*Übersichtsdatei aller **Skripte***

Jahresprojekt API Economy und Cloud Computing

Discoveroo API

Inhalt

1 data_scripts

1.1 getCitiesFromCountry

Skriptnamen	getCitiesFromCountry_geoDBCities.py
Information	Das Skript lädt alle in der Datenbank verfügbaren Städte aller Länder herunter, die mindestens eine Population von 100.000 Einwohnern aufweisen und kein Administrationsbereich (ADM2) sind. Es werden ebenfalls Daten über Koordinaten und Höhe über NN gesammelt.
Datenquelle	geoDBCities ist eine API, welche Daten über ca. 144.000 Städte der Welt beinhaltet. Dazu zählen Land, Population und Koordinaten.
Eingabeparameter (Input)	• Sprache • API-Key
Ausgabeparameter (Output)	• countryCode • type • cityName • cityId • regionName • regionCode • lat • lon • population • elevation • timezone
Technologie	Python
Abhängigkeiten	Die Datei hat keine vorausgehenden Abhängigkeiten.
Ausgabedateien	*data_raw/ allCitiesOver100k.csv*

1.2 getSightsForCity

Skriptname	getSightsForCity_OpenTripMap.py
Information	Ausgehend von Längen-, Lon und Einwohnerzahl werden interessante Plätze und Sehenswürdigkeiten gesammelt. Diese werden nach Klasse und Popularität getrennt, um eine bessere Berechnung von Indizes zu ermöglichen. (Countlevels)
Datenquelle	• Opentripmap • https://opentripmap.io/product • API

Eingabeparameter (Input)	• API-Key
	• lon
	• lat
	• Radius (abhängig von Population)
	  ○ > 1 Mio = 15km Radius
	  ○ 500.000 < x < 1 Mio = 10 km Radius
	  ○ <500.000 = 5km Radius

Ausgabeparameter (Output)	• cityId • searchRadius • population • history: hCountLevel 0 / 1 / 2 / 3 / 7 • cultural: cCountLevel 0 / 1 / 2 / 3 / 7 • religion: rCountLevel 0 / 1 / 2 / 3 / 7 • architecture: aCountLevel 0 / 1 / 2 / 3 / 7 • industrial: iCountLevel 0 / 1 / 2 / 3 / 7 • natural: nCountLevel 0 / 1 / 2 / 3 / 7
Technologie	Python
Abhängigkeiten	*allCitiesOver100k.csv*
Ausgabedateien	*data_raw/ sightsInCities.csv*

1.3 getWeatherForCity

Skriptname	getWeatherForCity_NOAA-GHCN-Daily.R
Information	Das Skript lädt Wetterdaten aus dem GHCN (Global Historical Climate Network).
Datenquelle	• NOAA-GHCN • https://www.ncdc.noaa.gov/
Eingabeparameter (Input)	• Lat der Städte • Lon der Städte
Ausgabeparameter (Output)	• Maximale und Minimale Temperatur (täglich, von 2016-2019) • Station ID • GPS-Daten der Station
Technologie	R
Abhängigkeiten	Die Datei hat keine vorausgehenden Abhängigkeiten.
Ausgabedateien	*data_raw/ weatherStationsByCity.csv* CSVs von jeder Wetterstation

1.4 getBeachesForCity

Skriptname	getBeachesForCity_openTripMap.py
Information	Das Skript sucht für alle Städte, die in der Datei allCitiesMinPopulation.csv vorhanden sind, in einem je nach Stadtgröße variierenden Suchradius nach Stränden. Diese werden nach Art und Popularität kategorisiert in einer weiteren .csv-Datei gespeichert.
Datenquelle	• Opentripmap • https://opentripmap.io/product • API
Eingabeparameter (Input)	• cityId • lat • lon • radius • population
Ausgabeparameter (Output)	• cityId • searchRadius • Anzahl der vorhandenen Strände nach Popularität und Strandart
Technologie	Python
Abhängigkeiten	Das Skript "*getBeachesForCity_openTripMap.py*" basiert auf dem *"getCitiesFromCountry_geoDBCities.py"*-Skript und damit auf der Ausgabedatei allCitiesOver100k.csv
Ausgabedateien	*data_raw/ beachesInCities.csv*

1.5 getGeologicalFormationsForCity

Skriptname	getGeologicalFormationsForCity_openTripMap.py
Information	Das Skript sucht für alle Städte, die in der Datei allCities.csv vorhanden sind, in einem je nach Stadtgröße variierenden Suchradius nach Berggipfeln und Felsformationen. Diese werden nach Art und Popularität kategorisiert in einer weiteren .csv-Datei gespeichert.
Datenquelle	• Opentripmap • https://opentripmap.io/product • API

Eingabeparameter (Input)	• cityId
	• lat
	• lon
	• radius
	  ○ > 1 Mio = 50km Radius
	  ○ 500.000 < x < 1 Mio = 30 km Radius
	  ○ <500.000 = 20 km Radius
	•
	• population

Ausgabeparameter (Output)	• cityId • searchRadius • Anzahl der vorhandenen **Berggipfel** und **Felsformationen** nach Popularität
Technologie	Python
Abhängigkeiten	Das Skript "*getGeologicalFormationsForCity_openTripMap.py*" basiert auf dem "getCitiesFromCountry_geoDBCities.py"-Skript und damit auf der Ausgabedatei allCitiesOver100k.csv.
Ausgabedateien	*data_raw/* geologicalFormationsInCities.csv

1.6 getCityPicture

Skriptname	getCityPicture.py
Information	Das Skript sucht für jede Stadt über bing.com ein Bild, speichert die Bild-URL und downloadet das Bild automatisch. Der Name des Bildes setzt sich aus CityId und „.jpg“ zusammen, um referenziert werden zu können.
Datenquelle	bing.com (Suchmaschine)
Eingabeparameter (Input)	• cityId • cityName
Ausgabeparameter (Output)	• Image: z.B. „659.jpg“
Technologie	Python
Abhängigkeiten	Das Skript "getCityPicture.py" basiert auf "cityData_final.csv", in der alle Städte gelistet sind, für die ein Bild benötigt wird.
Ausgabedateien	Image files (z.B. *659.jpg)*

1.7 getCostAndCrimeForCity

Skriptname	getCostAndCrimeForCity_Numbeo.py
Information	Das Skript sammelt Parameter für Lebenshaltungskosten und Kriminalität/Sicherheit von ca. 500 Städten.
Datenquelle	• Numbeo • https://www.numbeo.com • API Numbeo stellt Daten über Länder und vereinzelte Städte zur Verfügung, die entweder durch Umfragen oder eigene manuelle Datensammlung aggregiert werden. Methodik und Parametererläuterungen sind auf der Webseite einsehbar.
Eingabeparameter (Input)	• API Key • section 1 (für Lebenshaltungskosten) • section 7 (Für Kriminalität und Sicherheit)
Ausgabeparameter (Output)	Lebenshaltungskosten: • countryCode • cityId • cityName • cpi + rent Index • cpiIndex • rentIndex • groceriesIndex • purchasingPowerIndex • restaurantIndex Sicherheit und Kriminalität: • safetyIndex • crimeIndex
Technologie	Python
Abhängigkeiten	Die Datei hat keine vorausgehenden Abhängigkeiten.
Ausgabedateien	*data_raw/ costAndSafetyDataForCity.csv*

1.8 getCostAndCrimeForCountry

Skriptname	getCostAndCrimeForCountry_Numbeo.py
Information	Das Skript sammelt Parameter für Lebenshaltungskosten und Kriminalität/Sicherheit der verfügbaren Länder.
Datenquelle	• Numbeo • https://www.numbeo.com • API Numbeo stellt Daten über Länder und vereinzelte Städte zur Verfügung, die entweder durch Umfragen oder eigene manuelle Datensammlung aggregiert werden. Methodik und Parametererläuterungen sind auf der Webseite einsehbar.
Eingabeparameter (Input)	• API Key • section 1 (für Lebenshaltungskosten) • section 7 (Für Kriminalität und Sicherheit)
Ausgabeparameter (Output)	Lebenshaltungskosten: • countryCode • cityId • cityName • cpi + rent Index • cpiIndex • rentIndex • groceriesIndex • purchasingPowerIndex • restaurantIndex Sicherheit und Kriminalität: • safetyIndex • crimeIndex
Technologie	Python
Abhängigkeiten	Die Datei hat keine vorausgehenden Abhängigkeiten.
Ausgabedateien	*data_raw/ costAndSafetyDataForCountry.csv*

1.9 getLandData

Skriptname	getLandData_WorldBank.py
Information	Das Skript lädt Länderdaten wie die Fläche oder die Stadtfläche herunter. Dabei werden keine weiteren Berechnungen angestellt, sondern nur Rohdaten gesammelt.
Datenquelle	• WorldBank • data.worldbank.org • https://tcdata360.worldbank.org/ • API
Eingabeparameter (Input)	• ISO2 countryCode
Ausgabeparameter (Output)	• countryName • countryCode • population • countrySize • ruralSize • urbanSize • forestSize
Technologie	Python
Abhängigkeiten	Die Datei hat keine vorausgehenden Abhängigkeiten.
Ausgabedateien	*data_raw/ countryData.csv*

1.10 overallQualityOfInfrastructure

Dateiname	overallQualityOfInfrastructure.csv
Information	Die .csv-Datei wird ohne Skript direkt aus dem Internet heruntergeladen. Sie enthält eine Bewertung der insgesamten Infrastruktur der Länder.
Datenquelle	• WorldBank • data.worldbank.org • https://tcdata360.worldbank.org/
Eingabeparameter (Input)	• Keine
Ausgabeparameter (Output)	• ISO3 countryCode • countryName • indicatorID • IndicatorValue 1-7 (1 – niedrig, 7 – hoch)
Technologie	.csv
Abhängigkeiten	Die Datei hat keine vorausgehenden Abhängigkeiten.
Ausgabedateien	/

1.11 getLanguageForCountry

Skriptname	getLanguageForCountry.py
Information	Das Skript sammelt den Sprach-Code und den Sprachnamen eines Landes von einer API.
Datenquelle	• https://restcountries.eu/rest/v2/all • API Rest Countries stellt einen kostenlosen API-Service zur Verfügung, durch den Länderdaten gesammelt werden können.
Eingabeparameter (Input)	• None
Ausgabeparameter (Output)	Lebenshaltungskosten: • countryCode • languageCode • languageName
Technologie	Python
Abhängigkeiten	Das Skript sammelt die Daten nur für zuvor gefilterte Länder.
Ausgabedateien	*data_final/ language_data.csv*

1.12 getCurrencyForCountry

Skriptname	getCurrencyForCountry.py
Information	Das Skript sammelt Währungsdaten eines Landes. Darunter fallen Währungsname, -code und -symbol.
Datenquelle	• https://restcountries.eu/rest/v2/all • API Rest Countries stellt einen kostenlosen API-Service zur Verfügung, durch den Länderdaten gesammelt werden können.
Eingabeparameter (Input)	• None
Ausgabeparameter (Output)	• countryCode • currencyCode • currencyName • currencySymbol
Technologie	Python
Abhängigkeiten	Das Skript sammelt die Daten nur für zuvor gefilterte Länder.
Ausgabedateien	*data_final/ currency_data.csv*

1.13 getFlagsForCountry

Skriptname	getFlagsForCountry.py
Information	Das Skript sammelt Bilder der Flagge eines Landes in Form einer URL. Diese URL wird dem jeweiligen Land im Datensatz hinzugefügt.
Datenquelle	• https://restcountries.eu/rest/v2/all • API Rest Countries stellt einen kostenlosen API-Service zur Verfügung, durch den Länderdaten gesammelt werden können.
Eingabeparameter (Input)	• None
Ausgabeparameter (Output)	• countryCode • flagLink
Technologie	Python
Abhängigkeiten	Das Skript sammelt die Daten nur für zuvor gefilterte Länder.
Ausgabedateien	*data_processed/ flagLinks_clean.csv*

2 data_processing_scripts

2.1 cleanCityData

Skriptname	cleanCityData.py
Information	Dieses Skript …
Eingabeparameter (Input)	• countryCode • type • cityId • cityName • regionName • lat • lon population • elevation • timezone • regionCode
Ausgabeparameter (Output)	• countryCode • type • cityId • cityName • regionName • lat • lon population • elevation • timezone • regionCode
Technologie	Python
Abhängigkeiten	…/data_processed/cityData_regionCode.csv
Ausgabedateien	*../data_processed/cityData_clean.csv*

2.2 calculateUniqueRegionCode

Skriptname	calculateUnqiueRegionCode.py
Information	Das Skript verbindet Länder-Code und Regions-Code zu einer einzigartigen Regions-ID, um so die Region eindeutig identifizieren zu können.
Eingabeparameter (Input)	• countryCode • type • cityId • cityName • regionCode • regionName • lat • lon population • elevation • timezone
Ausgabeparameter (Output)	• countryCode • type • cityId • cityName • regionName • lat • lon population • elevation • timezone • Einzigartige Regions-ID: regionCode = countryCode_regionCode
Technologie	Python
Abhängigkeiten	…/data_raw/allCitiesOver100k.csv
Ausgabedateien	*../data_processed/cityData_regionCode.csv*

2.3 createRegionData

Skriptname	createRegionData.py
Information	Das Skript berechnet pro Stadtparameter drei entsprechende Regionsparameter: • den Durchschnitt aller in der Region liegenden Städte • das Maximum in der Region • das Minimum in der Region
Eingabeparameter (Input)	• regionCode • regionName • countryCode • culture_hIndex • culture_iIndex • culture_nIndex • culture_aIndex • culture_rIndex • culture_cIndex • formations_rIndex • formations_mIndex • beach_Index
Ausgabeparameter (Output)	• regionCode • regionName • countryCode • Durchschnitt pro Stadtparameter aller Städte einer Region • Minimum pro Stadtparamater aller Städte einer Region • Maximum pro Stadtparamater aller Städte einer Region
Technologie	Python
Abhängigkeiten	Die Regionsparamater basieren auf den Stadtparametern ../data_final/cityData_final.csv
Ausgabedateien	../data_final/regionData_final.csv

2.4 calculateCulturalIndex

Skriptname	calculateCulturalIndex.py
Information	Das Skript berechnet einen Kultur-Index auf Basis der Anzahl der kulturellen Sehenswürdigkeiten: Berechnete (veränderte) Parameter: • Kultur: Historie, Kultur, Religion, Architektur, Natur, Industrie Schritt 1: Multiplikation der ursprünglichen Werte, um eine Gewichtung darzustellen: ○ Countlevel0 * 0 (entfällt) ○ Countlevel1 * 2 ○ Countlevel2 * 3 ○ Countlevel3 * 4 ○ Countlevel7 * 6 Schritt 2: Aufsummierung der verschiedenen Countlevels eines Parameters
Eingabeparameter (Input)	• Radius: 5km/10km/15km • Countlevels 0,1,2,3,7 • Anzahl aller kulturellen Sehenswürdigkeiten
Ausgabeparameter (Output)	• cityId • searchRadius • culture_hIndex • culture_cIndex • culture_rIndex • culture_aIndex • culture_iIndex • culture_nIndex (Alle Indizes mit einem Faktor multipliziert und aufsummiert)
Technologie	Python
Abhängigkeiten	../data_raw/sightsInCities.csv
Ausgabedateien	*../data_processed/culturalIndices.csv*

2.5 calculateBeachIndex

Skriptname	calculateBeachIndex.py
Information	Das Skript berechnet einen Beach-Index auf Basis der Anzahl der Sehenswürdigkeiten: Berechnete (veränderte) Parameter: • Beaches: golden sand, white sand, black sand, shingle, rocks, urbans, nude, Other Schritt 1: Multiplikation der ursprünglichen Werte, um eine Gewichtung darzustellen: ○ Countlevel0 * 0 ○ Countlevel1 * 2 ○ Countlevel2 * 3 ○ Countlevel3 * 4 ○ Countlevel7 * 6 Schritt 2: Aufsummierung der verschiedenen Countlevels eines Parameters Schritt 3: Aufgrund verhältnismäßig geringer Anzahlen von verschiedenen Stränden werden alle Werte am Ende zu einem Index aufsummiert
Eingabeparameter (Input)	• radius: 5km/10km/15km • countlevels 0,1,2,3,7 • Anzahl aller kulturellen Strände
Ausgabeparameter (Output)	• cityId • searchRadius • beach_Index
Technologie	Python
Abhängigkeiten	../data_raw/ beachesInCities.csv
Ausgabedateien	*../data_processed/beachIndices.csv*

2.6 calculateGeologicalFormationsIndex

Skriptname	calculateGeologicalFormationsIndex.py

Information	Das Skript berechnet einen „Geologische Formationen“-Index auf Basis der Anzahl der Sehenswürdigkeiten:
	Berechnete (veränderte) Parameter:
	• Geological formations
	  • Mountain peaks
	  • Rock formations
	
	Schritt 1: Multiplikation der ursprünglichen Werte, um eine Gewichtung darzustellen:
	  ○ Countlevel0 * 0 (entfällt)
	  ○ Countlevel1 * 2
	  ○ Countlevel2 * 3
	  ○ Countlevel3 * 4
	  ○ Countlevel7 * 6
	Schritt 2: Aufsummierung der verschiedenen Countlevels eines Parameters
	Schritt 3: Aufgrund verhältnismäßig geringer Anzahlen von verschiedenen Stränden werden alle Werte am Ende zu einem Index aufsummiert

Eingabeparameter (Input)	• Radius: 20km/30km/50km • Countlevels 0,1,2,3,7 • Anzahl aller Bergspitzen und Steinformationen
Ausgabeparameter (Output)	• cityId • searchRadius • formations_mIndex • formations_rIndex
Technologie	Python
Abhängigkeiten	../data_raw/geologicalInformationInCities.csv.csv
Ausgabedateien	*../data_processed/formation_indices.csv*

2.7 processWeatherData

Skriptname	processWeatherData.py
Information	Das Skript berechnet den täglichen Durchschnitt der minimalen und maximalen Temperaturen für jede Wetterstation und bildet daraus einen Monatsdurchschnitt.
Eingabeparameter (Input)	• stationId
Ausgabeparameter (Output)	• stationId • maxTemp pro Monat (Durchschnitt) • min Temp pro Monat (Durchschnitt)
Technologie	Python
Abhängigkeiten	…/data_raw/weatherStationsByCity_50km.csv getWeatherForCity_NOAA-GHCN-daily.R
Ausgabedateien	*../data_final/weatherData_final.csv*

2.8 addCityImagePath

Skriptname	addCityImagePath.py
Information	Das Skript erstellt eine „image_links“-Datei, die später zu cityData_final hinzufügt wird, um auf das jeweilige Stadtbild zu verweisen.
Eingabeparameter (Input)	• cityId • S3 Bucket Link
Ausgabeparameter (Output)	• imageLink
Technologie	Python
Abhängigkeiten	../data_final/cityData_final.csv
Ausgabedateien	*../data_processed/cityImagePath.csv*

2.9 changeCountryCodeOfInfrastructure

Skriptname	changeCountryCodeOfInfrastructure.py
Information	Das Skript ändert ISO-3 country codes in ISO-2 country codes, welche wir in unserem Projekt verwenden. Außerdem werden NaN-Werte entfernt.
Eingabeparameter (Input)	• ISO3 countryCode
Ausgabeparameter (Output)	• ISO2 countryCode
Technologie	Python
Abhängigkeiten	../data_raw/overallQualityOfInfrastructure.csv
Ausgabedateien	*../data_processed/qualityOfInfrastructure.csv*

2.10 levelCreationCityData

Skriptname	levelCreationCityData.py
Information	Die Werte der Indizes werden durch ein Level-System ersetzt. Der Wertebereich eines Index wird in fast allen Fällen durch 5 geteilt. Dadurch entstehen 5 Abstufungen (Bsp.: Level 1 = niedriger Kulturwert, Level 5 = hoher Kulturwert).
Eingabeparameter (Input)	• culture_hIndex, culture_cIndex, culture_cIndex aIndex, culture_cIndex iIndex, culture_cIndex rIndex, culture_cIndex nIndex, Formations_mIndex, Formations_rIndex, beach_Index
Ausgabeparameter (Output)	• cultureIndizes: (0-20% = Level 1, 20-40% = Level 2, 40-60% = Level 3, 60-80% = Level 4, 80-100% = Level 5) • Beach_Index (0-4 = Level 1, 5-30 = Level 2, >30 = Level 3)
Technologie	Python
Abhängigkeiten	../data_final/cityData_final.csv
Ausgabedateien	*../data_final/cityData_level.csv*

2.11 levelCreationCountryData

Skriptname	levelCreationCountryData.py
Information	Die Werte der Indizes werden durch ein Level-System ersetzt. Der Wertebereich eines Index wird in fast allen Fällen durch 5 geteilt. Dadurch entstehen 5 Abstufungen (Bsp.: Level 1 = niedrige Sicherheit, Level 5 = hohe Sicherheit).
Eingabeparameter (Input)	• Infrastructure • Lebenshaltungskosten (cpiIndex, cpiRentIndex, Purchasing Power, groceriesIndex, restaurantIndex) • Sicherheit • Kriminalität
Ausgabeparameter (Output)	• Infrastructure (1-schlecht – 5-gut) • Lebenshaltungskosten (1-billig – 5-teuer) • Sicherheit (1-niedrig – 5-hoch) • Kriminalität (1-niedrig – 5-hoch)
Technologie	Python
Abhängigkeiten	../data_final/countryData_final.csv
Ausgabedateien	*../data_final/countryData_level.csv*

2.12 levelCreationRegionData

Skriptname	levelCreationRegionData.py
Information	Die Werte der Indizes werden durch ein Level-System ersetzt. Der Wertebereich eines Index wird in fast allen Fällen durch 5 geteilt. Dadurch entstehen 5 Abstufungen (Bsp. Level 1 = niedriger Kulturwert, Level 5 = hoher Kulturwert).
Eingabeparameter (Input)	• Mittelwerte: Culture_hIndexMean, culture_cIndexMean, culture_cIndexMean, culture_aIndexMean, culture_cIndex culture_iIndexMean, culture_cIndexMean, culture_rIndexMean, culture_cIndexMean, culture_nIndexMean, Formations_mIndexMean, Formations_rIndexMean, beach_IndexMean
Ausgabeparameter (Output)	• cultureIndizes: (0-20% = Level 1, 20-40% = Level 2, 40-60% = Level 3, 60-80% = Level 4, 80-100% = Level 5) • Beach_Index (0-4 = Level 1, 5-30 = Level 2, >30 = Level 3)
Technologie	Python
Abhängigkeiten	../data_final/regionData_final.csv
Ausgabedateien	*../data_final/regionData_level.csv*

3 data_merge_scripts

3.1 createCityData

Skriptname	createCityData.py
Information	Das Skript erstellt aus allen einzelnen Datenströmen, die zur Definierung einer Stadt dienen, einen Datensatz. Diese wird durch die Merge-Funktion in Python durchgeführt.
Eingabeparameter (Input)	• Städtedaten • Wetterstationen (Wetterschlüssel) • Sehenswürdigkeiten-Indizes • City Image Link
Ausgabeparameter (Output)	• cityData_final als merge
Technologie	Python
Abhängigkeiten	../data_final/weatherData_final.csv ../data_raw/weatherStationsByCity_50km.csv ../data_processed/cityData_clean.csv ../data_final/countryData_final.csv ../data_processed/Indices/culturalIndices.csv ../data_processed/Indices/formationsIndices.csv ../data_processed/Indices/beachIndices.csv ../data_processed/cityImagePath.csv
Ausgabedateien	../data_final/cityData_final

3.2 createCountryData

Skriptname	changeCountryCodeOfInfrastructure.py
Information	Das Skript erstellt aus den vielen einzelnen Landesdaten einen Datensatz durch die Merge-Funktion in Python.
Eingabeparameter (Input)	• Lebenshaltungskosten • Infrastruktur • Flächendaten • Währungsschlüssel • SprachschlüsselCity • Flaggenlinks
Ausgabeparameter (Output)	• CountryData_final als merge
Technologie	Python
Abhängigkeiten	../data_raw/costAndSafetyDataForCountry.csv ../data_processed/qualityOfInfrastructure.csv ../data_raw/landDataForCountry.csv ../data_raw/languageDataForCountry.csv ../data_raw/currencyDataForCountry.csv ../data_raw/flagLinksForCountry.csv
Ausgabedateien	*../data_final/countryData_final.csv*