Lernmodul zum Datenimport und Datenvorbereitung mit pandas

Import und Export

Herstellen einer Datenbankverbindung mit sqlalchemy	from sqlalchemy import create_engine engine = create_engine('sqlite:///*pfad-zur-datenbank*') db_connection = engine.connect()

Import von Daten aus einer Datenbank	# Voraussetzung: Verbindung zur Datenbank hergestellt
	df = pd.read_sql_query(*sql_query*, db_connection)

Import von Daten aus einer csv Datei	df = pd.read_csv(*pfad-zur-csv-datei*)
Mergen von Daten	merged_df = df1.merge(df2, on='spaltenname-zum-mergen')

Export von Daten in eine Datenbank	# Voraussetzung: Verbindung zur Datenbank hergestellt
	df.to_sql('name-für-db-tabelle', db_connnection, index = False, if_exists="replace")

Export von Daten in eine csv Datei	df.to_csv('pfad-für-csv-datei', index=False)

Datenanalyse

DataFrame anzeigen	df.head() # die ersten 5 Zeilen df.tail() # die letzten 5 Zeilen
Erstellen einer Kopie des DataFrames	df_copy = df.copy()
Beschreiben der Daten	df.shape # (Anzahl der Zeilen, Anzahl der Spalten) df.describe() # Anzeigen der Lageparameter df['spaltenname'].unique() # Anzeigen der einzigartigen Werte df['spaltenname'].value_counts() # Anzeigen der Vorkommen der Werte innerhalb einer Spalte
Zugriff auf Datenwerte	df['spaltenname'] # Zugriff auf eine Spalte df['spaltenname'][index] # Zugriff auf einen Wert einer Spalte df.iloc[zeilenindex,spaltenindex] # Index-basierte Auswahl von Daten df.loc[zeilenindex, 'spaltenname'] # label-basierte Auswahl von Daten
Bedingte Auswahl von Daten	df.loc[df['spaltenname'] == 'auszuwählender-wert'] # Mehrere Bedingungen mit & oder | verknüpfen
Umbenennen von Spaltennamen	df.rename(columns ={'alter-name':'neuer-name'}, inplace = True) df = df.rename(columns ={'alter-name':'neuer-name'})
Datentypen der Spalten	df['spaltenname'].dtype # Anzeigen des Datentyps einer Spalte df.dtypes # Anzeigen der Datentypen jeder Spalte df.astype({'spaltenname':datentyp}) # Konvertieren des Datentyps einer Spalte df['spaltenname'] = pd.to_datetime(df['spaltenname']) # Konvertieren in Datumswerte
Sortieren der Daten	df.sort_values('spaltenname') # Sortieren nach einer Spalte df.sort_index() # Sortieren nach den Indexwerten

Indexing Operatoren

	Python Indexing Operator	pandas iloc Operator	pandas loc Operator
Anwendung	df[Spaltenname][Zeilenindex] df[[Spaltennamen]]	df.iloc[Zeilenindex, Spaltenindex] : für alle Zeilen/Spalten	df.loc[Zeilenindex, Spaltenname] : für alle Zeilen/Spalten
Spalten-Zeilen-Reihenfolge	1. Spalten 2. Zeilen	1. Zeilen 2. Spalten	1. Zeilen 2. Spalten
Umgang mit dem letzten Indexwert	Nur einzelne Indizes	Nicht inklusive	Inklusive
Anwendungs-beispiele	# Alle Zeilen der Spalte 'title' df['title']	# Alle Zeilen der Spalte 'title' (Index 2) df.loc[:, 2] # Ausgabe der Zeilen 2 bis 4 df.loc[2:5] # Ausgabe der ersten 3 Zeilen und Spalten df.loc[:3, :3]	# Alle Zeilen der Spalte 'title' df.loc[:, 'title'] # Ausgabe der Zeilen 2 bis 4 df.loc[2:4] # Ausgabe der ersten 3 Zeilen und Spalten df.loc[:3, [Spaltennamen]]

Datenbereinigung

Anzeigen mit fehlenden Werten	df.isnull().sum() # Anzahl der Null Werte je Spalte df[pd.isnull(df.spaltennamen)] # Zeilen mit fehlenden Werten in einer Spalte
Ersetzen von fehlenden Werten	df.spaltenname.fillna('neuer-wert', inplace=True)
Entfernen von Zeilen mit fehlenden Werten in einer bestimmten Spalte	df = df.dropna(subset=['spaltenname'])
Anzeigen von Duplikaten	df.duplicated().sum() # Anzahl der Duplikate df[df.duplicated(subset ='spaltenname')] # Duplikate innerhalb einer Spalte
Entfernen von Duplikaten	df.drop_duplicates(inplace=True)

Datenmanipulation

Hinzufügen von Zeilen	df = df.append(anzuhängender-dataframe, ignore_index=True)
Löschen von Zeilen	df.drop(zu-löschender-index-wert, axis=0, inplace=True) df.drop(['spaltenname1', 'spaltenname2'] , axis=0, inplace=True)
Erzeugen einer Spalte	df['neuer-spaltenname'] = # Neue Werte z. B. mittels map()
Löschen einer Spalte	df.drop('zu-löschender-spaltenname', axis=1, inplace=True) df.drop(['spaltenname1', 'spaltenname2'], axis=1, inplace=True)
Ersetzen von Datenwerten	df.spaltenname.replace('alter-wert', 'neuer-wert')
Mappen von Datenwerten	df['spaltenname'] = df['spaltenname'].map(lambda d: #transformiere den Datenwert hier) df['spaltenname'] =df['spaltenname'] .apply(funktionsname)
Gruppieren von Daten	df.groupby('spaltenname') # nach einer Spalte df.groupby(['spaltenname1', 'spaltenname2']) # nach mehreren Spalten
Binning	bins = [# Definition der Grenzen der einzelnen Bins] labels = [# Labels der einzelnen Bins] df['spaltenname'] = pd.cut(df['spaltenname'], bins, labels)
One-hot Encoding	df = pd.concat([df,pd.get_dummies(df['spaltenname'])],axis=1)