ANSWERS

A15.2 Command words and levels of response

Exam preparation · A level · OCR H446 2.2.1, AQA 7517 4.4.1.2, Eduqas A500QS 1.8 · about 45 min

What this lesson is about

What each command word asks for, how extended answers are marked in levels, and a marker that levels answers with regular expressions.

- Command words at A level
- How short answers are marked
- How extended answers are marked
- Planning a 9-mark answer
- A model of a marker

Questions

5 marks in all

1. What does the command word evaluate require that describe does not? [1 mark]

- ☐ **A** Weighing strengths and weaknesses and reaching a judgement
- ☐ **B** Stating the fact with no explanation
- ☐ **C** Drawing a diagram
- ☐ **D** Giving the answer to a calculation

> **Answer:** A. Evaluate means weigh up and judge; describe only says what something is or does.

2. How is an extended answer marked by levels of response? [1 mark]

- ☐ **A** The examiner decides which level's descriptor fits the whole answer best, then places it within that level
- ☐ **B** One mark is given for every fact, up to the total
- ☐ **C** Only the conclusion is marked
- ☐ **D** The longest answer gets the most marks

> **Answer:** A. Levels are judged on the quality of the whole answer; indicative content is a guide, not a checklist.

3. Which sentence explains, rather than describes? [1 mark]

- ☐ **A** A stack suits undoing moves, because the last move made is the first that must be reversed
- ☐ **B** A stack is last in, first out
- ☐ **C** A stack has push and pop operations
- ☐ **D** A stack is an abstract data type

> **Answer:** A. An explanation gives a reason linked to the situation, with a word like because.

4. Which of these usually move an extended answer up a level? [1 mark]

Tick every answer that is true.

- ☐ **A** Developing each point with why it matters
- ☐ **B** Applying points to the scenario in the question
- ☐ **C** A conclusion that decides
- ☐ **D** Listing more facts in single lines
- ☐ **E** Writing the question out again

> **Answer:** A, B, C. Development, application, balance and judgement lift the level; more undeveloped facts do not.

5. This regular expression finds developed points. What does the code print? [1 mark]

```
import re
sentences = ["It is fast, so staff save time.", "It is also cheap.", "Sometimes it helps because it is quiet."]
print(sum(1 for s in sentences if re.search(r"\b(because|so)\b", s)))
```

> **Answer:**
>
> ```
> 2
> ```
>
> \b makes so and because whole words, so also and sometimes do not match: two sentences are developed.

The task: level the answers

Four students answered the face recognition question. Write a program that levels each answer using these rules. It is a crude model of a real examiner, but the rules are exactly defined. - Split an answer into **sentences** with `re.split(r"(?<=[.!?])\s+", answer.strip())`. - A sentence is the **conclusion** if it begins with `On balance`, `Overall` or `In conclusion`, ignoring capitals. A conclusion sentence is not counted as a point. - Any other sentence is a **point** if it contains a word from `FOR_WORDS` or `AGAINST_WORDS`, ignoring capitals (a word counts if it appears anywhere in the sentence). - A point is **developed** if it also contains `because` or `so` as a whole word, ignoring capitals. A developed point is on the **for** side if it contains a word from `FOR_WORDS`, and on the **against** side if it contains one from `AGAINST_WORDS` (it can be both). Write `analyse(answer)`, returning a tuple `(points, developed, both_sides, conclusion)`: two whole numbers, then `True` if the developed points include both sides, then `True` if there is a conclusion. Write `level_and_mark(points, developed, both_sides, conclusion)`, returning a tuple `(level, mark)`: - level 3 if `developed` is at least 3, `both_sides` is true and there is a conclusion; the mark is `7 + min(2, developed - 3)`; - otherwise level 2 if `developed` is at least 2; the mark is `4 + min(2, developed - 2)`; - otherwise level 1 if `points` is at least 1; the mark is `min(3, points)`; - otherwise level 0 and 0 marks. For each answer in order, print `<name>: level <level>, <mark> marks (<points> points, <developed> developed)`. The robot stays still.

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

import re

FOR_WORDS = ["benefit", "advantage", "helps"]
AGAINST_WORDS = ["risk", "drawback", "harm"]
answers = [
    ("Ada", "A benefit is that parcels reach the right teacher, because the robot
recognises them. "
            "A risk is that faces are biometric data, so the school must have a lawful
basis and keep the data secure. "
            "Another risk is bias, because a detector trained mostly on adults may fail for
younger students. "
            "On balance the school should use a PIN instead, because it gives the benefit
without storing faces."),
    ("Bolt", "One benefit is speed, so staff save time. Another benefit is accuracy,
because no parcel goes to the wrong room. "
             "There is a risk to privacy."),
    ("Cog", "Face recognition is a benefit. It is also a risk."),
    ("Dot", "Robots are interesting and schools are busy places."),
]
```

The hint students can ask for: Split each answer into sentences first. Decide for each sentence whether it is the conclusion, whether it makes a point, and whether that point is developed, and which side it is on. Only then apply the level rules, starting from the top level and working down.

A solution

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

import re
```

```
FOR_WORDS = ["benefit", "advantage", "helps"]
AGAINST_WORDS = ["risk", "drawback", "harm"]
answers = [
    ("Ada", "A benefit is that parcels reach the right teacher, because the robot
recognises them. "
            "A risk is that faces are biometric data, so the school must have a lawful
basis and keep the data secure. "
            "Another risk is bias, because a detector trained mostly on adults may fail for
younger students. "
            "On balance the school should use a PIN instead, because it gives the benefit
without storing faces."),
    ("Bolt", "One benefit is speed, so staff save time. Another benefit is accuracy,
because no parcel goes to the wrong room. "
             "There is a risk to privacy."),
    ("Cog", "Face recognition is a benefit. It is also a risk."),
    ("Dot", "Robots are interesting and schools are busy places."),
]

def sentences(answer):
    return re.split(r"(?<=[.!?])\s+", answer.strip())

def has_any(sentence, words):
    s = sentence.lower()
    return any(w in s for w in words)

def analyse(answer):
    points = developed = 0
    sides = set()
    conclusion = False
    for s in sentences(answer):
        if re.match(r"(on balance|overall|in conclusion)\b", s, re.I):
            conclusion = True
            continue
        is_for, is_against = has_any(s, FOR_WORDS), has_any(s, AGAINST_WORDS)
        if is_for or is_against:
            points += 1
            if re.search(r"\b(because|so)\b", s, re.I):
                developed += 1
                if is_for:
                    sides.add("for")
                if is_against:
                    sides.add("against")
    return points, developed, len(sides) == 2, conclusion

def level_and_mark(points, developed, both_sides, conclusion):
    if developed >= 3 and both_sides and conclusion:
        return 3, 7 + min(2, developed - 3)
    if developed >= 2:
        return 2, 4 + min(2, developed - 2)
    if points >= 1:
        return 1, min(3, points)
    return 0, 0

for name, answer in answers:
    points, developed, both, concl = analyse(answer)
    level, mark = level_and_mark(points, developed, both, concl)
    print(f"{name}: level {level}, {mark} marks ({points} points, {developed} developed)")
```

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