```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Creating Arrays	Notes
`np.array([[1, 2, 3], [4, 5, 5]])`	From nested list; optionally specify `dtype='float'`
`np.full((3, 5), 2)`	3x5 array full of 2s
`np.arange(0, 10, 3)`	Equivalent to `np.array(range(0, 10, 3))`
`np.linspace(0, 2, 10)`	10 numbers evenly spaced from 0 through 2
`np.random.random((5, 2))`	5x2 array of random numbers in [0, 1). See also `np.random.randint`, `np.random.normal`

NumPy Expression	Notes
`a.ndim`	Number of dimensions of `a`
`a.shape`	Shape of `a` (a tuple)
`a.size`	Total number of elements in `a`
`a.dtype`	Data type of `a`'s elements
`a[2, 3]`	Element in row 2, column 3 of `a`. Each index can be a slice
`a[2:4, 5:8]`	Rows 2-3 and 5-7 of `a`. A view without `.copy()`
`a.reshape(3, 16)`	Elements of `a` rearranged into a 3x16 array. A view
`a.concatenate(b)`	`a` and `b` stacked. To concatenate side-by-side: `axis=1`
`a + np.sin(b)`	Applied element-wise. If `a` is 4x1 and `b` is 1x6, result is 4x6, broadcast to make dimensions match
`np.sum(a, axis=0)`	Sums along row dimension, giving one sum per column. See also `prod`, `mean`, `std`, `min`, `max`
`a[(a > 3) & (a < 10)]`	Elements of `a` greater than and less than 10. `a > 3` itself is an array of bools. Use `|` for or, ~ for not

```
df = pd.read_csv('path/to/your_file.csv')
```

Pandas Expression	Notes
`pd.DataFrame({'a':[1, 2, 3], 'b':[4, 5, 6]})`	A dataframe with the specified values in columns `'a'` and `'b'`.
`df.columns`	Column names of `df`
`df.head()`	First five rows of `df`
`df.describe()`	Basic stats on numeric columns of `df`
`df['area']`	The `'area'` column of `df`
`df.loc['Oregon']`	The row of `df` indexed by `'Oregon'`. Can specify column in addition.

Pandas Expression	Notes
`df.iloc[3]`	Row 3 of `df`
`df.dropna()`	A copy of `df` without any rows containing `NaN`
`pd.concat([a, b])`	Concatenation of DataFrames `a` and `b`
`pd.merge(a, b, how='inner')`	Joins `a` and `b`; how can be `'inner'`, `'outer'`, `'left'`, or `'right'`
`df.sort_values(by='name')`	A copy of `df`, sorted by the `'name'` column
`df.pivot_table('survived', index='sex', columns='class', aggfunc='mean')`	Pivot table showing the mean survival rate for each combination of sex and class

Matplotlib Expression or Statement	Notes
`plt.plot(x, y)`	Line plot of `x` and `y` (Series, DataFrame columns, or lists). Optional third argument like `'--c'` for dashed cyan line
`plt.scatter(x, y)`	Scatter plot of `x` and `y`
`plt.hist(x)`	Histogram of `x`
`plt.xlim(0, 10)`	Set x limits of plot
`plt.xlabel('Year')`	Set x label of plot
`plt.title('Duck Prices')`	Set title of plot
`plt.legend()`	Add legend
`plt.text(x, y, 'look here')`	Add annotation
`fig, ax = plt.subplots(2, 3)`	Creates a figure with a 2x3 grid of axes

Named Argument	Matplotlib Functions	Notes
`c`	`plot`, `scatter`	Color (`plot`) or sequence of colors (`scatter`)
`s`	`scatter`	Sequence of sizes (numbers)
`label`	`plot`, `scatter`	Label to use in legend
`marker`	`plot`, `scatter`	Marker to use for each point

Marker	Linestyle	Color
`'.'` point	`'-'` solid	`'blue'`
`'o'` circle	`':'` dotted	`'g'` green, from among `rgbcmyk`
`'v'`, `'^'`, `'<'` triangles	`'--'` dashed	`'0.75'` grayscale
`'s'` square	`'-.'` dashdot	`'#FFDD44'` hex code
`'+'` plus		`(1.0, 0.2, 0.3)` RGB tuple
`'x'` x		
`'D'` diamond		Color examples from VanderPlas, *Python Data Science Handbook*