

Assignment 13.2

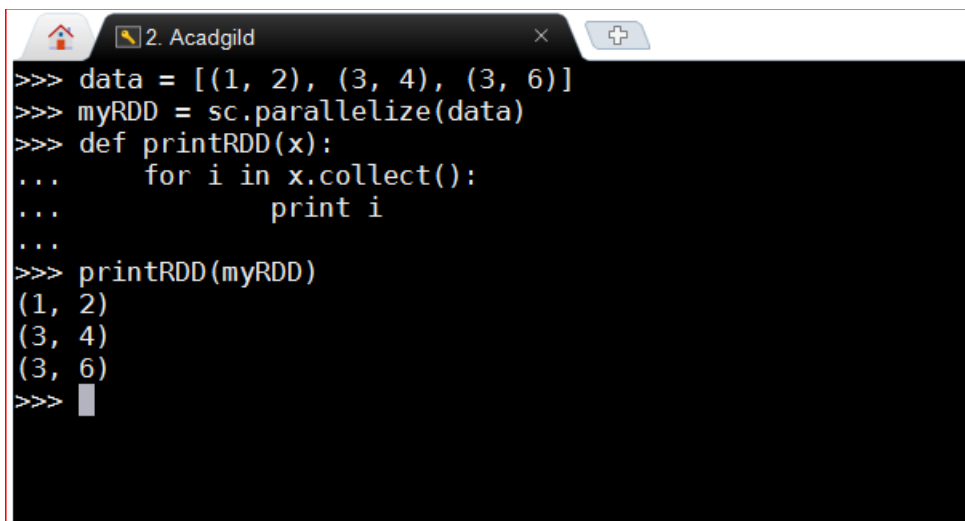
1. Create a simple pairRDD of (1, 2), (3, 4), (3, 6).

```
data = [(1, 2), (3, 4), (3, 6)]          # create list of pairs (tuples)

myRDD = sc.parallelize(data)            # create RDD

def printRDD(x):                         # function to print RDD
    for i in x.collect():
        print i

printRDD(myRDD)                          # RDD printed
```

A screenshot of a Jupyter Notebook terminal window. The window has a title bar with a home icon, a tab labeled '2. Acadgild', and a close button. The terminal shows the following code being executed in a Python shell:

```
>>> data = [(1, 2), (3, 4), (3, 6)]
>>> myRDD = sc.parallelize(data)
>>> def printRDD(x):
...     for i in x.collect():
...         print i
...
>>> printRDD(myRDD)
(1, 2)
(3, 4)
(3, 6)
>>> █
```

2. Transform an RDD of ("a","b","c","d","e") to PairRDD (a,0), (b,1), (c,2), (d,3), (e,4)

```
data = ["a","b","c","d","e"]           # create a list of elements as mentioned

myRDD = sc.parallelize(data)            #creating RDD with list of values

def printRDD(x):                         # function to print RDD
    for i in x.collect():
        print i

count = -1                              # counter set to -1
```

<code>def evaluate(x):</code>	<code># evaluate function to prepare paired element or value for the keys</code>
<code> global count</code>	<code># global variable declared for the computation on the RDD</code>
<code> count = count+1</code>	
<code> return (x,count)</code>	<code># returns the ('a',0) for first element, subsequently ('b',1)</code>
<code>printRDD(myRDD)</code>	<code># print myRDD</code>
<code>mappedRDD = myRDD.map(evaluate)</code>	<code># evaluating pair of the keys</code>
<code>printRDD(mappedRDD)</code>	<code># printing final RDD.</code>

```

>>> data = ["a","b","c","d","e"]
>>> myRDD = sc.parallelize(data)
>>> def printRDD(x):
...     for i in x.collect():
...         print i
...
>>> count = -1
>>> def evaluate(x):
...     global count
...     count = count+1
...     return (x,count)
...
>>>
>>> printRDD(myRDD)
a
b
c
d
e
>>> mappedRDD = myRDD.map(evaluate)
>>> printRDD(mappedRDD)
('a', 0)
('b', 1)
('c', 2)
('d', 3)
('e', 4)
>>>

```