ANSWERS

F12.8 Technology, jobs and daily life

Technology and society · GCSE · OCR J277 1.6.1, AQA 8525 3.8, Edexcel 1CP2 5.2.2 · about 15 min

What this lesson is about

Automation and work, how daily life changed, and how automatable a job is.

- Work
- Daily life
- How automatable is a job?

Questions

5 marks in all

1. Which kind of work is automated first? [1 mark]

- ☐ **A** Routine and repetitive work
- ☐ **B** Work needing care and judgement
- ☐ **C** Work needing persuasion
- ☐ **D** Unpredictable work

> **Answer:** A. Repetitive tasks are the easiest to describe to a machine.

2. Which statement about automation is most accurate? [1 mark]

- ☐ **A** It removes some jobs, changes many and creates others
- ☐ **B** It removes all jobs
- ☐ **C** It creates jobs only
- ☐ **D** It has no effect on jobs

> **Answer:** A. The gains and losses rarely fall on the same people.

3. Give a drawback of remote work. [1 mark]

- ☐ **A** It can blur work and home and leave people isolated
- ☐ **B** It needs a commute
- ☐ **C** It shrinks the pool of applicants
- ☐ **D** It always lowers pay

> **Answer:** A. The benefits are real too: no commute, and wider access to jobs.

4. Online shopping grew quickly. Give a cost of that. [1 mark]

- ☐ **A** High street shops closing, plus delivery miles and packaging
- ☐ **B** Less choice for shoppers
- ☐ **C** Higher prices for everything
- ☐ **D** Slower delivery

> **Answer:** A. Most changes have both a benefit and a cost.

5. A job is 40% stacking (0.5 automatable) and 60% advice (0). What percentage is automatable? [1 mark]

> **Answer:** 20. 0.4 × 0.5 = 0.2, so 20%.

The task: how automatable?

For each job in `jobs`, work out its automatable share: the total of each task's share times how automatable it is. Print `<job>: <n>%` rounded to the nearest whole number, sorted from the most automatable to the least. Add `AT RISK` to any job of 60% or more. At the end print `most human task: <task>`, the task with the lowest automatable value across every job.

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

# job: [(task, share of the job, how automatable 0 to 1), ...]
jobs = {
    "shop assistant": [("stacking shelves", 0.4, 0.9), ("helping customers", 0.4, 0.2),
("counting stock", 0.2, 1.0)],
    "nurse": [("taking notes", 0.3, 0.7), ("caring for patients", 0.6, 0.05), ("ordering
supplies", 0.1, 0.9)],
    "warehouse picker": [("finding items", 0.6, 0.95), ("packing", 0.3, 0.8), ("checking
damage", 0.1, 0.4)],
}
```

The hint students can ask for: A job's score is the total of each task's share times how automatable that task is. Sort the jobs by that score, mark the ones at or above the threshold, and find the single task with the lowest automatable value anywhere.

A solution

```
from bugbot import *
connect()
jobs = {
    "shop assistant": [("stacking shelves", 0.4, 0.9), ("helping customers", 0.4, 0.2),
("counting stock", 0.2, 1.0)],
    "nurse": [("taking notes", 0.3, 0.7), ("caring for patients", 0.6, 0.05), ("ordering
supplies", 0.1, 0.9)],
    "warehouse picker": [("finding items", 0.6, 0.95), ("packing", 0.3, 0.8), ("checking
damage", 0.1, 0.4)],
}
scores = {}
for job, tasks in jobs.items():
    scores[job] = sum(share * auto for task, share, auto in tasks)
for job, score in sorted(scores.items(), key=lambda kv: kv[1], reverse=True):
    if score >= 0.6:
        print(f"{job}: {score * 100:.0f}% AT RISK")
    else:
        print(f"{job}: {score * 100:.0f}%")
every = [t for tasks in jobs.values() for t in tasks]
print("most human task:", min(every, key=lambda t: t[2])[0])
```

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