Python Cheat Sheet [Data Science Introductie]

Strings	
string = 'appel'	Geef de variabele 'string' de string 'appel' als waarde
string.capitalize()	Alleen de eerste letter als hoofdletter
string.strip()	Verwijdert alle spaties aan het begin en eind
string.split('_')	Deelt de string op in losse elementen, bij iedere '_'

Lists	
lijst = [element_1, element_2, element_n]	Syntax voor een list. Voorbeeld: fruit_soorten = ['appel', 'banaan', 'mango']
lijst[5]	Geeft de waarde van het element met index 5
lijst[5:7]	Geeft de waarden van de elementen met index 5 tot 7
lijst[5:]	Geeft alle waarden van de elementen vanaf index 5
len(lijst)	Geeft het aantal elementen in een list
del(lijst[1])	Verwijdert het element met index 1
sorted(lijst)	Sorteert de list
min(lijst)	Geeft de minimale waarde
max(lijst)	Geeft de maximale waarde
sum(lijst)	Geeft de som
lijst.append('appel')	Voegt 'appel' toe aan de list
lijst.remove('appel')	Verwijdert 'appel' uit de list
lijst.count('appel')	Telt hoe vaak 'appel' voorkomt in de list
lijst.clear()	Verwijdert alle elementen

Dictionaries	
dictionary = {key_1: value_1, key_2: value_2, key_n: value_n}	Syntax voor een dictionary. Voorbeeld: gewichten = {'appel': 150, 'banaan': 120, 'mango': 90}
dictionary['key']	Geeft de waarde van de key met naam 'key'
dictionary['key'] = 0	Past de waarde van de key met naam 'key' aan naar 0
del(dictionary['key'])	Verwijdert het element met naam 'key'
dictionary.keys()	Toont alle keys in een list
dictionary.values()	Toont alle waarden in een list
dictionary.update(dictionary1)	Voegt elementen uit 'dictionary1' toe aan 'dictionary'
dictionary.clear()	Verwijdert alle elementen

Logica	
==	Gelijk aan (relationele operator)
<	Kleiner dan (relationele operator)
>	Groter dan (relationele operator)
>=	Groter dan of gelijk aan (relationele operator)
<=	Kleiner dan of gelijk aan (relationele operator)
!=	Ongelijk aan (relationele operator)
and	En (logische operator)
or	Of (logische operator)
not	Niet (logische operator)

if getal == 0:
print('Het getal is 0.')
elif getal > 0:
print('Het getal is positief.')
else:
print('Het getal is negatief.')
	Een if statement, in dit voorbeeld om te bekijken of de waarde in variabele 'getal' positief of negatief is

for getal in [0, 1, 2, 3] print(getal)	Een for loop, om over de elementen in een list te itereren
for getal in range(0, 4) print(getal)	Een for loop, om over een reeks getallen te itereren

Functies	
print('Hallo!')	Print de tekst 'Hallo!'
type(x)	Geeft het datatype van 'x'
abs(x)	Geeft de absolute waarde van 'x'
round(x,n)	Rond 'x' af op 'n' decimalen
def functienaam(getal_1, getal_2): som = getal_1 + getal_2 return som	Maak een functie aan met 2 argumenten 'getal_1' en 'getal_2', waarbij je in de functie de som berekent en teruggeeft als output

Numpy	
import numpy as np	Importeer package Numpy en noem het 'np'
array = np.array(lijst)	Maak een Numpy array van een lijst
array.shape	Het aantal rijen en kolommen van het Numpy array
array[0,1]	Selecteert de waarde op de eerste rij en de tweede kolom
array[1:,1:3]	Selecteert alle waarden vanaf de eerste rij, en van de tweede en derde kolom
array.astype(float)	Stel het datatype in als float
np.mean(array[:,1])	Bereken het gemiddelde van de tweede kolom
np.median(array[:,1])	Bereken de mediaan van de tweede kolom
np.corrcoef(array[:,1], array[:,2])	Bereken de correlatie tussen de tweede en derde kolom
np.std(array[:,1])	Bereken de standaarddeviatie van de tweede kolom
np.sort(array[:,1])	Sorteer de waarden uit de tweede kolom

Python Cheat Sheet [Data Science Introductie]

Pandas: data importeren	
import pandas as pd	Importeer package Pandas en noem het 'pd'
df = pd.read_csv(bestandsnaam)	Uit een CSV bestand
df = pd.read_excel(bestandsnaam)	Uit een Excel bestand
df = pd.read_sql(query, connectie_object)	Uit een SQL database / tabel
df = pd.read_json(json_string)	Data in JSON format
df = pd.DataFrame(dictionary)	Data in een dictionary

Pandas: data verkennen	
df.head(n)	De eerste 'n' rijen
df.tail(n)	De laatste 'n' rijen
df.shape	Het aantal rijen en kolommen van het dataframe
df.dtypes	Informatie over de datatypes van kolommen
df.info()	Informatie over de index, datatypes en geheugen
df.describe()	Statistische gegevens van numerieke kolommen
df['kolom'].unique()	De unieke waarden in kolom 'kolom'
df['kolom'].value_counts()	De unieke waarden en voorkomendheid hiervan voor kolom 'kolom'
df.count()	Het aantal niet null waarden van elke kolom
df.sum()	De som van alle waarden per kolom
df.max()	De hoogste waarde van elke kolom
df.min()	De laagste waarde van elke kolom
df.mean()	Het gemiddelde per kolom
df.median()	De mediaan per kolom
df.corr()	De correlatie tussen kolommen
df.std()	De standaarddeviatie per kolom

Pandas: data opschonen	
df.isnull()	True bij missende waarden
df.notnull()	False bij missende waarden
df.dropna()	Verwijder rijen met missende waarden
df.dropna(axis=1)	Verwijder kolommen met missende waarden
df.fillna('missend')	Vervang missende waarden door tekst 'missend'
df['kolom'].fillna(df['kolom'].mean())	Vervang missende waarden in kolom 'kolom' door het gemiddelde van de kolom
df['kolom'].astype(float)	Stel het datatype van kolom 'kolom' in als float (kommagetal)
df['kolom'].replace(2,'twee')	Vervang alle waarden in kolom 'kolom' gelijk aan 2 door 'twee'
df.columns = ['a','b','c']	Hernoem de eerste 3 kolommen naar 'a', 'b', en 'c'
df.rename(columns={'kolom': 'kolom1'})	Hernoem kolom 'kolom' naar 'kolom1'

Pandas: data selecteren, aanpassen en groeperen	
&	Logische 'and' operator in Pandas
|	Logische 'or' operator in Pandas
~	Logische 'not' operator in Pandas
df['kolom']	Selecteer de waarden van kolom 'kolom' als serie
df[['kolom1', 'kolom2']]	Selectie van de 2 genoemde kolommen
df[df['kolom'] == x]	DataFrame met alleen rijen waarbij waarden uit de kolom gelijk zijn aan 'x'
df[df['kolom'].str.contains('string', regex=False)]	DataFrame met alleen rijen waarbij waarden uit de kolom 'string' bevatten
df.loc[i]	Een selectie op de index
df.iloc[i]	Een selectie op de positie van de rij
df.iloc[0,0]	De waarde van het de eerste kolom van de rij op de eerste positie

for index, row in df.iterrows():
  if row['kolom'] == a:
  df.loc[index, 'kolom1'] = b
	Pas met .iterrows() en een for loop waarden in kolom 'kolom1' aan naar waarde 'b', wanneer waarden in kolom 'kolom' gelijk zijn aan 'a'

df.loc[df['kolom'] = a, 'kolom1'] = b	Pas met loc[] waarden in kolom 'kolom1' aan naar waarde 'b', wanneer waarden in kolom 'kolom' gelijk zijn aan 'a'
df['kolom1'] = df['kolom'].apply(lambda x: b if x == a else x)	Pas met .apply() waarden in kolom 'kolom1' aan naar waarde 'b', wanneer waarden in kolom 'kolom' gelijk zijn aan 'a'
df['kolom1'] = df['kolom'].mask(df['kolom'] == a, b)	Pas met .mask() waarden in kolom 'kolom1' aan naar waarde 'b', wanneer waarden in kolom 'kolom' gelijk zijn aan 'a'

Pandas: data selecteren, aanpassen en groeperen	
df[df['kolom'] > 0.5]	Selecteer alleen rijen waarbij waarden in 'kolom' groter zijn dan 0.5
df[(df['kolom'] > 0.5) & (df['kolom'] < 0.7)]	Selecteer rijen met waarden voor 'kolom' tussen 0.5 en 0.7
df.sort_values('kolom')	Sorteer rijen op basis van waarden van 'kolom', oplopend
df.sort_values('kolom', ascending=False)	Sorteer aflopend
df.sort_values(['kolom', 'kolom2'], ascending=[True,False])	Sorteer oplopend en aflopend, voor 2 kolommen
df.groupby('kolom').mean()	Groepeer op unieke waarden in 'kolom', en neem het gemiddelde
df.groupby('kolom')['kolom2'].mean()	Groepeer op 'kolom', en toon alleen waarden van 'kolom2'
df.groupby('kolom').agg({'kolom2': 'mean', 'kolom3': 'count'})	Groepeer, en pas verschillende berekeningen toe
df.pivot_table(index='kolom', values=['kolom2', 'kolom3'], aggfunc=mean)	Maak een pivottabel
df.apply(np.mean)	Pas de functie np.mean toe op alle rijen

Pandas: data samenvoegen	
df1.append(df2)	Met append() voeg je alle rijen van 'df1' toe aan het eind van 'df2' (gelijk aantal kolommen)
pd.concat([df1, df2])	Voeg alle rijen van 'df1' toe aan het eind van 'df2' (gelijk aantal kolommen)
pd.concat([df1, df2],axis=1)	Voeg alle kolommen van 'df1' toe aan het eind van 'df2' (gelijk aantal rijen)
df_left.join(df_right, on='kolom')	Voeg alle kolommen van 'df_right' toe aan 'df_left' op key 'kolom'
pd.merge(df_left, df_right, left_on='kolom', right_on='kolom1', right_index=True, how='left', sort=False)	Voeg met meerdere opties alle kolommen van 'df_right' toe aan 'df_left'

Pandas: data exporteren	
df.to_csv(bestandsnaam)	Opslaan als CSV bestand
df.to_excel(bestandsnaam)	Opslaan als Excel bestand
df.to_sql(tabel, connectie_object)	Opslaan in een SQL database / tabel
df.to_json(bestandsnaam)	Opslaan in JSON format

Pandas: data visualiseren	
df.plot.line(x='kolom1', y='kolom2')	Maak een lijngrafiek van waarden uit 'kolom2', tegen waarden uit 'kolom1'
df.plot(kind='line', x='kolom1', y='kolom2')	Maak een lijngrafiek van waarden uit 'kolom2', tegen waarden uit 'kolom1'
df.plot.bar(x='kolom1', y='kolom2')	Maak een staafgrafiek van waarden uit 'kolom2', tegen waarden uit 'kolom1'
df.plot.bar(x='kolom1', y=['kolom2', 'kolom3'])	Maak een staafgrafiek van waarden uit 'kolom2' en 'kolom3', tegen waarden uit 'kolom1'

Matplotlib: data visualiseren	
import matplotlib.pyplot as plt %matplotlib inline	Importeer module pyplot uit package Matpotlib en noem het 'plt' Wanneer je Jupyter Notebook gebruikt zie je hiermee het resultaat
x = [0, 1, 2, 3, 4, 5] y = [10, 11, 12, 13, 14, 15]	Kies waarden voor de x-as Kies waarden voor de y-as
fig = plt.figure(figsize=(15, 4)) ax = fig.add_subplot(1,1,1) ax.plot(x, y) plt.show()	Maak een figuur aan met een bepaalde grootte Maak op het figuur een assenstelsel aan (1 rij, 1 kolom, positie 1) Plot op het assenstelsel de x en y waarden Toon het resultaat
ax.bar(x, y)	Maak een staafgrafiek
ax.scatter(x, y)	Maak een scattergrafiek