**Харківський національний економічний університет**

**імені Семена Кузнеця**

**ЗВІТ**

**З ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 16**

**за дисципліною: *“*Технології програмування**”

**на тему: “Обробка даних”**

**Варіант: 11**

**Виконав: студент факультету Інформаційних технологій**

**2 курсу, спец. Кібербезпека,**

**групи 6.04.125.010.21.2**

**Загнібеда Анастасія Олександрівна**

**Перевірив:**

**Венгріна Олена Сергіївна**

**ХНЕУ ім. С. Кузнеця**

**2023**

**Мета:** Метою роботи є набуття навичок технологій розробки програмного коду для обробки масивів даних із використання бібліотек NumPy та Pandas.

**Завданя:** Відповідно до варіанту завдання розробити клас для обробки даних. Розробити конструктор, що приймає шлях до файлу, роздільник, що використовується в CSV файлі, флаг наявності заголовків стовпців в файлі. (роздільник та флаг зробити параметрами із значеннями за замовчуванням). При неможливості прочитати файл із даними генерувати виняток із відповідним повідомленням користувачеві. Конструктор повинен ініціалізувати внутрішнє поле датафрейму, що буде зберігати дані та використовуватися іншими методами. Розробити методи, що узагальнюють завдання, наведені у варіантах (якщо в завданні вказане конкретне значення, то його слід приймати у якості параметру методу). Методи повинні повертати датафрейми із відповідними даними.

Варіант 11

* Список всіх країн
* Вибрати країни з населенням понад 10 млн
* Вибрати країни з населенням понад 10 млн та у яких площа більше ніж в Україні
* Вибрати топ 10 країн в сільському господарстві
* Вибрати країни, які не межують з морем

Я написала наступний код:

*import* numpy *as* np  
  
*class* DataParser:  
 *def \_\_init\_\_*(self, *path*: str = *None*, *separator* = ","):  
 *if not path*:  
 *raise* Exception('Path must be a string')  
 *try*:  
 self.path = *path* file\_array = np.loadtxt(*path*, delimiter=*separator*, dtype=str)  
 self.headers = file\_array[0][0:]  
 self.data\_array = np.array(file\_array[1:])  
 *except*:  
 *raise* Exception('Can\'t open a file')  
 self.data\_enum = {}  
 *for* i *in* range(len(self.headers)):  
 self.data\_enum[self.headers[i]] = i  
  
 *def* country\_list(self):  
 countries = []  
 *for* row *in* self.data\_array:  
 countries.append(row[0])  
 *return* countries  
  
 *def* area\_more\_than\_another\_country\_area(self, *country\_name* = *None*):  
 *if not country\_name*:  
 *raise* Exception('Country name must be provided')  
 countries = []  
 compare\_row = []  
 *for* row *in* self.data\_array:  
 *if country\_name in* row[self.data\_enum['Country']]:  
 compare\_row = row  
 *break* compared\_area = float(compare\_row[self.data\_enum['Area (sq. mi.)']])  
 *for* row *in* self.data\_array:  
 *if* float(row[self.data\_enum['Area (sq. mi.)']]) > compared\_area:  
 countries.append(row[self.data\_enum['Country']])  
 *return* countries  
  
 *def* countries\_which\_has\_population\_more\_than(self, *population* = *None*):  
 *if not population*:  
 *raise* Exception('Population must be provided')  
 countries = []  
 *for* row *in* self.data\_array:  
 *if* int(row[self.data\_enum['Population']]) > *population*:  
 countries.append(row[self.data\_enum['Country']])  
 *return* countries  
  
 *def* inner\_join(self, *arr\_1* = [], *arr\_2* = []):  
 countries = []  
 *for* data *in arr\_1*:  
 *if* data *in arr\_2*:  
 countries.append(data)  
 *return* countries  
  
 *def* countries\_without\_sea(self):  
 countries = []  
 *for* row *in* self.data\_array:  
 arable = row[self.data\_enum['Arable (%)']]  
 *if* arable == '':  
 *continue* crops = row[self.data\_enum['Crops (%)']]  
 *if* crops == '':  
 *continue* other = row[self.data\_enum['Other (%)']]  
 *if* other == '':  
 *continue  
 if* float(arable) + float(crops) + float(other) != 100:  
 countries.append(row[self.data\_enum['Country']])  
 *return* countries  
  
 *def* top\_agricultural\_countries(self, *limit* = 10):  
 *if limit* < 1:  
 *raise* Exception('Limit must be a positive number')  
 arr = np.copy(self.data\_array)  
 *for* i *in* range(len(arr) - 1):  
 *for* j *in* range(i, len(arr) - 1):  
 *if* arr[i][self.data\_enum['Agriculture']] == '' *or* arr[j][self.data\_enum['Agriculture']] == '':  
 *continue  
 if* float(arr[i][self.data\_enum['Pop. Density (per sq. mi.)']]) < float(arr[j][self.data\_enum['Pop. Density (per sq. mi.)']]):  
 arr[i], arr[j] = np.copy(arr[j]), np.copy(arr[i])  
 countries = []  
 limit = *limit if limit* <= len(arr) *else* len(arr)  
 *for* i *in* range(*limit*):  
 countries.append(arr[i][self.data\_enum['Country']])  
 *return* countries  
  
  
  
  
*try*:  
 dp\_error = DataParser()  
*except*:  
 print('Exception')  
*try*:  
 dp\_not\_correct\_file = DataParser("qwe")  
*except*:  
 print('Exception')  
dp = DataParser("countries\_of\_the\_world.csv", ",")  
  
# Список країн  
print(dp.country\_list())  
  
# Список країн з населенням понад 10 млн  
countries\_which\_has\_population\_more\_than\_10\_millions = dp.countries\_which\_has\_population\_more\_than(10000000)  
print(countries\_which\_has\_population\_more\_than\_10\_millions)  
  
# Список країн у яких площа більша ніж в Українни  
area\_more\_than\_Ukraine\_area = dp.area\_more\_than\_another\_country\_area('Ukraine')  
  
# Список країн у яких площа більша ніж в Українни та з населенням понад 10 млн  
print(dp.inner\_join(area\_more\_than\_Ukraine\_area, countries\_which\_has\_population\_more\_than\_10\_millions))  
  
# Топ 10 країн в сільському господарстві  
print(dp.top\_agricultural\_countries())  
  
# Список країн які не межують з морем  
print(dp.countries\_without\_sea())

Результат



Exception

Exception

['Afghanistan ', 'Albania ', 'Algeria ', 'American Samoa ', 'Andorra ', 'Angola ', 'Anguilla ', 'Antigua & Barbuda ', 'Argentina ', 'Armenia ', 'Aruba ', 'Australia ', 'Austria ', 'Azerbaijan ', 'Bahamas. The ', 'Bahrain ', 'Bangladesh ', 'Barbados ', 'Belarus ', 'Belgium ', 'Belize ', 'Benin ', 'Bermuda ', 'Bhutan ', 'Bolivia ', 'Bosnia & Herzegovina ', 'Botswana ', 'Brazil ', 'British Virgin Is. ', 'Brunei ', 'Bulgaria ', 'Burkina Faso ', 'Burma ', 'Burundi ', 'Cambodia ', 'Cameroon ', 'Canada ', 'Cape Verde ', 'Cayman Islands ', 'Central African Rep. ', 'Chad ', 'Chile ', 'China ', 'Colombia ', 'Comoros ', 'Congo. Dem. Rep. ', 'Congo. Repub. of the ', 'Cook Islands ', 'Costa Rica ', "Cote d'Ivoire ", 'Croatia ', 'Cuba ', 'Cyprus ', 'Czech Republic ', 'Denmark ', 'Djibouti ', 'Dominica ', 'Dominican Republic ', 'East Timor ', 'Ecuador ', 'Egypt ', 'El Salvador ', 'Equatorial Guinea ', 'Eritrea ', 'Estonia ', 'Ethiopia ', 'Faroe Islands ', 'Fiji ', 'Finland ', 'France ', 'French Guiana ', 'French Polynesia ', 'Gabon ', 'Gambia. The ', 'Gaza Strip ', 'Georgia ', 'Germany ', 'Ghana ', 'Gibraltar ', 'Greece ', 'Greenland ', 'Grenada ', 'Guadeloupe ', 'Guam ', 'Guatemala ', 'Guernsey ', 'Guinea ', 'Guinea-Bissau ', 'Guyana ', 'Haiti ', 'Honduras ', 'Hong Kong ', 'Hungary ', 'Iceland ', 'India ', 'Indonesia ', 'Iran ', 'Iraq ', 'Ireland ', 'Isle of Man ', 'Israel ', 'Italy ', 'Jamaica ', 'Japan ', 'Jersey ', 'Jordan ', 'Kazakhstan ', 'Kenya ', 'Kiribati ', 'Korea. North ', 'Korea. South ', 'Kuwait ', 'Kyrgyzstan ', 'Laos ', 'Latvia ', 'Lebanon ', 'Lesotho ', 'Liberia ', 'Libya ', 'Liechtenstein ', 'Lithuania ', 'Luxembourg ', 'Macau ', 'Macedonia ', 'Madagascar ', 'Malawi ', 'Malaysia ', 'Maldives ', 'Mali ', 'Malta ', 'Marshall Islands ', 'Martinique ', 'Mauritania ', 'Mauritius ', 'Mayotte ', 'Mexico ', 'Micronesia. Fed. St. ', 'Moldova ', 'Monaco ', 'Mongolia ', 'Montserrat ', 'Morocco ', 'Mozambique ', 'Namibia ', 'Nauru ', 'Nepal ', 'Netherlands ', 'Netherlands Antilles ', 'New Caledonia ', 'New Zealand ', 'Nicaragua ', 'Niger ', 'Nigeria ', 'N. Mariana Islands ', 'Norway ', 'Oman ', 'Pakistan ', 'Palau ', 'Panama ', 'Papua New Guinea ', 'Paraguay ', 'Peru ', 'Philippines ', 'Poland ', 'Portugal ', 'Puerto Rico ', 'Qatar ', 'Reunion ', 'Romania ', 'Russia ', 'Rwanda ', 'Saint Helena ', 'Saint Kitts & Nevis ', 'Saint Lucia ', 'St Pierre & Miquelon ', 'Saint Vincent and the Grenadines ', 'Samoa ', 'San Marino ', 'Sao Tome & Principe ', 'Saudi Arabia ', 'Senegal ', 'Serbia ', 'Seychelles ', 'Sierra Leone ', 'Singapore ', 'Slovakia ', 'Slovenia ', 'Solomon Islands ', 'Somalia ', 'South Africa ', 'Spain ', 'Sri Lanka ', 'Sudan ', 'Suriname ', 'Swaziland ', 'Sweden ', 'Switzerland ', 'Syria ', 'Taiwan ', 'Tajikistan ', 'Tanzania ', 'Thailand ', 'Togo ', 'Tonga ', 'Trinidad & Tobago ', 'Tunisia ', 'Turkey ', 'Turkmenistan ', 'Turks & Caicos Is ', 'Tuvalu ', 'Uganda ', 'Ukraine ', 'United Arab Emirates ', 'United Kingdom ', 'United States ', 'Uruguay ', 'Uzbekistan ', 'Vanuatu ', 'Venezuela ', 'Vietnam ', 'Virgin Islands ', 'Wallis and Futuna ', 'West Bank ', 'Western Sahara ', 'Yemen ', 'Zambia ', 'Zimbabwe ']

['Afghanistan ', 'Algeria ', 'Angola ', 'Argentina ', 'Australia ', 'Bangladesh ', 'Belarus ', 'Belgium ', 'Brazil ', 'Burkina Faso ', 'Burma ', 'Cambodia ', 'Cameroon ', 'Canada ', 'Chile ', 'China ', 'Colombia ', 'Congo. Dem. Rep. ', "Cote d'Ivoire ", 'Cuba ', 'Czech Republic ', 'Ecuador ', 'Egypt ', 'Ethiopia ', 'France ', 'Germany ', 'Ghana ', 'Greece ', 'Guatemala ', 'India ', 'Indonesia ', 'Iran ', 'Iraq ', 'Italy ', 'Japan ', 'Kazakhstan ', 'Kenya ', 'Korea. North ', 'Korea. South ', 'Madagascar ', 'Malawi ', 'Malaysia ', 'Mali ', 'Mexico ', 'Morocco ', 'Mozambique ', 'Nepal ', 'Netherlands ', 'Niger ', 'Nigeria ', 'Pakistan ', 'Peru ', 'Philippines ', 'Poland ', 'Portugal ', 'Romania ', 'Russia ', 'Saudi Arabia ', 'Senegal ', 'South Africa ', 'Spain ', 'Sri Lanka ', 'Sudan ', 'Syria ', 'Taiwan ', 'Tanzania ', 'Thailand ', 'Tunisia ', 'Turkey ', 'Uganda ', 'Ukraine ', 'United Kingdom ', 'United States ', 'Uzbekistan ', 'Venezuela ', 'Vietnam ', 'Yemen ', 'Zambia ', 'Zimbabwe ']

['Afghanistan ', 'Algeria ', 'Angola ', 'Argentina ', 'Australia ', 'Brazil ', 'Burma ', 'Canada ', 'Chile ', 'China ', 'Colombia ', 'Congo. Dem. Rep. ', 'Egypt ', 'Ethiopia ', 'India ', 'Indonesia ', 'Iran ', 'Kazakhstan ', 'Mali ', 'Mexico ', 'Mozambique ', 'Niger ', 'Nigeria ', 'Pakistan ', 'Peru ', 'Russia ', 'Saudi Arabia ', 'South Africa ', 'Sudan ', 'Tanzania ', 'Turkey ', 'United States ', 'Venezuela ', 'Zambia ']

['Monaco ', 'Macau ', 'Singapore ', 'American Samoa ', 'Andorra ', 'Hong Kong ', 'Gaza Strip ', 'Malta ', 'Bermuda ', 'Maldives ']

['Brazil ', 'Croatia ', 'Madagascar ', 'Oman ', 'Zambia ']

Process finished with exit code 0