

به صورت کلی راجع به ابزار های استفاده شده :

از ویژوال استودیو کد و زبان پایتون برای کد نویسی استفاده شده است ، هر سه بخش با استفاده از زبان پایتون هندل شده است.

### بخش اول ، SAX :

از کتابخانه xml.sax استفاده شده است ، یک کلاس تحت عنوان GroupHandler تعریف شده است که علاوه بر \_\_init\_\_ دارای سه فانکشن با عناوین characters ، startElement ، endElement میباشد.  
در این کد از کار کردن با فایل ها و text استفاده شده و چند بار در فایل مینویسیم و میخوانیم . هر بار که یک نوشتن و رایت کردن بر روی یک فایل تمام شد ، در ادامه آن قطعه کد که مخصوص نوشتن بود را کامنت کرده ام.

خب به توضیح قسمت های اصلی کد میپردازیم.  
در فانکشن endElement توانستیم به نوعی پیمایش خطی انجام دهیم (در فایل ایکس ام ال) و تمامی مقادیر و دیتا ها را به یک آرایه تحت عنوان dataset منصوب و در آن درج کنیم .

```
def endElement(self, name):  
  
    # if(name == "cases") and (int(self.cases) >= 100) :  
    if(name == "cases") :  
        # self.barrier.append(" record ")  
        self.barrier.append(str(self.cases))  
        dataset.append(str(self.cases+";"))  
    if(name == "deaths") :  
        self.barrier.append(";"+self.deaths+";")  
        dataset.append(self.deaths+";")  
    if(name == "countriesAndTerritories") :  
        self.barrier.append(self.countriesAndTerritories+";")  
        dataset.append(self.countriesAndTerritories+";")  
    if(name == "continentExp") :  
        self.barrier.append(self.continentExp)  
        dataset.append(self.continentExp+" ;")  
        #self.barrier.append(" end ")
```

سپس در ادامه تمام دیتا های درون dataset را پیمایش کرده و یکی پس از دیگری آن ها را درون فایل sax3.txt درج میکنیم.

```
#done writing  
#with open('sax3.txt', 'w') as f:  
#    for item in dataset:  
#        f.write(" %s " % item)
```

همانطور که بیان شد پس از انجام عملیات رایت ، قطعه کد را کامنت میکنیم که در اجرا های بعدی کد به مشکل بر نخوریم و دوباره کاری نشود.

```
sax3 - Notepad
File Edit Format View Help
222; 4; Afghanistan; Asia ; 122; 0; Afghanistan; Asia ; 124; 3; Afghanistan; Asia ; 172; 0; Afghanistan;
a ; 15; 1; Afghanistan; Asia ; 16; 1; Afghanistan; Asia ; 0; 0; Afghanistan; Asia ; 33; 0; Afghanistan;
a ; 0; 0; Afghanistan; Asia ; 0; 0; Afghanistan; Asia ; 0; 0; Afghanistan; Asia ; 0; 0; Afghanistan; Asia
0; 0; Afghanistan; Asia ; 0; 0; Afghanistan; Asia ; 0; 0; Afghanistan; Asia ; 0; 0; Afghanistan; Asia
0; Albania; Europe ; 16; 1; Albania; Europe ; 28; 2; Albania; Europe ; 29; 2; Albania; Europe ; 27;
Africa ; 129; 4; Algeria; Africa ; 120; 8; Algeria; Africa ; 97; 5; Algeria; Africa ; 99; 10; Algeria
62; 2; Algeria; Africa ; 37; 5; Algeria; Africa ; 12; 3; Algeria; Africa ; 18; 1; Algeria; Africa ;
Algeria; Africa ; 0; 0; Algeria; Africa ; 0; 0; Algeria; Africa ; 0; 0; Algeria; Africa ; 0; 0; Alge
; 0; 0; Algeria; Africa ; 0; 0; Algeria; Africa ; 0; 0; Algeria; Africa ; 0; 0; Algeria; Africa ; 0
6; 4; Andorra; Europe ; 36; 2; Andorra; Europe ; 26; 2; Andorra; Europe ; 41; 1; Andorra; Europe ;
ica ; 0; 0; Angola; Africa ; 0; 0; Angola; Africa ; 0; 0; Angola; Africa ; 0; 0; Angola; Africa ; 2
Anguilla; America ; 0; 0; Anguilla; America ; 0; 0; Anguilla; America ; 0; 0; Anguilla; America ; 0;
ua_and_Barbuda; America ; 1; 0; Antigua_and_Barbuda; America ; 0; 1; Antigua_and_Barbuda; America ; 0; 0;
gua_and_Barbuda; America ; 0; 0; Antigua_and_Barbuda; America ; 4; 0; Antigua_and_Barbuda; America ; 0; 0
7; Argentina; America ; 81; 3; Argentina; America ; 99; 14; Argentina; America ; 80; 5; Argentina; Ame
0; Argentina; America ; 1; 1; Argentina; America ; 6; 0; Argentina; America ; 1; 0; Argentina; Americ
; 0; Armenia; Europe ; 58; 0; Armenia; Europe ; 0; 0; Armenia; Europe ; 52; 2; Armenia; Europe ; 43;
0; Armenia; Europe ; 0; 0; Armenia; Europe ; 0; 0; Armenia; Europe ; 0; 0; Armenia; Europe ; 0; 0;
rope ; 0; 0; Armenia; Europe ; 0; 0; Armenia; Europe ; 0; 0; Armenia; Europe ; 0; 0; Armenia; Europe
Aruba; America ; 2; 0; Aruba; America ; 5; 0; Aruba; America ; 0; 0; Aruba; America ; 5; 0; Aruba;
alia; Oceania ; 100; 2; Australia; Oceania ; 96; 5; Australia; Oceania ; 112; 3; Australia; Oceania ;
nia ; 20; 0; Australia; Oceania ; 6; 0; Australia; Oceania ; 11; 1; Australia; Oceania ; 4; 0; Austr
eania ; 1; 0; Australia; Oceania ; 1; 0; Australia; Oceania ; 0; 0; Australia; Oceania ; 0; 0; Austr
ania ; 0; 0; Australia; Oceania ; 0; 0; Australia; Oceania ; 0; 0; Australia; Oceania ; 0; 0; Austral
tria; Europe ; 529; 18; Austria; Europe ; 564; 20; Austria; Europe ; 805; 22; Austria; Europe ; 522;
3; 0; Austria; Europe ; 0; 0; Austria; Europe ; 2; 0; Austria; Europe ; 0; 0; Austria; Europe ; 0;
Europe ; 0; 0; Austria; Europe ; 0; 0; Austria; Europe ; 0; 0; Austria; Europe ; 0; 0; Austria; Eur
an; Europe ; 33; 3; Azerbaijan; Europe ; 57; 0; Azerbaijan; Europe ; 30; 2; Azerbaijan; Europe ; 56;
```

در ادامه دیتا را ازین فایل که نامش را sax3.txt گذاشتیم میخوانیم . در مرحله بعدی تمامی دیتا را درون یک استرینگ به نام foo میریزیم و با split کردن تمام این استرینگ ، محتوا را به شکل خانه های یک آرایه درون arr که یک آرایه از قبل تعریف شده است درج میکنیم . حالا ما یک آرایه داریم که هر خانه از این آرایه یکی از اجزای دیتای ماست.

```
f = open('sax3.txt', 'r+')
my_file_data = f.read()
f.close()

#print(my_file_data)
foo = ''
foo = my_file_data
arr = foo.split(';');
#print(arr)
```

حاله به مهم ترین بخش و قلب این کد میرسیم . در ادامه کاری که انجام شده است این است که ، یک چالش را در نظر گرفته و حل کرده ایم. چالش ما این است که تمایز بین یک رکورد و اجزای درون آن ، و رکورد بعدی را نمیدانیم زیرا که دیتا ها به صورت خطی در یک آرایه به ترتیب و پشت سر هم درج شده است.

```
222; 4; Afghanistan; Asia    122; 0; Afghanistan; Asia    124; 3;
```

دیتا به شکل بالا به شکل زیر تبدیل شده است


```
arr = [222, 4, Afghanistan, Asia , 122, 0, Afghanistan, Asia , 124, 3; ... ]
```

حال که ما میدانیم هر رکورد 4 دیتا درون خود دارد ، از یک روش که در بحث الگوریتم های موازی با آن آشنا شده ام استفاده میکنم.

این روش در واقع برای بهینه سازی هزینه لوپ ها استفاده میشود و به آن **strided Access** هم گفته میشود. ما حلقه های **for** را به جای پیمایش به ترتیب و پشت سر هم ، 4 تا 4 تا جلو میبریم .یعنی خانه بعدی 4 خانه از قبلی که مشاهده میشود جلو تر است.

سپس در هر پیمایش که متناسب با روش ذکر شده است ، 4 عنصر که حد فاصل بین پیمایش ها میباشد را در یک **element** به شکل استرینگ ذخیره میکنیم ، در نهایت هر پیمایش یک خانه در آرایه از قبل تعریف شده با نام **plus100** درج میکند. حال هر خانه از این آرایه حاوی دیتای کامل یک رکورد که 4 عضو داشت میباشد. در نهایت با پیمایش کردن این آرایه دیتا ها را در **sax.txt** درج میکنیم.

```
124
125 #print(len(arr))
126 for i in range(0,len(arr)-4,4):
127     #if i == len(arr) or i+4 == len(arr):
128     #     break
129     if int(arr[i]) >= 100 :
130         print(arr[i]+";"+str(arr[i+1])+";"+arr[i+2]+";"+arr[i+3])
131         element = arr[i]+";"+str(arr[i+1])+";"+arr[i+2]+";"+arr[i+3]
132         plus100.append(element)
133
134 # writing to sax.txt is Done , task is Done!
135 with open('sax.txt', 'w') as f:
136     for item in plus100:
137         f.write(" %s\n " % item)
```

 sax - Notepad

File	Edit	Format	View	Help
222;	4;	Afghanistan;	Asia	
122;	0;	Afghanistan;	Asia	
124;	3;	Afghanistan;	Asia	
172;	0;	Afghanistan;	Asia	
112;	4;	Afghanistan;	Asia	
105;	2;	Afghanistan;	Asia	
158;	6;	Algeria;	Africa	
199;	7;	Algeria;	Africa	
132;	5;	Algeria;	Africa	
135;	7;	Algeria;	Africa	
126;	6;	Algeria;	Africa	
129;	4;	Algeria;	Africa	
120;	8;	Algeria;	Africa	

## بخش دوم ، DOM :

یک فایل پایتون با نام Dom.py که از کتابخانه xml.dom استفاده کرده ایم. با استفاده از ویژگی های dom توانستیم به دیتای درون تگ های dateRep ، day،month،year دسترسی پیدا کنیم.

با استفاده از یک حلقه دیتای درون dateRep که تاریخ روز و ماه و سال است را پیمایش میکنیم ، یک استرینگ میسازیم که حاوی هر سه ی این دیتا های روز و ماه و سال است ، تحت عنوان res این دیتا که در هر پیمایش حلقه تولید میشود را در هر پیمایش درون یکی از خانه های آرایه ی dates درج میکنیم.

```
dates = []

#Done making a time format and save it to an array
#IT DOES NOT WORK WHEN U ARE DELETING DAY MONTH YEAR
dateRep = doc.getElementsByTagName("dateRep")
#print(dateRep)
for date in dateRep:
    day = date.getElementsByTagName("day")[0]
    months = date.getElementsByTagName("month")[0]
    year = date.getElementsByTagName("year")[0]
    res = str(day.firstChild.data)+"/"+str(months.firstChild.data)+"/"+str(year.firstChild.data)
    dates.append(str(res))
    #print(day.firstChild.data)
#print(dates)
```

همانطور که تصویر گویاست فرمت مناسب برای تاریخ شکل گرفته است . برای مثال : 1/5/2020  
این دیتا ها درون یک فایل text با نام dates درج شده اند.

برای حذف مقادیر خواسته شده اقدام میکنیم ، چهار حلقه نیاز است که یکی از این حلقه ها country\_code را درون فایل xml پیمایش میکند و تک تک عناصرش را حذف میکند ، 3 حلقه بعدی زحمت حذف روز و ماه و سال را که در تگ های جدا گانه بوده اند را میکشند.

```
country_code = doc.getElementsByTagName("countryterritoryCode")
for code in country_code:
    # print(code.parentNode.child(code))
    parent = code.parentNode
    #done deleting it !
    parent.removeChild(code)
days = doc.getElementsByTagName("day")
month = doc.getElementsByTagName("month")
years = doc.getElementsByTagName("year")

for day in days:
    parent = day.parentNode
    #done deleting it !
    parent.removeChild(day)

for mnt in month:
    parent = mnt.parentNode
    #done deleting it !
    parent.removeChild(mnt)

for yr in years:
    parent = yr.parentNode
    #done deleting it !
    parent.removeChild(yr)
```

در نهایت باید تگ `dateRep` را درون فایل `xml` پیمایش کنیم و در هر پیمایش میان این تگ یک تگ جدید حاوی تاریخ با فرمت دلخواهمان درج کنیم ، حالا این دیتای مناسب درج را از آرایه ی `dates` استخراج میکنیم.

```
i = 0
for tag_type in doc.getElementsByTagName('dateRep'):
    #while tag_type.hasChildNodes():
    #    tag_type.removeChild(tag_type.firstChild)
    if i == len(dates):
        break
    tag_type.appendChild(doc.createTextNode(str(dates[i])))
    i=i+1

#print(doc.toxml())

#Done task , Done !
with open("dom.xml","w") as fs:
    fs.write(doc.toxml())
    fs.close()
```

در نهایت هم یک فایل با نام `dom.xml` ایجاد میکنیم.

## بخش سوم ، XSLT :

یک فایل تحت عنوان `xslt.py` حاوی قطعه کد زیر است که مسئله تبدیل فایل `xsl` به `html` را حل کرده است.

```
1 import lxml.html
2 from lxml import etree
3
4 xslt_doc = etree.parse("dom.xsl")
5 xslt_transformer = etree.XSLT(xslt_doc)
6
7 source_doc = etree.parse("dom.xml")
8 output_doc = xslt_transformer(source_doc)
9
10 #print(str(output_doc))
11 #first HTML page with All data is Done!
12 #the task is completed and its Done 100% !
13 output_doc.write("transformed.html", pretty_print=True)
```

خب ما برای به دست آوردن html ابتدا باید یک فایل xslt تشکیل دهیم ، اینکار را انجام داده ایم و بخش اصلی آن در تصویر زیر موجود است ، به کمک یک لوپ و xpath تگ های درون تگ record را پیمایش میکنیم و آن ها را در قطع کدی مشابه html که در همین فایل نوشته شده درج میکنیم .

```

</tr>
<xsl:for-each select="/records/record">
<xsl:if test="countriesAndTerritories = 'Iran'">

    <tr>
        <td>
            <xsl:value-of select='dateRep' />
        </td>
        <td>
            <xsl:value-of select='cases' />
        </td>
        <td>
            <xsl:value-of select='deaths' />
        </td>
        <td>
            <xsl:value-of select='countriesAndTerritories' />
        </td>
        <td>
            <xsl:value-of select='popData2018' />
        </td>
        <td>
            <xsl:value-of select='continentExp' />
        </td>
    </tr>

```

در واقع درون این فایل یک table می نویسیم که به کمک xpath مقادیر را از xml میخوانیم و در تگ همنام درج میکنیم. اگر دقت کنید در یکی از لاین ها به کمک یک شرط if، توانستیم بحث خواندن فقط داده های مربوط به ایران را هندل کنیم.

## Covid19

Date	Cases	Deaths	Country	Population	Continent
1/5/2020	983	71	Iran	81800269	Asia
30/4/2020	1073	80	Iran	81800269	Asia
29/4/2020	1112	71	Iran	81800269	Asia
28/4/2020	991	96	Iran	81800269	Asia
27/4/2020	1153	60	Iran	81800269	Asia
26/4/2020	1134	76	Iran	81800269	Asia
25/4/2020	1168	93	Iran	81800269	Asia
24/4/2020	1030	90	Iran	81800269	Asia
23/4/2020	1194	94	Iran	81800269	Asia
22/4/2020	1297	88	Iran	81800269	Asia
21/4/2020	1294	91	Iran	81800269	Asia

