the array</td>
</tr>
<tr>
<td><code>np.reshape(a, (new_shape))</code></td>
<td>Reshape the array without changing data</td>
</tr>
<tr>
<td><code>a.ravel()</code></td>
<td>Flatten the array</td>
</tr>
<tr>
<td><code>np.delete(a, indices)</code></td>
<td>Delete elements from the array</td>
</tr>
<tr>
<td><code>np.insert(a, indices, values)</code></td>
<td>Insert values into the array at specified positions</td>
</tr>
<tr>
<td><code>np.concatenate((a, b))</code></td>
<td>Stack arrays vertically (row-wise)</td>
</tr>
<tr>
<td><code>np.hstack((a, b))</code></td>
<td>Stack arrays horizontally (column-wise)</td>
</tr>
<tr>
<td><code>np.vstack((a, b))</code></td>
<td>Stack arrays vertically (row-wise)</td>
</tr>
</tbody>
</table>

**Sorting Arrays**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.sort()</code></td>
<td>Sort the array in ascending order</td>
</tr>
<tr>
<td><code>a.sort(0)</code></td>
<td>Sort the elements of an array by their 0th axis</td>
</tr>
</tbody>
</table>

**Subsetting, Slicing, Indexing**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[0]</code></td>
<td>Select the element at the 0th index</td>
</tr>
<tr>
<td><code>a[0:2]</code></td>
<td>Select elements from index 0 to 1</td>
</tr>
<tr>
<td><code>a[:,0]</code></td>
<td>Select all elements at the 0th axis</td>
</tr>
<tr>
<td><code>a[0, 1]</code></td>
<td>Select the element at indices (0, 1)</td>
</tr>
</tbody>
</table>

**Array Operations**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a + b</code></td>
<td>Add arrays element-wise</td>
</tr>
<tr>
<td><code>a * b</code></td>
<td>Multiply arrays element-wise</td>
</tr>
<tr>
<td><code>a / b</code></td>
<td>Divide arrays element-wise</td>
</tr>
</tbody>
</table>

**Boolean Indexing**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[a &gt; 0]</code></td>
<td>Select elements greater than 0</td>
</tr>
<tr>
<td><code>a[a == 0]</code></td>
<td>Select elements equal to 0</td>
</tr>
<tr>
<td><code>a[a &lt; 0]</code></td>
<td>Select elements less than 0</td>
</tr>
</tbody>
</table>

**I/O**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>np.savetxt('filename.txt', a, delimiter=',')</code></td>
<td>Save the array to a text file</td>
</tr>
<tr>
<td><code>np.load('filename.npy')</code></td>
<td>Load the array from a binary file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>DataCamp</strong> Learn Python for Data Science Interactively at <a href="http://www.DataCamp.com">www.DataCamp.com</a></th>
<th>NumPy Basics</th>
<th>NumPy Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D array</td>
<td>2D array</td>
<td>3D array</td>
</tr>
<tr>
<td><img src="image1" alt="1D array" /></td>
<td><img src="image2" alt="2D array" /></td>
<td><img src="image3" alt="3D array" /></td>
</tr>
<tr>
<td>axis 0</td>
<td>axis 1</td>
<td>axis 2</td>
</tr>
<tr>
<td><img src="image4" alt="Axis" /></td>
<td><img src="image5" alt="Axis" /></td>
<td><img src="image6" alt="Axis" /></td>
</tr>
</tbody>
</table>

**Examining Your Array**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.shape</code></td>
<td>Return the shape of the array</td>
</tr>
<tr>
<td><code>len(a)</code></td>
<td>Return the length of the array</td>
</tr>
<tr>
<td><code>np.size(a)</code></td>
<td>Return the total number of elements in the array</td>
</tr>
<tr>
<td><code>np.dtype(a)</code></td>
<td>Return the data type of the array</td>
</tr>
<tr>
<td><code>a.ndim</code></td>
<td>Return the number of dimensions of the array</td>
</tr>
<tr>
<td><code>a.dtype</code></td>
<td>Return the data type of the array elements</td>
</tr>
<tr>
<td><code>a.dtype.name</code></td>
<td>Return the name of the data type of the array elements</td>
</tr>
</tbody>
</table>

**Array Mathematics**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>g = a - b</code></td>
<td>Subtract arrays element-wise</td>
</tr>
<tr>
<td><code>g = a + b</code></td>
<td>Add arrays element-wise</td>
</tr>
<tr>
<td><code>g = a * b</code></td>
<td>Multiply arrays element-wise</td>
</tr>
<tr>
<td><code>g = a / b</code></td>
<td>Divide arrays element-wise</td>
</tr>
</tbody>
</table>

**Comparing Arrays**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a == b</code></td>
<td>Check if two arrays are equal</td>
</tr>
<tr>
<td><code>a &lt; b</code></td>
<td>Check if elements of <code>a</code> are less than <code>b</code></td>
</tr>
<tr>
<td><code>a &gt; b</code></td>
<td>Check if elements of <code>a</code> are greater than <code>b</code></td>
</tr>
</tbody>
</table>

**Aggregate Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.sum()</code></td>
<td>Compute the sum of all elements</td>
</tr>
<tr>
<td><code>a.min()</code></td>
<td>Compute the minimum value of all elements</td>
</tr>
<tr>
<td><code>a.max()</code></td>
<td>Compute the maximum value of all elements</td>
</tr>
<tr>
<td><code>a.mean()</code></td>
<td>Compute the mean of all elements</td>
</tr>
<tr>
<td><code>a.median()</code></td>
<td>Compute the median of all elements</td>
</tr>
<tr>
<td><code>a.std()</code></td>
<td>Compute the standard deviation of all elements</td>
</tr>
</tbody>
</table>

**Copying Arrays**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.copy()</code></td>
<td>Create a copy of the array</td>
</tr>
</tbody>
</table>

**Slicing**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[0:2]</code></td>
<td>Select elements from index 0 to 1</td>
</tr>
<tr>
<td><code>a[0,1]</code></td>
<td>Select the element at indices (0, 1)</td>
</tr>
</tbody>
</table>

**Subsetting**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[2]</code></td>
<td>Select the element at the 2nd index</td>
</tr>
<tr>
<td><code>a[0:2]</code></td>
<td>Select elements from index 0 to 1</td>
</tr>
</tbody>
</table>

**Boolean Indexing**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[a &gt; 0]</code></td>
<td>Select elements greater than 0</td>
</tr>
<tr>
<td><code>a[a == 0]</code></td>
<td>Select elements equal to 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>DataCamp</strong> Learn Python for Data Science Interactively at <a href="http://www.DataCamp.com">www.DataCamp.com</a></th>
<th>NumPy Basics</th>
<th>NumPy Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D array</td>
<td>2D array</td>
<td>3D array</td>
</tr>
<tr>
<td><img src="image1" alt="1D array" /></td>
<td><img src="image2" alt="2D array" /></td>
<td><img src="image3" alt="3D array" /></td>
</tr>
<tr>
<td>axis 0</td>
<td>axis 1</td>
<td>axis 2</td>
</tr>
<tr>
<td><img src="image4" alt="Axis" /></td>
<td><img src="image5" alt="Axis" /></td>
<td><img src="image6" alt="Axis" /></td>
</tr>
</tbody>
</table>

**Inspecting Your Array**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.shape</code></td>
<td>Return the shape of the array</td>
</tr>
<tr>
<td><code>len(a)</code></td>
<td>Return the length of the array</td>
</tr>
<tr>
<td><code>np.size(a)</code></td>
<td>Return the total number of elements in the array</td>
</tr>
<tr>
<td><code>np.dtype(a)</code></td>
<td>Return the data type of the array</td>
</tr>
<tr>
<td><code>a.ndim</code></td>
<td>Return the number of dimensions of the array</td>
</tr>
<tr>
<td><code>a.dtype</code></td>
<td>Return the data type of the array elements</td>
</tr>
<tr>
<td><code>a.dtype.name</code></td>
<td>Return the name of the data type of the array elements</td>
</tr>
</tbody>
</table>

**Array Mathematics**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>g = a - b</code></td>
<td>Subtract arrays element-wise</td>
</tr>
<tr>
<td><code>g = a + b</code></td>
<td>Add arrays element-wise</td>
</tr>
<tr>
<td><code>g = a * b</code></td>
<td>Multiply arrays element-wise</td>
</tr>
<tr>
<td><code>g = a / b</code></td>
<td>Divide arrays element-wise</td>
</tr>
</tbody>
</table>

**Comparing Arrays**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a == b</code></td>
<td>Check if two arrays are equal</td>
</tr>
<tr>
<td><code>a &lt; b</code></td>
<td>Check if elements of <code>a</code> are less than <code>b</code></td>
</tr>
<tr>
<td><code>a &gt; b</code></td>
<td>Check if elements of <code>a</code> are greater than <code>b</code></td>
</tr>
</tbody>
</table>

**Aggregate Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.sum()</code></td>
<td>Compute the sum of all elements</td>
</tr>
<tr>
<td><code>a.min()</code></td>
<td>Compute the minimum value of all elements</td>
</tr>
<tr>
<td><code>a.max()</code></td>
<td>Compute the maximum value of all elements</td>
</tr>
<tr>
<td><code>a.mean()</code></td>
<td>Compute the mean of all elements</td>
</tr>
<tr>
<td><code>a.median()</code></td>
<td>Compute the median of all elements</td>
</tr>
<tr>
<td><code>a.std()</code></td>
<td>Compute the standard deviation of all elements</td>
</tr>
</tbody>
</table>

**Copying Arrays**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.copy()</code></td>
<td>Create a copy of the array</td>
</tr>
</tbody>
</table>

**Slicing**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[0:2]</code></td>
<td>Select elements from index 0 to 1</td>
</tr>
<tr>
<td><code>a[0,1]</code></td>
<td>Select the element at indices (0, 1)</td>
</tr>
</tbody>
</table>

**Subsetting**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[2]</code></td>
<td>Select the element at the 2nd index</td>
</tr>
<tr>
<td><code>a[0:2]</code></td>
<td>Select elements from index 0 to 1</td>
</tr>
</tbody>
</table>

**Boolean Indexing**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a[a &gt; 0]</code></td>
<td>Select elements greater than 0</td>
</tr>
<tr>
<td><code>a[a == 0]</code></td>
<td>Select elements equal to 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>DataCamp</strong> Learn Python for Data Science Interactively at <a href="http://www.DataCamp.com">www.DataCamp.com</a></th>
<th>NumPy Basics</th>
<th>NumPy Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D array</td>
<td>2D array</td>
<td>3D array</td>
</tr>
<tr>
<td><img src="image1" alt="1D array" /></td>
<td><img src="image2" alt="2D array" /></td>
<td><img src="image3" alt="3D array" /></td>
</tr>
<tr>
<td>axis 0</td>
<td>axis 1</td>
<td>axis 2</td>
</tr>
<tr>
<td><img src="image4" alt="Axis" /></td>
<td><img src="image5" alt="Axis" /></td>
<td><img src="image6" alt="Axis" /></td>
</tr>
</tbody>
</table>
### Scikit-Learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

#### A Basic Example

```python
>>> from sklearn import neighbors, datasets, preprocessing
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X, y)
>>> y_pred = knn.predict(X)
>>> accuracy_score(y, y_pred)
```

#### Loading The Data

```python
>>> X = np.random.random((10,5))
>>> y = np.array(['M', 'M', 'F', 'M', 'M', 'F', 'M', 'M', 'F', 'M'])
>>> X[X < 0.7] = 0
```

#### Preprocessing The Data

```python
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
```

#### Standardization

```python
>>> from sklearn.preprocessing import StandardScaler
>>> X_train = datasets.load_iris()
>>> X = X_train.data[:, :2]
>>> y = X_train.target
>>> scaler = StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
```

#### Normalization

```python
>>> from sklearn.preprocessing import Normalizer
>>> X_train = datasets.load_iris()
>>> X = X_train.data[:, :2]
>>> y = X_train.target
>>> normalizer = Normalizer().fit(X_train)
>>> X_train = normalizer.transform(X_train)
```

#### Binarization

```python
>>> from sklearn.preprocessing import Binarizer
>>> X_train = datasets.load_iris()
>>> X = X_train.data[:, :2]
>>> y = X_train.target
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binarized = binarizer.transform(X)
```

#### Generating Polynomial Features

```python
>>> from sklearn.preprocessing import PolynomialFeatures
>>> X_train = datasets.load_iris()
>>> X = X_train.data[:, :2]
>>> y = X_train.target
>>> polynomial_features = PolynomialFeatures(2)
>>> transformed_X = polynomial_features.fit_transform(X)
```

#### Encoding Categorical Features

```python
>>> from sklearn.preprocessing import LabelEncoder
>>> y = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
>>> encoder = LabelEncoder()
>>> y = encoder.fit_transform(y)
```

#### Imputing Missing Values

```python
>>> from sklearn.preprocessing import Imputer
>>> data = [1, 2, np.nan, 4, 5, 6]
>>> imputer = Imputer(strategy='mean', axis=0)
>>> X = imputer.fit_transform(data)
```

#### Cross-Validation

```python
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
```

#### Grid Search

```python
>>> from sklearn.model_selection import GridSearchCV
>>> grid = GridSearchCV(estimator=knn, param_grid=params)
>>> grid.fit(X_train, y_train)
```

#### Randomized Parameter Optimization

```python
>>> from sklearn.model_selection import RandomizedSearchCV
>>> randomized_search = RandomizedSearchCV(estimator=knn, param_distributions=params, n_iter=8)
>>> randomized_search.fit(X_train, y_train)
```

#### Evaluation Your Model's Performance

#### Classification Metrics

```python
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y, y_pred)

>>> from sklearn.metrics import classification_report
>>> classification_report(y_true, y_pred)
```

#### Mean Absolute Error

```python
>>> from sklearn.metrics import mean_absolute_error
>>> mean_absolute_error(y_true, y_pred)
```

#### Mean Squared Error

```python
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_true, y_pred)
```

#### R² Score

```python
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

#### Homogeneity

```python
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

#### V-measure

```python
>>> from sklearn.metrics import v_measure_score
>>> v_measure_score(y_true, y_pred)
```

#### Adjusted Rand Index

```python
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

#### Adjusted Rand Error

```python
>>> from sklearn.metrics import adjusted_rand_error
>>> adjusted_rand_error(y_true, y_pred)
```

#### Cross-Validation

```python
>>> from sklearn.cross_validation import cross_val_score
>>> cross_val_score(knn, X_train, y_train, cv=4)
```

#### Search Grid

```python
>>> grid = GridSearchCV(estimator=knn, param_grid=params)
>>> grid.fit(X_train, y_train)
```

#### Evaluate Your Model's Performance

#### Classification Metrics

```python
>>> from sklearn.metrics import classification_report
>>> classification_report(y_true, y_pred)
```

#### Accuracy Score

```python
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_true, y_pred)
```

#### Clustering Metrics

```python
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

#### Adjusted Rand Index

```python
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

#### Adjusted Rand Error

```python
>>> from sklearn.metrics import adjusted_rand_error
>>> adjusted_rand_error(y_true, y_pred)
```

#### V-measure

```python
>>> from sklearn.metrics import v_measure_score
>>> v_measure_score(y_true, y_pred)
```

#### Adjusted Rand Error

```python
>>> from sklearn.metrics import adjusted_rand_error
>>> adjusted_rand_error(y_true, y_pred)
```

#### Cross-Validation

```python
>>> from sklearn.cross_validation import cross_val_score
>>> cross_val_score(knn, X_train, y_train, cv=4)
```

#### Search Grid

```python
>>> grid = GridSearchCV(estimator=knn, param_grid=params)
>>> grid.fit(X_train, y_train)
```

#### Evaluate Your Model's Performance

#### Classification Metrics

```python
>>> from sklearn.metrics import classification_report
>>> classification_report(y_true, y_pred)
```

#### Accuracy Score

```python
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_true, y_pred)
```

#### Clustering Metrics

```python
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

#### Adjusted Rand Index

```python
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

#### Adjusted Rand Error

```python
>>> from sklearn.metrics import adjusted_rand_error
>>> adjusted_rand_error(y_true, y_pred)
```

#### V-measure

```python
>>> from sklearn.metrics import v_measure_score
>>> v_measure_score(y_true, y_pred)
```
Matplotlib
Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

## 1. Prepare the Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

## 2. Create Plot

```
>>> plt.plot(x, y, marker='o', linestyle='--', color='darkgreen')
>>> plt.scatter(x, y, marker='^', color='lightblue')
>>> plt.legend()  # optional for automatic legend
```

## 3. Plotting Routines

```
>>> fig, ax = plt.subplots()  # creates figure and axis
>>> ax.plot(x, y)
>>> ax.set_title('An Example Axes')  # set title
>>> ax.set_xlabel('X-Axis')  # set x-axis label
>>> ax.set_ylabel('Y-Axis')  # set y-axis label
```

### Data Distributions

```
>>> plt.hist(y, bins=10)  # make a histogram
```

### Colorbar

```
>>> im = ax.imshow(data)  # create image
>>> fig.colorbar(im)  # add color bar
>>> ax.text(0, 0, 'Example Text

另一行

示例', fontsize=20)  # add text
```

### Vector Fields

```
>>> plt.streamplot(X, Y, U, V)  # make a vector field
>>> plt.quiver(y, z)  # make a quiver plot
```

### Axes

```
>>> plt.figure()  # create a figure (also works with plt.subplot)
```

### Subplot

```
>>> fig, axes = plt.subplots(2, 2)  # create a grid of subplots
>>> axes[0, 0].bar(x, y)  # plot bar chart
>>> axes[1, 1].scatter(x, y)  # plot scatter plot
```

### Ticks

```
>>> ax.tick_params(axis='x', which='major', direction='in', length=10)  # set major tick direction
```

## 4. Customize Plot

### Colors, Color Bars & Color Maps

```
>>> plt.imshow(data, cmap='seismic')  # create colormap
```

### Lines and Legends

```
>>> ax.plot(x, y, label='Example Line')  # add line plot to legend
>>> ax.legend()  # show legend
```

### Text & Annotations

```
>>> ax.text(0.9, 0.1, 'Example Text', fontsize=20)  # add text
```

## 5. Save Plot

```
>>> plt.savefig('foo.png')  # save plot
```

## 6. Show Plot

```
>>> plt.show()  # display plot
```

---

**Python For Data Science Cheat Sheet**

**Matplotlib**

Learn Python Interactively at www.DataCamp.com
Python For Data Science Cheat Sheet
Seaborn
Learn Data Science Interactively at www.DataCamp.com

Statistical Data Visualization With Seaborn
The Python visualization library Seaborn is based on matplotlib and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

```python
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
```

The basic steps to creating plots with Seaborn are:
1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot

```python
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips")
>>> sns.set_style("whitegrid")
>>> g = sns.lmplot(x="total_bill", y="tip", data=tips)
```

Seaborn also offers built-in data sets:

```python
>>> titanic = sns.load_dataset("titanic")
>>> iris = sns.load_dataset("iris")
```

1. Data
   Also see Lists, NumPy & Pandas

```python
>>> import pandas as pd
>>> import numpy as np
>>> data = pd.DataFrame({"x":np.arange(1,101),
                      "y":np.random.normal(0,4,100)})
```

2. Figure Aesthetics
   Also see Matplotlib

```python
>>> f, ax = plt.subplots(figsize=(5,6))
>>> sns.set(style="whitegrid")
```

3. Plotting With Seaborn
   Also see Matplotlib

```python
>>> sns.set_style("whitegrid")
>>> sns.barplot(x="total_bill", data=titanic)
```

4. Further Customizations
   Also see Matplotlib

```python
>>> sns.set_theme(style="ticks")
>>> sns.color_palette("bright")
```

5. Show or Save Plot
   Also see Matplotlib

```python
>>> plt.savefig("foo.png")
>>> plt.clf()
```

DataCamp
Learn Python for Data Science Interactively
**Python For Data Science Cheat Sheet**

**Bokeh**


### Plotting With Bokeh

The Python interactive visualization library Bokeh enables high-performance visual presentation of large datasets in modern web browsers.

Bokeh’s mid-level general purpose bokeh.plotting interface is centered around two main components: data and glyphs.

The basic steps to creating plots with the bokeh.plotting interface are:

1. **Prepare some data:** Python lists, NumPy arrays, Pandas DataFrames and other sequences of values.
2. **Create a new plot:**
3. **Add renderers for your data, with visual customizations:** glyphs.
4. **Specify where to generate the output:**
5. **Show or save the results:**

#### Data

Under the hood, your data is converted to Column Data Sources. You can also do this manually:

```python
>>> from bokeh.models import ColumnDataSource
>>> df = ColumnDataSource(data=dict(x=[1,2,3,4,5], y=[3,4,5,6,7]))

>>> p = figure(title='my_plot')
>>> p.circle('x', 'y', size=10, legend='my data')
```

#### Plotting

```python
>>> from bokeh.plotting import figure
>>> p1 = figure(title='my_plot')
>>> p2 = figure(title='my_plot2')

>>> p1.circle('x', 'y', size=10, legend='my data')
>>> p2.circle('x', 'y', size=10, legend='my data2')
```

#### Rendering & Visual Customizations

**Scatter Markers**

```python
>>> p1.circle('x', 'y', size=10, legend='my data')
>>> p1.circle('x', 'y', size=10, legend='my data2')
```

**Line Glyphs**

```python
>>> p1.line('x', 'y', legend='my data')
>>> p1.line('x', 'y', legend='my data2')
```

**Hover Glyphs**

```python
>>> p = figure(title='my_plot')
>>> p.circle('x', 'y', legend='my data')
>>> p.circle('x', 'y', legend='my data2')
```

**Colormapping**

```python
>>> p1.circle('x', 'y', legend='my data', color='blue')
>>> p1.circle('x', 'y', legend='my data2', color='red')
```

### Customized Glyphs

**Selection and Non-Selection Glyphs**

```python
>>> p = figure()  
>>> p.circle('x', 'y', legend='my data')  
>>> p.circle('x', 'y', legend='my data2')
```

**Hover Glyphs**

```python
>>> p = figure()  
>>> p.circle('x', 'y', legend='my data')  
>>> p.circle('x', 'y', legend='my data2')
```

**Color mapping**

```python
>>> p1.circle('x', 'y', legend='my data', color='red')
```

### Grid Layout

```python
>>> from bokeh.layouts import gridplot
>>> grid = gridplot([[p1, p2], [p3]])
>>> show(grid)
```

### Tabbed Layout

```python
>>> from bokeh.models.widgets import Panel, Tabs
>>> tab1 = Panel(child=p1, title='tab1')
>>> tab2 = Panel(child=p2, title='tab2')
>>> layout = Tabs(tabs=[tab1, tab2])
```

### Linked Plots

**Linked Axes**

```python
>>> from bokeh.models import Legend
>>> legend = Legend(items=[('my data', [p1]), ('my data2', [p2])])
>>> layout = gridplot([[p1, p2], [p3]])
```

**Linked Brushing**

```python
>>> from bokeh.models import Legend
>>> legend = Legend(items=[('my data', [p1]), ('my data2', [p2])])
```

### Output & Export

**Notebook**

```python
>>> from bokeh.io import output_notebook, show
>>> output_notebook()
>>> show(p)
```

**HTML**

- **Standalone HTML**
  ```python
  >>> from bokeh.io import output_file, show
  >>> output_file('my_plot.html', title='my_plot')
  >>> show(p)
  ```

- **Components**
  ```python
  >>> from bokeh.embed import file_html
  >>> html = file_html(p, CDN, 'my_plot')
  >>> from bokeh.models import ColumnDataSource
  >>> df = ColumnDataSource(data=dict(x=[1,2,3,4,5], y=[3,4,5,6,7]))
  ```

**PNG**

```python
>>> from bokeh.io import export_png
>>> export_png(p, filename='my_plot.png')
```

**SVG**

```python
>>> from bokeh.io import export_svgs
>>> export_svgs(p, filename='my_plot.svg')
```

**JSON**

```python
>>> from bokeh.io import export_json
>>> export_json(p, filename='my_plot.json')
```

**CSV**

```python
>>> from bokeh.io import export_csv
>>> export_csv(p, filename='my_plot.csv')
```