PKA-RDD

February 7, 2024

[1]:
```
import pyspark
from pyspark import SparkConf,SparkContext
```

[2]:
```
conf=SparkConf().setAppName('RDD').setMaster('local[*]')
sc=SparkContext(conf=conf).getOrCreate()
```

[3]:
```
Result=sc.textFile('file:///G:/anaconda/Lib/site-packages/pyspark/licenses')
```

[4]:
```
Result.getNumPartitions()
```

[4]:
```
21
```

[5]:
```
Result.count()
```

[5]:
```
568
```

[6]:
```
Result.take(1)
```

[6]:
```
['The MIT License (MIT)']
```

[7]:
```
Result2=sc.wholeTextFiles('file:///G:/anaconda/Lib/site-packages/pyspark/
↪licenses')
```

[8]:
```
Result2.count()
```

[8]:
```
21
```

[9]:
```
Result2.getNumPartitions()
```

[9]:
```
2
```

[10]:
```
Result2.take(1)
```

[10]:
```
[('file:/G:/anaconda/Lib/site-packages/pyspark/licenses/LICENSE-AnchorJS.txt',
  'The MIT License (MIT)\n\nCopyright (c) <year> <copyright
holders>\n\nPermission is hereby granted, free of charge, to any person
obtaining a copy\nof this software and associated documentation files (the
"Software"), to deal\nin the Software without restriction, including without
limitation the rights\nto use, copy, modify, merge, publish, distribute,
```

```
sublicense, and/or sell\ncopies of the Software, and to permit persons to whom
the Software is\nfurnished to do so, subject to the following conditions:\n\nThe
above copyright notice and this permission notice shall be included in\nall
copies or substantial portions of the Software.\n\nTHE SOFTWARE IS PROVIDED "AS
IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR\nIMPLIED, INCLUDING BUT NOT
LIMITED TO THE WARRANTIES OF MERCHANTABILITY,\nFITNESS FOR A PARTICULAR PURPOSE
AND NONINFRINGEMENT. IN NO EVENT SHALL THE\nAUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER\nLIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM,\nOUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN\nTHE SOFTWARE.')]
```

[11]:
```
Result3=sc.parallelize([[1,2,3],[4],[5,6,7],[8],[[[9,10]]]],5)
```

[12]:
```
Result3.getNumPartitions()
```

[12]:
```
5
```

[13]:
```
y=Result3.map(lambda x:x)
y.collect()
```

[13]:
```
[[1, 2, 3], [4], [5, 6, 7], [8], [[[9, 10]]]]
```

[14]:
```
y=Result3.flatMap(lambda x:x)
z=y.collect()
z
```

[14]:
```
[1, 2, 3, 4, 5, 6, 7, 8, [[9, 10]]]
```

[15]:
```
def recursively_flatten(item):
    if isinstance(item, list):
        return [element for sublist in item for element in␣
 ↪recursively_flatten(sublist)]
    else:
        return [item]


z=sc.parallelize(z)
z=z.flatMap(recursively_flatten)
```

[16]:
```
z.collect()
```

[16]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```