ANSWERS

A11.9 Big data

Databases and big data · A level · AQA 7517 4.11.1, Eduqas A500QS 2.5 · about 60 min

BugBotLab

What this lesson is about

Volume, velocity and variety; why one server is not enough; distributed processing with map and reduce; the fact-based model and graph schema.

- Volume, velocity and variety
- Why one server is not enough
- Distributed processing
- The fact-based model
- Graph schema

Questions

6 marks in all

1. Which are the three Vs usually used to describe big data? [1 mark]

Tick every answer that is true.

- ☐ **A** Volume
- ☐ **B** Velocity
- ☐ **C** Variety
- ☐ **D** Validation

> **Answer:** A, B, C. Too much data, arriving too fast, in too many forms.

2. Camera images, voice clips and free text arrive alongside sensor tables. Which V does this show? [1 mark]

- ☐ **A** Variety
- ☐ **B** Volume
- ☐ **C** Velocity
- ☐ **D** Veracity

> **Answer:** A. Data in many forms, much of it unstructured, is variety.

3. Why does functional programming suit distributed processing of big data? [1 mark]

- ☐ **A** Functions with no side effects on immutable data give the same result whichever machine runs them, in any order
- ☐ **B** Functional programs never need more than one server
- ☐ **C** Functional languages store data in relational tables
- ☐ **D** Functional programs cannot fail

> **Answer:** A. Statelessness and immutability let the work be split, run in parallel and repeated safely.

4. Which describes the fact-based model? [1 mark]

- ☐ **A** Immutable, timestamped facts are only ever added; the current state is worked out from them
- ☐ **B** Each fact is overwritten when it changes
- ☐ **C** Facts are stored in 3NF tables with foreign keys
- ☐ **D** Only the latest value of each attribute is stored

> **Answer:** A. Facts are never updated or deleted, which keeps the full history and survives mistakes.

5. In a graph schema, what does an edge represent? [1 mark]

- ☐ **A** A relationship between two nodes
- ☐ **B** An entity
- ☐ **C** A property of a node
- ☐ **D** A timestamp

> **Answer:** A. Nodes are entities, edges are the relationships between them, and properties are the data on a node.

6. What does this print? [1 mark]

```
from functools import reduce
log = ["Ada,bump", "Bolt,bump", "Ada,bump", "Ada,stall"]
pairs = [(line.split(",")[0], 1) for line in log if line.endswith("bump")]
groups = {}
for key, value in pairs:
    groups.setdefault(key, []).append(value)
for key in sorted(groups):
    print(key, reduce(lambda a, b: a + b, groups[key]))
```

> **Answer:**
>
> ```
> Ada 2
> Bolt 1
> ```
>
> The map keeps only bumps as (robot, 1), the shuffle groups them, and the reduce adds each group.

The task: map, shuffle, reduce

Three servers each hold part of the robots' event log. `servers` is a list of three lists; every item is a string `"<robot>,<event>"`, where event is `bump` or `stall`. 1. Write `mapper(line)`: it takes one line and returns a list of *(key, value)* pairs, `[(robot, 1)]` if the event is `bump`, or `[]` otherwise. 2. Call `mapper` on every line of every server, and shuffle: group the values by key. 3. Write `reducer(key, values)`: it takes a robot name and the list of its values, and returns `(key, total)`. 4. Call `reducer` once for each robot, in name order, and print each result as `<robot> <bumps>`, such as `Cog 2`. The robot does not move.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

# each server holds part of the event log: "robot,event"
servers = [
    ["Ada,bump", "Bolt,bump", "Ada,stall"],
    ["Cog,bump", "Ada,bump", "Bolt,stall", "Bolt,bump"],
    ["Ada,bump", "Cog,bump"],
]
```

The hint students can ask for: The mapper sees one line and knows nothing else: it gives back a list of key and value pairs, empty for a line that is not a bump. The shuffle gathers every value with the same key. The reducer then sees one key and all its values.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

# each server holds part of the event log: "robot,event"
servers = [
    ["Ada,bump", "Bolt,bump", "Ada,stall"],
    ["Cog,bump", "Ada,bump", "Bolt,stall", "Bolt,bump"],
    ["Ada,bump", "Cog,bump"],
]

def mapper(line):
    robot, event = line.split(",")
    return [(robot, 1)] if event == "bump" else []

def reducer(key, values):
    return (key, sum(values))

mapped = [pair for server in servers for line in server for pair in mapper(line)]

groups = {}
for key, value in mapped:
    groups.setdefault(key, []).append(value)

for key in sorted(groups):
    robot, total = reducer(key, groups[key])
    print(robot, total)
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.