Untitled2

Below is a comprehensive list of **15 Transformation APIs** and **15 Action APIs** in PySpark, with corresponding code examples and outputs.

Transformation APIs¶

1. *map*¶

Applies a function to each element in the RDD.

```
rdd = sc.parallelize([1, 2, 3])
result = rdd.map(lambda x: x * 2).collect()
print(result)
# Output: [2, 4, 6]
```

2. *filter*¶

Filters elements based on a condition.

```
result = rdd.filter(lambda x: x % 2 == 0).collect()
print(result)
# Output: [2]
```

3. *flatMap*¶

Splits each element into multiple elements.

```
rdd = sc.parallelize(["a b", "c d"])
result = rdd.flatMap(lambda x: x.split(" ")).collect()
print(result)
# Output: ['a', 'b', 'c', 'd']
```

4. *distinct*¶

Removes duplicate elements.

```
rdd = sc.parallelize([1, 2, 2, 3])
result = rdd.distinct().collect()
print(result)
# Output: [1, 2, 3]
```

5. *union*¶

Combines two RDDs.

```
rdd2 = sc.parallelize([4, 5])
result = rdd.union(rdd2).collect()
```

```
print(result)
# Output: [1, 2, 3, 4, 5]
```

6. *intersection*¶

Finds common elements between two RDDs.

```
rdd2 = sc.parallelize([2, 3, 4])
result = rdd.intersection(rdd2).collect()
print(result)
# Output: [2, 3]
```

7. *cartesian*¶

Computes the cartesian product of two RDDs.

```
rdd2 = sc.parallelize([4, 5])
result = rdd.cartesian(rdd2).collect()
print(result)
# Output: [(1, 4), (1, 5), (2, 4), (2, 5), (3, 4), (3, 5)]
```

8. *groupBy*¶

Groups elements by a function.

```
result = rdd.groupBy(lambda x: x % 2).mapValues(list).collect()
print(result)
# Output: [(1, [1, 3]), (0, [2])]
```

9. *reduceByKey*¶

Applies a function to reduce values by key.

```
rdd = sc.parallelize([("a", 1), ("b", 2), ("a", 3)])
result = rdd.reduceByKey(lambda x, y: x + y).collect()
print(result)
# Output: [('a', 4), ('b', 2)]
```

10. *sortBy*¶

Sorts RDD by a function.

```
result = rdd.sortBy(lambda x: x[1]).collect()
print(result)
# Output: [('b', 2), ('a', 3)]
```

11. *coalesce*¶

Reduces the number of partitions.

```
result = rdd.coalesce(1).getNumPartitions()
print(result)
# Output: 1
```

12. keyBy¶

Creates a PairRDD with a key-value pair.

```
result = rdd.keyBy(lambda x: x[1]).collect()
print(result)
# Output: [(1, ('a', 1)), (2, ('b', 2)), (3, ('a', 3))]
```

13. partitionBy¶

Partitions an RDD using a partitioning function.

```
rdd = sc.parallelize([("a", 1), ("b", 2)], 2)
result = rdd.partitionBy(2).getNumPartitions()
print(result)
# Output: 2
```

14. zip¶

Combines two RDDs element-wise.

```
rdd2 = sc.parallelize([4, 5, 6])
result = rdd.zip(rdd2).collect()
print(result)
# Output: [(1, 4), (2, 5), (3, 6)]
```

15. subtract¶

Removes elements of another RDD.

```
rdd2 = sc.parallelize([2, 3])
result = rdd.subtract(rdd2).collect()
print(result)
# Output: [1]
```

Action APIs¶

1. collect¶

Returns all elements.

```
result = rdd.collect()
print(result)
# Output: [1, 2, 3]
```

2. count¶

Counts the number of elements.

```
result = rdd.count()
print(result)
# Output: 3
```

3. `first`¶

Gets the first element.

```
result = rdd.first()
print(result)
# Output: 1
```

4. `take`¶

Takes the first n elements.

```
result = rdd.take(2)
print(result)
# Output: [1, 2]
```

5. `reduce`¶

Aggregates elements using a function.

```
result = rdd.reduce(lambda x, y: x + y)
print(result)
# Output: 6
```

6. `countByKey`¶

Counts occurrences of each key.

```
rdd = sc.parallelize([("a", 1), ("b", 2), ("a", 3)])
result = rdd.countByKey()
print(dict(result))
# Output: {'a': 2, 'b': 1}
```

7. `foreach`¶

Applies a function to each element.

```
rdd.foreach(lambda x: print(x))
# Output (printed): 1, 2, 3
```

8. `saveAsTextFile`¶

Saves RDD to a text file.

```
rdd.saveAsTextFile("/tmp/output")
print("Data saved to /tmp/output")
```

9. `lookup`¶

Looks up values by key.

```
result = rdd.lookup("a")
print(result)
# Output: [1, 3]
```

10. takeSample¶

Returns a sample of elements.

```
result = rdd.takeSample(False, 2)
print(result)
# Output: [2, 1] (varies)
```

11. takeOrdered¶

Takes the first n elements ordered.

```
result = rdd.takeOrdered(2, key=lambda x: -x[1])
print(result)
# Output: [('a', 3), ('b', 2)]
```

12. max¶

Finds the maximum element.

```
result = rdd.max()
print(result)
# Output: 3
```

13. min¶

Finds the minimum element.

```
result = rdd.min()
print(result)
# Output: 1
```

14. sum¶

Computes the sum of elements.

```
result = rdd.sum()
print(result)
# Output: 6
```

15. mean¶

Computes the mean of elements.

```
result = rdd.mean()
print(result)
# Output: 2.0
```

In [ ]: