1) What kind of object does the ParDo transform expect?

A. A DoFn subclass
B. A DoFn super class
C. A DoFn abstract class
2) The process method in the DoFn subclass takes as input a single element from a PCollection.

A. True
B. False
3) The **GroupByKey** transform takes a collection of key/value pairs.

A. True
B. False
4) ParDo most closely resembles which SQL operation?

A. FROM clause
B. WHERE clause
C. ORDER BY clause
D. JOIN clause
5) CoGroupByKey most closely resembles which SQL operation?

A. FROM clause
B. WHERE clause
C. ORDER BY clause
D. JOIN clause
Recall: **ParDo Transform**

- Maps 1 input element to (1, 0, many) output elements
- Invokes a user-specified function on each of the elements of the input PCollection
- User code is implemented as a subclass of `DoFn` with a `process(self, element)` method
- Input elements are processed independently and in parallel
- Output elements are bundled into a new PCollection
- Typical usage: filtering, formatting, extracting parts of data, performing computations on data elements
**ParDo Side Inputs**

- A side input is an optional input passed to `DoFn`
- Passed as an extra argument to `process` method:

  ```python
  process(self, element, side_input1)
  ```

- Side inputs can be ordinary values or entire `PCollections`
- `DoFn` reads side inputs while processing an individual element

- Multiple side inputs per `DoFn` are supported:

  ```python
  process(self, element, side_input1, side_input2,
          side_inputn)
  ```
Hands-on Exercises

git pull origin master
Hands-on Exercise 1

Run Student_beam_dataflow2.py
iClicker Question 1

How many records are in the resulting Student_Beam_DF table?

A. 10
B. 12
C. 15
D. None of the above
Hands-on Exercise 2

How should we fix the `college_modeled.Class` table?
How should we fix the `college_modeled.Class` table?

A. Update the primary key
B. Assign a new primary key
C. Remove duplicate primary key values
D. All of the above
How should we fix the `college_modeled.Takes` table?
iClicker Question 3

How should we fix the college_modeled.Takes table?

A. Standardize the cno values
B. Add the cid field
C. Remove the cno field
D. All of the above
Demo: Takes_beam.py

Show Side Inputs
CoGroupByKey Transform

- Takes two or more PCollections as input
- Every element in the input is a (key, value) pair
- Groups values from all input PCollections by common key
- Produces a PCollection as output where each element is a (key, value) pair
- Output value is a list of dictionaries containing all data associated with unique key
- Analogous to SQL’s FULL OUTER JOIN operation
CoGroupByKey Transform

```python
def MakeTuple():
    # Define your implementation here

q1 = 'SELECT sid, cid, grade FROM college_modeled.Takes_Beam'
q2 = 'SELECT cid, cno, cname FROM college_modeled.Class_Beam'

takes_pcoll = p | 'Run Q1' >> beam.io.Read(beam.io.BigQuerySource(query=q1))
class_pcoll = p | 'Run Q2' >> beam.io.Read(beam.io.BigQuerySource(query=q2))

takes_tuple = takes_pcoll | 'Takes Tuple' >> beam.ParDo(MakeTuple())
class_tuple = class_pcoll | 'Class Tuple' >> beam.ParDo(MakeTuple())

joined_pcoll = (takes_tuple, class_tuple) | 'Join' >> beam.CoGroupByKey()
```
Flatten Transform

- Takes a list of PCollections as input
- Produces a single PCollection as output
- Results contain all the elements from the input PCollections
- Note: Input PCollections must have matching schemas

```python
a_pcoll = p | 'Read File 1' >> ReadFromText('oscars_data_archive.tsv')
b_pcoll = p | 'Read File 2' >> ReadFromText('oscars_data_2019.tsv')

# Union the two PCollections

# a_pcoll, b_pcoll = (a_pcoll, b_pcoll) | 'Merge PCollections' >> beam.Flatten()
```
Milestone 6