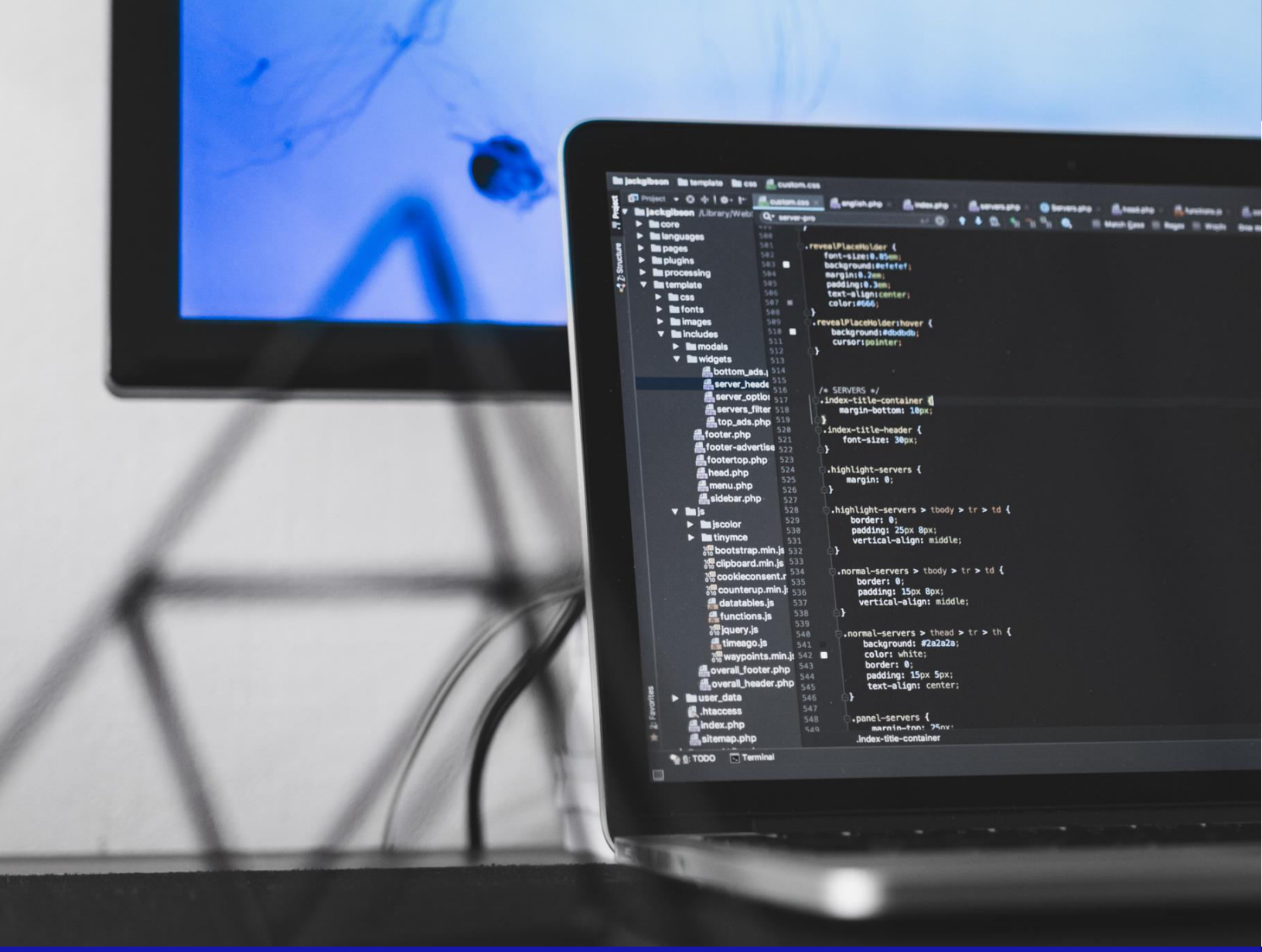

Python for Finance
Quandl Kurs-Handbuch

DATAMICS
MACHINE INTELLIGENCE CONSULTING SERVICES

NumPy/SciPy

arr = array([ ])	Numpy Array erstellen	arr = array([1,3,7,3,6,8,2,9])
arr.shape	Form eines Arrays	(8,) (acht Einträge in einer Dimension, null in allen anderen)
convolve(a,b)	lineare Konvolution zweier Sequenzen	np.convolve([1,2,3],[0,1,0.5]) --> array([0. , 1. , 2.5, 4. , 1.5])
arr.reshape()	Array neu formatieren	arr.reshape(4,2) --> array([1,3],[7,3],[6,8],[2,9])
sum(arr)	alle Elemente des Arrays aufsummieren	sum(arr) --> (1+3+7+3+6+8+2+9 =)39
mean(arr)	Durchschnitt des Arrays berechnen	mean(arr) --> (39/8 =) 4.875
std(arr)	Standardabweichung des Arrays berechnen	std(arr) --> (sqrt(mean(abs(x-x.mean()**2) =) 2.8035
dot(arr1,arr2)	Skalarprodukt zweier Arrays berechnen	dot(arr, [2,5,6,8,2,7,5,1]) = 170
vectorize()	skalare Funktion in eine Funktion umwandeln, die mit Vektoren und Arrays umgehen kann	

Pandas

Strukturen erstellen

s = Series (data, index)	Reihe (Series) erstellen	s = Series([2,6,1,8], index = [100,101,102,103] --> [100,2],[101,6],[102,1],[103,8]
df = DataFrame (data, index, columns)	DataFrame erstellen	d = {'col1':[1,2],'col2':[3,4] \\ df = DataFrame(data=d)
p = Panel (data, items, major_axis, minor_axis)	Panel erstellen	p = Panel(data=d)

DataFrame Befehle

df[col]	Spalte auswählen	df[1] --> [1,2]
df.iloc[label]	Reihe anhand eines Labels auswählen	
df.index	Gibt den Index eines DataFrames zurück	
df.drop()	Zeile oder Spalte löschen; für Spalte axis=1 übergeben	df.drop('col1') entfernt alle Werte aus col1 (1 und 2)
df1 = df1.reindex_like(df1,df2)	df1 neu indizieren mit dem Index von df2	
df.reset_index()	Index zurücksetzen; alten Index in Spalte "index" schreiben	
df.reindex()	DataFrame Index ändern, neue Indices als "NaN"	new_index = ['a','b'] >> df.reindex(new_index)
df.head(n)	erste n Zeilen anzeigen	
df.tail(n)	letzte n Zeilen anzeigen	
df.sort()	Indices sortieren	result = df.sort(['A', 'B'], ascending=[1, 0])
df.sort(axis=1)	Spalten sortieren	result = df.sort(['A', 'B'], axis=1)
df.pivot(index,column,values)	Gibt umgeformtes DataFrame nach geordneten Index- / Spaltenwerten zurück	df.pivot(index='foo', columns='bar', values='baz')
df.T	DataFrame transponieren	df1_transposed = df1.T # or df1.transpose()
df.stack()	niedrigstes Spaltenlabel zu innerstem Zeilenindex ändern	
df.unstack()	innersten Zeilenindex zu niedrigstem Spaltenlabel ändern	
df.applymap()	Funktion auf jedes Element des DataFrames anwenden	df.applymap(lambda x: x**2)
df.apply()	Funktion entlang einer gegebenen Achse anwenden	df.apply(np.sum, axis=0)
df.dropna()	Entfernt fehlende Elemente	df.dropna(axis='columns')
df.count()	Gibt Serie von Zeilenanzahlen für jede Spalte zurück	
df.min()	Gibt Minimum jeder Spalte zurück	
df.max()	Gibt Maximum jeder Spalte zurück	
df.describe()	Erstellt mehrere beschreibende Statistiken für jede Spalte	
concat()	Verbindet DataFrame- oder Series-Objekte	pd.concat([s1, s2], ignore_index=True)

Groupby

groupby()	Aufteilung nach Spalten und Erstellung eines Groupby-Objekts	**df.groupby(['Animal']).mean()**
gb.agg()	Anwendung einer Funktion auf ein Groupby-Objekt	**df.groupby('A').agg('min')**
gb.transform()	Anwendung einer Funktion und Rückgabe eines Objekts mit selbem Index wie das, auf welches Funktion angewendet wurde	**df.transform(lambda x: x + 1)**
gb.filter()	Groupby-Objekt anhand bestimmter Funktion filtern	**grouped.filter(lambda x: x['B'].mean() > 3.)**
gb.groups	Gibt ein dict zurück mit den einzelnen Gruppen als keys und den zugehörigen Achsenbeschriftungen als values	

I/O

df.to_csv('foo.csv')	als CSV speichern
read_csv('foo.csv')	CSV auslesen und in DataFrame umwandeln
to_excel('foo.xlsx', sheet_name)	als Excel-Datei speichern
read_excel('foo.xlsx','sheet1', index_col = None, na_values = ['NA'])	Excel-Datei auslesen und in DataFrame umwandeln

Pandas Zeitreihen

Beliebige Strukur mit DateTimeIndex

date_range(start, end, freq)	Erstellt einen Zeitreihenindex	**pd.date_range(start='1/1/2018', periods=5, freq='M')**

Freq hat viele Optionen, darunter:

B	Arbeitstag
D	Kalendertag
W	wöchentlich
M	monatlich
Q	pro Quartal
A	jährlich
H	stündlich

ts.resample()	Neue Stichprobe mit neuer Häufigkeit	**series.resample('3T').sum()**
ts[]	Gibt Daten für bestimmten Zeitpunkt zurück	
ts.between_time()	Gibt Daten zwischen bestimmten Intervallen zurück	**ts['2013-01-02':'2013-01-03'].between_time('20:00', '22:00')**
to_pydatetime()	Wandelt Pandas DatetimeIndex in in datetime.datetime Objekt um	
to_datetime()	Wandelt Liste von Zeitobjekten (Strings, "epochs", etc.) in ein DatetimeIndex um	**pd.to_datetime('13000101', format='%Y%m%d', errors='ignore')**

Grafische Darstellung (Plotting)

Matplotlib ist ein extrem leistungsfähiges Modul. Schau dir www.matplotlib.org für die genaue Dokumentation an.

plot()	Daten oder eine Funktion in einem Intervall plotten
xlabel()	x-Achse beschriften
ylabel()	y-Achse beschriften
title()	Dem Graphen eine Überschrift geben

`subplot(n,x,y)`	Erstellt mehrere Plots: n-Anzahl der Plots, x-Zahl auf der horizontalen Achse, y-Zahl auf der vertikalen Achse
`xticks([ ],[ ])`	Schrittweite für x-Achse festlegen, erster Array für Werte, zweiter für Beschriftungen
`yticks([ ],[ ])`	Schrittweite für y-Achse festlegen, erster Array für Werte, zweiter für Beschriftungen
`ax=gca()`	Aktuelle Achse auswählen
`ax.spines[].set_color()`	Achsenfarbe ändern, "none" um sie zu entfernen
`ax.spines[].set_position()`	Achsenposition ändern, kann Koordinatenraum ändern
`legend(loc=' ')`	Legende erstellen, "best" für automatische Beschriftung
`savefig('foo.png')`	Plot speichern

Quandl

Das Quandl Paket erlaubt Quandl API-Zugang aus Python heraus, was es schnell und einfach macht numerische Daten zu erfassen und zu verändern.

In deinem ersten Quandl Funktionsaufruf solltest du dein authtoken (welches du nach der Anmeldung auf der Website von Quandl findest) angeben, um Einschränkungen bei bestimmten API Aufrufe zu verhindern.

Schau dir www.quandl.com/help/packages/python an für mehr Informationen.

Quandl ist eine Suchmaschine für numerische Daten, die einfachen Zugang zu finanziellen, sozialen und demographischen Daten aus hunderten Quellen ermöglicht.

`authtoken = 'YOURTOKENHERE'`	Füge das Folgende zu jedem Funktionsaufruf hinzu
`get('QUANDL/CODE')`	Lade Quandl Daten für bestimmten Quandl Code als DataFrame runter

`search('searchterm')`	Quandl durchsuchen und die ersten vier Ergebnisse zurückgeben
`push(data, code, name)`	Ein Pandas DataFrame (mit Zeitreihenindex) auf Quandl hochladen. Der Code muss komplett aus Großbuchstaben und Zahlen bestehen
`authtoken = 'YOURTOKENHERE'`	Füge das Folgende zu jedem Funktionsaufruf hinzu
`get('QUANDL/CODE')`	Lade Quandl Daten für bestimmten Quandl Code als DataFrame runter
`search('searchterm')`	Quandl durchsuchen und die ersten vier Ergebnisse zurückgeben
`push(data, code, name)`	Ein Pandas DataFrame (mit Zeitreihenindex) auf Quandl hochladen. Der Code muss komplett aus Großbuchstaben und Zahlen bestehen

Grafische Darstellung Beispiel

```
import Quandl as q
import matplotlib.pyplot as plt
rural =  q.get('WORLDBANK/USA_SP_RUR_TOTL_ZS')
urban = q.get('WORLDBANK/USA_SP_URB_TOTL_IN_ZS')

plt.subplot(2, 1, 1)
plt.plot(rural.index,rural)

plt.xticks(rural.index[0::3],[])

plt.title('American Population')

plt.ylabel('% Rural')
plt.subplot(2, 1, 2)
plt.plot(urban.index,urban)

plt.xlabel('year')
plt.ylabel('% Urban')
plt.show()
```