Basics of Scala

Vicente De Leon

UID: 2001014594

Assignment 6

1. Question 1

   Resources:

   - Largest number among int: https://www.w3resource.com/scala-exercises/basic/scala-basic-exercise-19.php
   - Varargs Scala: https://www.geeksforgeeks.org/scala-varargs/
   - Max method: https://www.geeksforgeeks.org/scala-int-max-method-with-example/

   Observations:

   - We use Scala varargs to place the * on the last argument to make it variable length. We also apply the maximum value of the specified numbers. The code example came from the first resource listed: Largest number among int.

```scala
// Question 1
// largest number among integers: https://www.w3resource.com/scala-exercises/basic/scala-basic-exercise-19.php
// varargs Scala: https://www.geeksforgeeks.org/scala-varargs/
// max method: https://www.geeksforgeeks.org/scala-int-max-method-with-example/
object HW6Question1 {
  private def test(numbers: Int*): Int = { // Scala varargs -> Place * on the last argument to make it variable length.
    numbers.max // returning maximum value of the specified numbers
  }
  def main(args: Array[String]): Unit = {
    println("Result: " + test( numbers = 1,2,3,4,5,6,7,8,9,10));
    println("Result: " + test( numbers = 70,9,8,7,6,5,4,3,2,1));
    println("Result: " + test( numbers = 1,1,1,1,1,1,1,1,1,1));
    println("Result: " + test( numbers = 1,2,3,4,5,5,50,31,2,1));
  }
}
```

```
Result: 10
Result: 70
Result: 1
Result: 50
```

2. Question 2

   Resources:

   - Checking for prime numbers: https://stackoverflow.com/questions/36882103/checking-whether-a-number-is-prime-in-scala
   - Integer user input: https://stackoverflow.com/questions/5055349/how-to-take-input-from-a-user-in-scala

   Observations:

   - The function CheckPrime came from the StackOverFlow code example. This code basically says that if “n” is less than or equal to 1 returns false, else if “n” is equal to 2 return true meaning prime number. Finally, the last expression evaluates to true if “n” is prime. It returns true if “n” is not divisible by any number other than 1 and itself. If “n” is found, it will return false meaning that “n” is not prime in that case. Again, just like HW3, we try using integer user input from StackOverFlow to test the selected input. Please check both sources.

```scala
// Question 2
// Checking for prime numbers: https://stackoverflow.com/questions/36882103/checking-whether-a-number-is-prime-in-scala
// Integer user input: https://stackoverflow.com/questions/5055349/how-to-take-input-from-a-user-in-scala
object HW6Question2 { // StackOverFlow isPrime2 function
  private def CheckPrime(n: Int): Boolean = {
    if (n <= 1)
      false
    else if (n == 2)
      true
    else
      !(2 ≤ until < n).exists(i => n % i == 0)
  }

  def main(args: Array[String]): Unit = { // HW3 -> user input
    println("Please enter a number: ")
    val input = scala.io.StdIn.readInt() // standard way to read integer values
    if (CheckPrime(input))
      println(s"The input integer $input is a prime number.")
    else
      println(s"The input integer $input is not a prime number.")
  }
}
```

```
Please enter a number:
5
The input integer 5 is a prime number.
```

```
Please enter a number:
10
The input integer 10 is not a prime number.
```

3. Question 3

   Resources:

   - New string “n” times code example: https://www.w3resource.com/scala-exercises/basic/scala-basic-exercise-26.php

   Observations:

   - This simple code basically creates a new string by repeating the input string “n” times. Please check resource to explore the original code.

```scala
// Question 3
// code example: https://www.w3resource.com/scala-exercises/basic/scala-basic-exercise-26.php
object HW6Question3 {
  private def copy(str: String, n: Int): String = {
    str * n // new string by repeating string "n" times
  }

  def main(args: Array[String]): Unit = {
    println(copy( str = "Scala", 3)) // 3 times
    println(copy( str = "Carolina", 2)) // twice
    println(copy( str = "2", 7)) // 7 times
  }
}
```

```
ScalaScalaScala
CarolinaCarolina
2222222
```

4. Question 4

   Resources:

   - Reverse: https://www.geeksforgeeks.org/scala-stack-reverse-method-with-example/
   - SplitAt(): https://www.geeksforgeeks.org/scala-list-splitat-method-with-example/

   Observations:

   - We create a list and apply the reverse method to return a new list names “reversedList” with the elements in reverse order. Then, we proceed to apply the “splitAt()” to split the reversed list into a prefix/suffix pair at a stated position. In this case, the position is determined by “reversedList.size/2”. This will give us two new lists: first list and second list.

```scala
// Question 4
// reverse method: https://www.geeksforgeeks.org/scala-stack-reverse-method-with-example/
// splitAt(): https://www.geeksforgeeks.org/scala-list-splitat-method-with-example/
object HW6Question4 {
  def main(args: Array[String]): Unit = {
    val list = List(1, 2, 3, 4, 5, 6)
    val reversedList = list.reverse // apply reverse method
    // applying splitAt to split the given list into a prefix/suffix pair at a stated position.
    val (firstList, secondList) = reversedList.splitAt(reversedList.size / 2) // total elements of reversed list/2
    println(s"Printing the original list: $list")
    println(s"Printing the reversed list: $reversedList")
    println(s"Printing first list: $firstList") // same number of elements as second list
    println(s"Printing second list: $secondList") // same number of elements as first list
  }
}
```

```
Printing the original list: List(1, 2, 3, 4, 5, 6)
Printing the reversed list: List(6, 5, 4, 3, 2, 1)
Printing first list: List(6, 5, 4)
Printing second list: List(3, 2, 1)
```

5. Question 5

   Resources:

   - Code example: https://www.w3resource.com/scala-exercises/list/scala-list-exercise-7.php
   - Distinct: https://www.geeksforgeeks.org/scala-list-distinct-method-with-example/

   Observations:

   - Basic Scala object in which we create a list in which we apply the distinct method to get a new list (result) of elements without any duplicates as shown below.

```scala
// Question 5
// code example: https://www.w3resource.com/scala-exercises/list/scala-list-exercise-7.php
// distinct: https://www.geeksforgeeks.org/scala-list-distinct-method-with-example/
object HW6Question5 {
  def main(args: Array[String]): Unit = {
    val list = List(1, 3, 3, 5, 2, 7, 7, "Carolina", "Rigo", "Rigo")
    println(s"Printing the original list: $list")
    val result = list.distinct // apply distinct to return a new list of elements without any duplicates
    println(s"Unique elements of the list: $result")
  }
}
```

```
Printing the original list: List(1, 3, 3, 5, 2, 7, 7, Carolina, Rigo, Rigo)
Unique elements of the list: List(1, 3, 5, 2, 7, Carolina, Rigo)
```

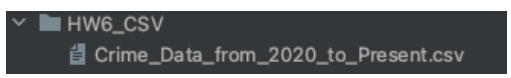
6. Question 6

Resources:

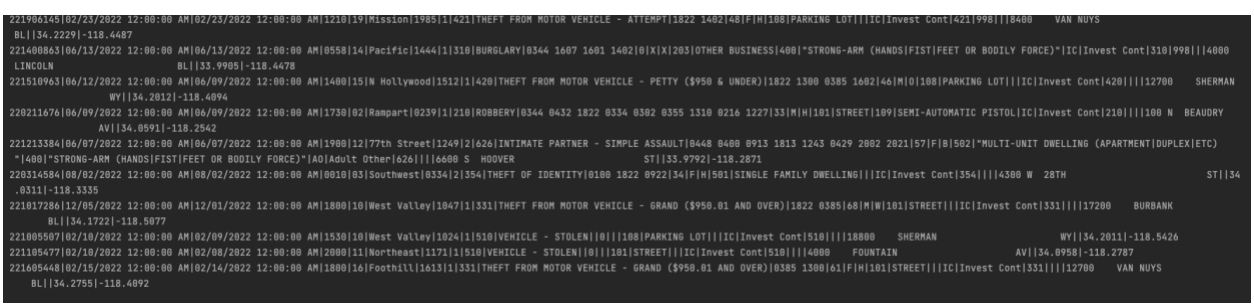
- Reading csv: https://alvinalexander.com/scala/csv-file-how-to-process-open-read-parse-in-scala/
- Reading csv: https://stackoverflow.com/questions/3614041/in-scala-how-to-read-a-simple-csv-file-having-a-header-in-its-first-line
- toList: https://www.geeksforgeeks.org/scala-stack-tolist-method-with-example/
- takeRight: https://www.geeksforgeeks.org/scala-list-takeright-method-with-example/
- mkString: https://www.geeksforgeeks.org/scala-list-mkstring-method-with-a-separator-with-example/
- Setting up IntelliJ for Spark and Scala: https://medium.com/@bartoszgajda55/setting-up-intellij-idea-for-apache-spark-and-scala-development-ce26886552fd
- YouTube tutorial: https://www.youtube.com/watch?v=1x-o9X8DwRI
- Coalesce: https://stackoverflow.com/questions/31610971/spark-repartition-vs-coalesce

Observations:

- This code was quite challenging for me since I have no real experience using IntelliJ besides working with Scala worksheets and Scala objects/Programs. For this code to work I had to create a directory within my HW6 Scala folder so I could store the csv file and have it in my working directory.



- Reading csv (first) resource explains how to read a csv file using a Scala object. We start by reading the csv file from our working directory, and then apply the "toList" method to the results so we can later perform operations like taking the last 10 rows of the CSV file.
- We proceed to use the takeRight method to return the last "n" elements of the list (in this case 10). We apply a for loop.
- The object also explains how the magic is in the code "line.split(",").map(_.trim)". This code splits each line using the comma as a field separator character, and then uses the map method to trim each field to remove leading and trailing blank spaces. Finally, we apply the "mkString" method to separate columns with "|" and close the buffered source.
- This is the results of how it reads the last 10 lines individually:



```
221906145|02/23/2022 12:00:00 AM|02/23/2022 12:00:00 AM|1210|19|Mission|1985|1|421|THEFT FROM MOTOR VEHICLE - ATTEMPT|1822 1402|48|F|H|108|PARKING LOT|||IC|Invest Cont|421|998|||8400    VAN NUYS
 BL||34.2229|-118.4487
221400863|06/13/2022 12:00:00 AM|06/13/2022 12:00:00 AM|0558|14|Pacific|1444|1|310|BURGLARY|0344 1607 1601 1402|0|X|X|203|OTHER BUSINESS|400|"STRONG-ARM (HANDS|FIST|FEET OR BODILY FORCE)"|IC|Invest Cont|310|998|||4000
 LINCOLN                    BL||33.9905|-118.4478
221510963|06/12/2022 12:00:00 AM|06/09/2022 12:00:00 AM|1400|15|N Hollywood|1512|1|420|THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)|1822 1300 0385 1602|46|M|0|108|PARKING LOT|||IC|Invest Cont|420||||12700    SHERMAN
                WY||34.2012|-118.4094
220211676|06/09/2022 12:00:00 AM|06/09/2022 12:00:00 AM|1730|02|Rampart|0239|1|210|ROBBERY|0344 0432 1822 0334 0302 0355 1310 0216 1227|33|M|H|101|STREET|109|SEMI-AUTOMATIC PISTOL|IC|Invest Cont|210||||100 N  BEAUDRY
                AV||34.0591|-118.2542
221213384|06/07/2022 12:00:00 AM|06/07/2022 12:00:00 AM|1900|12|77th Street|1249|2|626|INTIMATE PARTNER - SIMPLE ASSAULT|0448 0400 0913 1813 1243 0429 2002 2021|57|F|B|502|"MULTI-UNIT DWELLING (APARTMENT|DUPLEX|ETC)
 "|400|"STRONG-ARM (HANDS|FIST|FEET OR BODILY FORCE)"|AO|Adult Other|626||||6600 S  HOOVER                    ST||33.9792|-118.2871
220314584|08/02/2022 12:00:00 AM|08/02/2022 12:00:00 AM|0010|03|Southwest|0334|2|354|THEFT OF IDENTITY|0100 1822 0922|34|F|H|501|SINGLE FAMILY DWELLING|||IC|Invest Cont|354||||4300 W  28TH                        ST||34
 .0311|-118.3335
221017286|12/05/2022 12:00:00 AM|12/01/2022 12:00:00 AM|1800|10|West Valley|1047|1|331|THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)|1822 0385|68|M|W|101|STREET|||IC|Invest Cont|331||||17200    BURBANK
        BL||34.1722|-118.5077
221005507|02/10/2022 12:00:00 AM|02/09/2022 12:00:00 AM|1530|10|West Valley|1024|1|510|VEHICLE - STOLEN||0|||108|PARKING LOT|||IC|Invest Cont|510||||18800    SHERMAN                    WY||34.2011|-118.5426
221105477|02/10/2022 12:00:00 AM|02/08/2022 12:00:00 AM|2000|11|Northeast|1171|1|510|VEHICLE - STOLEN||0|||101|STREET|||IC|Invest Cont|510||||4000    FOUNTAIN                  AV||34.0958|-118.2787
221605448|02/15/2022 12:00:00 AM|02/14/2022 12:00:00 AM|1800|16|Foothill|1613|1|331|THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)|0385 1300|61|F|H|101|STREET|||IC|Invest Cont|331||||12700    VAN NUYS
    BL||34.2755|-118.4092
```

This is how the CSV looks like in Colab:

```python
import pandas as pd

df = pd.read_csv('/content/Crime_Data_from_2020_to_Present.csv')
df.head()
```

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION	Cross Street	LAT	LON
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	...	AO	Adult Other	624.0	NaN	NaN	NaN	1100 W 39TH PL	NaN	34.0141	-118.2978
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	...	IC	Invest Cont	624.0	NaN	NaN	NaN	700 S HILL ST	NaN	34.0459	-118.2545
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	Central	155	2	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	...	AA	Adult Arrest	845.0	NaN	NaN	NaN	200 E 6TH ST	NaN	34.0448	-118.2474
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	N Hollywood	1543	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	...	IC	Invest Cont	745.0	998.0	NaN	NaN	5400 CORTEEN PL	NaN	34.1685	-118.4019
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	Mission	1998	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	...	IC	Invest Cont	740.0	NaN	NaN	NaN	14400 TITUS ST	NaN	34.2198	-118.4468

5 rows × 28 columns

This is how it looks using PySpark in Colab:

```python
df2 = spark.read.option("header", "true").csv("Crime_Data_from_2020_to_Present.csv")
df2.show(n = 5, truncate = False)
```

```
+---------+----------------------+----------------------+--------+----+-----------+---------+-------+-----+-----------------------------------------------------+-----------------+-----+
|DR_NO    |Date Rptd             |DATE OCC              |TIME OCC|AREA|AREA NAME  |Rpt Dist No|Part 1-2|Crm Cd|Crm Cd Desc                                          |Mocodes          |Vict
+---------+----------------------+----------------------+--------+----+-----------+---------+-------+-----+-----------------------------------------------------+-----------------+-----+
|010304468|01/08/2020 12:00:00 AM|01/08/2020 12:00:00 AM|2230    |03  |Southwest  |0377     |2      |624  |BATTERY - SIMPLE ASSAULT                             |0444 0913        |36
|190101086|01/02/2020 12:00:00 AM|01/01/2020 12:00:00 AM|0330    |01  |Central    |0163     |2      |624  |BATTERY - SIMPLE ASSAULT                             |0416 1822 1414   |25
|200110444|04/14/2020 12:00:00 AM|02/13/2020 12:00:00 AM|1200    |01  |Central    |0155     |2      |845  |SEX OFFENDER REGISTRANT OUT OF COMPLIANCE            |1501             |0
|191501505|01/01/2020 12:00:00 AM|01/01/2020 12:00:00 AM|1730    |15  |N Hollywood|1543     |2      |745  |VANDALISM - MISDEAMEANOR ($399 OR UNDER)             |0329 1402        |76
|191921269|01/01/2020 12:00:00 AM|01/01/2020 12:00:00 AM|0415    |19  |Mission    |1998     |2      |740  |VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)|0329             |31
+---------+----------------------+----------------------+--------+----+-----------+---------+-------+-----+-----------------------------------------------------+-----------------+-----+
only showing top 5 rows
```

It is a big and large CSV file, and many columns are missing.

Continue:

- Part c is interesting because of how challenging it was. Since I am currently taking the Spark mini course, I had the idea to implement Spark into my Scala program. I've always read how people work with Scala and Apache Spark at the same time, so it came to my mind after failing many times to correctly save the CSV file into JSON file.

- This was quite challenging for me after failing many attempts. The two resources "Setting up IntelliJ for Spark and Scala" and "YouTube tutorial" helped me achieve this.

- The first step was to add the highlighted section (the one you see at the image below) into HW6 "build.sbt". These dependencies helped me incorporate Spark into the program, I ended up updating those dependencies to IntelliJ's suggestion (3.3.2 instead of 2.3.3) and refreshing them.

```
build.sbt   hw6.scala
ThisBuild / version := "0.1.0-SNAPSHOT"

ThisBuild / scalaVersion := "2.13.11"

ThisBuild / libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % "3.3.2",
  "org.apache.spark" %% "spark-sql" % "3.3.2"
)

lazy val root = (project in file("."))
  .settings(
    name := "HW6"
  )
```

- The next step was to run the following code. As you can see, it is the same process I used every time I am going to run a PySpark Code using Google Colab, but this time everything is within a Scala object. I am just applying orderBy() to the df using the “DR_N0” column. Also, I decided to include coalesce within the spark code to write JSON file into 1 single file. I was initially getting 2 JSON files after running the code due to how Spark handles data.

```scala
// Question 6
// reading csv: https://alvinalexander.com/scala/csv-file-how-to-process-open-read-parse-in-scala/
// reading csv: https://stackoverflow.com/questions/3614041/in-scala-how-to-read-a-simple-csv-file-having-a-header-in-its-first-line
// toList: https://www.geeksforgeeks.org/scala-stack-tolist-method-with-example/
// takeRight: https://www.geeksforgeeks.org/scala-list-takeright-method-with-example/
// mkString: https://www.geeksforgeeks.org/scala-list-mkstring-method-with-a-separator-with-example/
// setting up IntelliJ for Spark and Scala: https://medium.com/@bartoszgajda55/setting-up-intellij-idea-for-apache-spark-and-scala-development-ce26886552fd
// youtube tutorial IntelliJ, Spark, Scala: https://www.youtube.com/watch?v=1x-o9X8DwRI
// coalesce: https://stackoverflow.com/questions/31618971/spark-repartition-vs-coalesce
import scala.io.Source
import org.apache.spark.sql.SparkSession

object HW6Question6ABC {
  private def ReadCSV10Lines(): Unit = {
    val bufferedSource = Source.fromFile("/Users/deleonv/Desktop/Scala/HW6/HW6_CSV/Crime_Data_from_2020_to_Present.csv")
    val lines = bufferedSource.getLines().toList // toList method to perform operations
    val last10 = lines.takeRight(10) // takeRight method
    for (line <- last10) { // for loop to get the last 10 rows of the csv file
      val cols = line.split( regex = ",").map(_.trim)
      println(cols.mkString("|")) // column separation
    }
    bufferedSource.close // closing
  }

  private def SparkJSON(): Unit = {
    val spark = SparkSession.builder.appName( name = "HW6Question6AB").master( master = "local").getOrCreate()
    val df = spark.read.format( source = "csv").option("header", "true").load( path = "/Users/deleonv/Desktop/Scala/HW6/HW6_CSV/Crime_Data_from_2020_to_Present.csv")
    df.coalesce( numPartitions = 1).write.json( path = "/Users/deleonv/Desktop/Scala/HW6/HW6_CSV/Crime_Data_2020.json") // writing data into 1 single file
    spark.stop()
  }

  def main(args: Array[String]): Unit = {
    ReadCSV10Lines() // calling first function to print last 10 rows
    SparkJSON() // transform csv file into JSON format
  }
}
```

- By the way why did I decided to orderBy DR_N0? This is like an ID number than has unique values and I personally though it was the easiest way. I checked them myself in Colab:

```python
is_unique = df['DR_NO'].nunique() == len(df['DR_NO'])

if is_unique:
    print("The 'DR_NO' column has unique values.")
else:
    print("The 'DR_NO' column does not have unique values.")
```

```
The 'DR_NO' column has unique values.
```

- After running the object HW6Question6ABC, it returns a JSON file located in the same directory as the CSV file for parts A and B as a JSON file.

- As the above image shows, we can see the directory I created for the CSV file located in HW6, named HW6_CSV. Within that same folder you can see the new generated JSON file.

- If I take the JSON just to check how it looks using PySpark in Colab, this will be my final result (still missing one column because the JSON file has lots of columns and is way too big. The JSON file into df is just a basic example of how it looks within Colab.):

```
df3 = spark.read.json('/content/part-00000-be315d42-5965-4c2e-913d-5a016dad2337-c000.json')
df3.show(n = 5, truncate = False)
```

```
+----+----------+------+--------+--------+--------+--------+-----------------------------------------------------+-----------+----------------------+--------+----------------------+-----+-----------------------
|AREA|AREA NAME |Crm Cd|Crm Cd 1|Crm Cd 2|Crm Cd 3|Crm Cd 4|Crm Cd Desc                                          |Cross Street|DATE OCC              |DR_NO   |Date Rptd             |LAT  |LOCATION
+----+----------+------+--------+--------+--------+--------+-----------------------------------------------------+-----------+----------------------+--------+----------------------+-----+-----------------------
|03  |Southwest |624   |624     |null    |null    |null    |BATTERY - SIMPLE ASSAULT                             |null       |01/08/2020 12:00:00 AM|010304468|01/08/2020 12:00:00 AM|34.0141|1100 W  39TH
|01  |Central   |624   |624     |null    |null    |null    |BATTERY - SIMPLE ASSAULT                             |null       |01/01/2020 12:00:00 AM|190101086|01/02/2020 12:00:00 AM|34.0459|700 S  HILL
|01  |Central   |845   |845     |null    |null    |null    |SEX OFFENDER REGISTRANT OUT OF COMPLIANCE            |null       |02/13/2020 12:00:00 AM|200110444|04/14/2020 12:00:00 AM|34.0448|200 E  6TH
|15  |N Hollywood|745   |745     |998     |null    |null    |VANDALISM - MISDEAMEANOR ($399 OR UNDER)             |null       |01/01/2020 12:00:00 AM|191501505|01/01/2020 12:00:00 AM|34.1685|5400    CORTEEN
|19  |Mission   |740   |740     |null    |null    |null    |VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)|null       |01/01/2020 12:00:00 AM|191921269|01/01/2020 12:00:00 AM|34.2198|14400    TITUS
+----+----------+------+--------+--------+--------+--------+-----------------------------------------------------+-----------+----------------------+--------+----------------------+-----+-----------------------
only showing top 5 rows
```

This is how my final code looks like:

```scala
// Question 6
// reading csv: https://alvinalexander.com/scala/csv-file-how-to-process-open-read-parse-in-scala/
// reading csv: https://stackoverflow.com/questions/3614041/in-scala-how-to-read-a-simple-csv-file-having-a-header-in-its-first-line
// toList: https://www.geeksforgeeks.org/scala-stack-tolist-method-with-example/
// takeRight: https://www.geeksforgeeks.org/scala-list-takeright-method-with-example/
// mkString: https://www.geeksforgeeks.org/scala-list-mkstring-method-with-a-separator-with-example/
// setting up IntelliJ for Spark and Scala: https://medium.com/@bartoszgajda55/setting-up-intellij-idea-for-apache-spark-and-scala-development-ce26886552fd
// youtube tutorial IntelliJ, Spark, Scala: https://www.youtube.com/watch?v=1x-o9X8DwRI
// coalesce: https://stackoverflow.com/questions/31610971/spark-repartition-vs-coalesce
import scala.io.Source
import org.apache.spark.sql.SparkSession

object HW6Question6ABC {
  private def ReadCSV10Lines(): Unit = {
    val bufferedSource = Source.fromFile("/Users/deleony/Desktop/Scala/HW6/HW6_CSV/Crime_Data_from_2020_to_Present.csv")
    val lines = bufferedSource.getLines().toList // toList method to perform operations
    val last10 = lines.takeRight(10) // takeRight method
    for (line <- last10) { // for loop to get the last 10 rows of the csv file
      val cols = line.split( regex = ",").map(_.trim)
      println(cols.mkString("|")) // column separation
    }
    bufferedSource.close // closing
  }

  private def SparkJSON(): Unit = {
    val spark = SparkSession.builder.appName( name = "HW6Question6AB").master( master = "local").getOrCreate()
    val df = spark.read.format( source = "csv").option("header", "true").load( path = "/Users/deleony/Desktop/Scala/HW6/HW6_CSV/Crime_Data_from_2020_to_Present.csv")
    df.coalesce( numPartitions = 1).write.json( path = "/Users/deleony/Desktop/Scala/HW6/HW6_CSV/Crime_Data_2020.json") // writing data into 1 single file
    spark.stop()
  }

  def main(args: Array[String]): Unit = {
    ReadCSV10Lines() // calling first function to print last 10 rows
    SparkJSON() // transform csv file into JSON format
  }
}
```


```
221906145|02/23/2022 12:00:00 AM|02/23/2022 12:00:00 AM|1210|19|Mission|1985|1|421|THEFT FROM MOTOR VEHICLE - ATTEMPT|1822 1402|48|F|H|108|PARKING LOT|||IC|Invest Cont|421|998|||8400    VAN NUYS
 BL||34.2229|-118.4487
221400863|06/13/2022 12:00:00 AM|06/13/2022 12:00:00 AM|0558|14|Pacific|1444|1|310|BURGLARY|0344 1607 1601 1402|0|X|X|203|OTHER BUSINESS|400|"STRONG-ARM (HANDS|FIST|FEET OR BODILY FORCE)"|IC|Invest Cont|310|998|||4000
 LINCOLN                    BL||33.9905|-118.4478
221510963|06/12/2022 12:00:00 AM|06/09/2022 12:00:00 AM|1400|15|N Hollywood|1512|1|420|THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)|1822 1300 0385 1602|46|M|0|108|PARKING LOT|||IC|Invest Cont|420||||12700    SHERMAN
                 WY||34.2012|-118.4094
220211676|06/09/2022 12:00:00 AM|06/09/2022 12:00:00 AM|1730|02|Rampart|0239|1|210|ROBBERY|0344 0432 1822 0334 0302 0355 1310 0216 1227|33|M|H|101|STREET|109|SEMI-AUTOMATIC PISTOL|IC|Invest Cont|210||||100 N  BEAUDRY
                AV||34.0591|-118.2542
221213384|06/07/2022 12:00:00 AM|06/07/2022 12:00:00 AM|1900|12|77th Street|1249|2|626|INTIMATE PARTNER - SIMPLE ASSAULT|0448 0400 0913 1813 1243 0429 2002 2021|57|F|B|502|"MULTI-UNIT DWELLING (APARTMENT|DUPLEX|ETC)
 "|400|"STRONG-ARM (HANDS|FIST|FEET OR BODILY FORCE)"|AO|Adult Other|626||||6600 S  HOOVER                        ST||33.9792|-118.2871
220314584|08/02/2022 12:00:00 AM|08/02/2022 12:00:00 AM|0010|03|Southwest|0334|2|354|THEFT OF IDENTITY|0100 1822 0922|34|F|H|501|SINGLE FAMILY DWELLING|||IC|Invest Cont|354||||4300 W  28TH                          ST||34
 .0311|-118.3335
221017286|12/05/2022 12:00:00 AM|12/01/2022 12:00:00 AM|1800|10|West Valley|1047|1|331|THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)|1822 0385|68|M|W|101|STREET|||IC|Invest Cont|331||||17200    BURBANK
       BL||34.1722|-118.5077
221005507|02/10/2022 12:00:00 AM|02/09/2022 12:00:00 AM|1530|10|West Valley|1024|1|510|VEHICLE - STOLEN||0||||108|PARKING LOT|||IC|Invest Cont|510||||18800    SHERMAN                     WY||34.2011|-118.5426
221105477|02/10/2022 12:00:00 AM|02/08/2022 12:00:00 AM|2000|11|Northeast|1171|1|510|VEHICLE - STOLEN||0|||101|STREET|||IC|Invest Cont|510||||4000    FOUNTAIN                AV||34.0958|-118.2787
221605448|02/15/2022 12:00:00 AM|02/14/2022 12:00:00 AM|1800|16|Foothill|1613|1|331|THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)|0385 1300|61|F|H|101|STREET|||IC|Invest Cont|331||||12700    VAN NUYS
    BL||34.2755|-118.4892
```

```
df3 = spark.read.json('/content/part-00000-be315d42-5965-4c2e-913d-5a016dad2337-c000.json')
df3.show(n = 5, truncate = False)
```

```
+----+----------+------+--------+--------+--------+--------+-----------------------------------------------------+-----------+----------------------+--------+----------------------+-----+-----------------------
|AREA|AREA NAME |Crm Cd|Crm Cd 1|Crm Cd 2|Crm Cd 3|Crm Cd 4|Crm Cd Desc                                          |Cross Street|DATE OCC              |DR_NO   |Date Rptd             |LAT  |LOCATION
+----+----------+------+--------+--------+--------+--------+-----------------------------------------------------+-----------+----------------------+--------+----------------------+-----+-----------------------
|03  |Southwest |624   |624     |null    |null    |null    |BATTERY - SIMPLE ASSAULT                             |null       |01/08/2020 12:00:00 AM|010304468|01/08/2020 12:00:00 AM|34.0141|1100 W  39TH
|01  |Central   |624   |624     |null    |null    |null    |BATTERY - SIMPLE ASSAULT                             |null       |01/01/2020 12:00:00 AM|190101086|01/02/2020 12:00:00 AM|34.0459|700 S  HILL
|01  |Central   |845   |845     |null    |null    |null    |SEX OFFENDER REGISTRANT OUT OF COMPLIANCE            |null       |02/13/2020 12:00:00 AM|200110444|04/14/2020 12:00:00 AM|34.0448|200 E  6TH
|15  |N Hollywood|745   |745     |998     |null    |null    |VANDALISM - MISDEAMEANOR ($399 OR UNDER)             |null       |01/01/2020 12:00:00 AM|191501505|01/01/2020 12:00:00 AM|34.1685|5400    CORTEEN
|19  |Mission   |740   |740     |null    |null    |null    |VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)|null       |01/01/2020 12:00:00 AM|191921269|01/01/2020 12:00:00 AM|34.2198|14400    TITUS
+----+----------+------+--------+--------+--------+--------+-----------------------------------------------------+-----------+----------------------+--------+----------------------+-----+-----------------------
only showing top 5 rows
```

Codes:

```
// Question 1

object HW6Question1 {

  private def test(numbers: Int*): Int = { // Scala varargs -> Place * on the last argument to make it
variable length.

    numbers.max // returning maximum value of the specified numbers

  }

  def main(args: Array[String]): Unit = {

    println("Result: " + test(1,2,3,4,5,6,7,8,9,10));

    println("Result: " + test(70,9,8,7,6,5,4,3,2,1));

    println("Result: " + test(1,1,1,1,1,1,1,1,1));

    println("Result: " + test(1,2,3,4,5,5,50,31,2,1));

  }

}
```

```
// Question 2
object HW6Question2 { // StackOverFlow isPrime2 function
  private def CheckPrime(n: Int): Boolean = {
    if (n <= 1)
      false
    else if (n == 2)
      true
    else
      !(2 until n).exists(i => n % i == 0)
  }

  def main(args: Array[String]): Unit = { // HW3 -> user input
    println("Please enter a number: ")
    val input = scala.io.StdIn.readInt() // standard way to read integer values
    if (CheckPrime(input))
      println(s"The input integer $input is a prime number.")
    else
      println(s"The input integer $input is not a prime number.")
  }
}
```

```
// Question 3
object HW6Question3 {
  private def copy(str: String, n: Int): String = {
    str * n // new string by repeating string "n" times
  }

  def main(args: Array[String]): Unit = {
    println(copy("Scala", 3)) // 3 times
    println(copy("Carolina", 2)) // twice
    println(copy("2", 7)) // 7 times
  }
}


// Question 4
object HW6Question4 {
  def main(args: Array[String]): Unit = {
    val list = List(1, 2, 3, 4, 5, 6)
    val reversedList = list.reverse // apply reverse method
    // applying splitAt to split the given list into a prefix/suffix pair at a stated position.
    val (firstList, secondList) = reversedList.splitAt(reversedList.size / 2) // total elements of reversed list/2
    println(s"Printing the original list: $list")
    println(s"Printing the reversed list: $reversedList")
    println(s"Printing first list: $firstList") // same number of elements as second list
    println(s"Printing second list: $secondList") // same number of elements as first list
  }
}
```

```
// Question 5
object HW6Question5 {
  def main(args: Array[String]): Unit = {
    val list = List(1, 3, 3, 5, 2, 7, 7, "Carolina", "Rigo", "Rigo")
    println(s"Printing the original list: $list")
    val result = list.distinct // apply distinct to return a new list of elements without any duplicates
    println(s"Unique elements of the list: $result")
  }
}
```

```
// Question 6

import scala.io.Source

import org.apache.spark.sql.SparkSession

object HW6Question6ABC {

  private def ReadCSV10Lines(): Unit = {

    val bufferedSource =
Source.fromFile("/Users/deleonv/Desktop/Scala/HW6/HW6_CSV/Crime_Data_from_2020_to_Present.c
sv")

    val lines = bufferedSource.getLines().toList // toList method to perform operations

    val last10 = lines.takeRight(10) // takeRight method

    for (line <- last10) { // for loop to get the last 10 rows of the csv file

      val cols = line.split(",").map(_.trim)

      println(cols.mkString("|")) // column separation

    }

    bufferedSource.close // closing

  }

  private def SparkJSON(): Unit = {

    val spark = SparkSession.builder.appName("HW6Question6AB").master("local").getOrCreate()

    val df = spark.read.format("csv").option("header",
"true").load("/Users/deleonv/Desktop/Scala/HW6/HW6_CSV/Crime_Data_from_2020_to_Present.csv")

    df.coalesce(1).write.json("/Users/deleonv/Desktop/Scala/HW6/HW6_CSV/Crime_Data_2020.json") //
writing data into 1 single file

    spark.stop()

  }

  def main(args: Array[String]): Unit = {

    ReadCSV10Lines() // calling first function to print last 10 rows

    SparkJSON() // transform csv file into JSON format

  }

}
```