ANSWERS

4.5 Searching

Vision · Robot club · about 20 min

What this lesson is about

Spin until it is in view; find and approach as functions; list comprehensions.

- Nothing in view
- Spin until you see it
- Spin the right way
- Find, then approach

Questions

7 marks in all

1. What does `while not apriltags():` mean? [1 mark]

- ☐ **A** Keep looping while no tag is in view
- ☐ **B** Keep looping while a tag is in view
- ☐ **C** Loop once, then stop
- ☐ **D** Loop until the tag detector is switched off

> **Answer:** A. An empty list counts as false, so `not apriltags()` is true while the list is empty.

2. What does this program print? [1 mark]

```
tags = [[2, 60, 120, 35.0], [5, 210, 118, 60.0], [8, 300, 125, 90.0]]
mine = [t for t in tags if t[0] == 5]
print(mine)
print(len(mine))
```

> **Answer:**
>
> ```
> [[5, 210, 118, 60.0]]
> 1
> ```
>
> The list comprehension keeps only the tags whose id, `t[0]`, is 5. The result is still a list, with one tag in it.

3. The spin loop ends the moment a tag appears. Why do `stop()` and `wait(0.3)` come next? [1 mark]

- ☐ **A** The robot coasts, so the tag may slide to the edge of the view or out of it
- ☐ **B** The camera switches off when a tag is found
- ☐ **C** `apriltags()` only works when the robot is still
- ☐ **D** To give the list time to sort itself

> **Answer:** A. The tag entered the picture at the edge, and the robot keeps turning a little after the stop. Settle before you trust what you see.

4. What is a heuristic search? [1 mark]

- ☐ **A** A search that uses a clue, like where the thing was last seen
- ☐ **B** A search that always spins right
- ☐ **C** A search that gives up after one second
- ☐ **D** A search that uses the blob detector

> **Answer:** A. Without a clue, right is as good as left. With one, such as the last place you saw it, search that way first.

5. The camera sees 120 degrees. How many degrees of the full circle around the robot are out of view? [1 mark]

> **Answer:** 240. 360 minus 120 is 240. Most of the world is behind or beside you, which is why a robot has to search.

6. In `approach`, what happens if the tag goes out of view on the way in? [1 mark]

- ☐ **A** It turns slowly with `turn_right(30)` until the tag is back
- ☐ **B** It stops and the program ends
- ☐ **C** It drives forward at full speed
- ☐ **D** It goes back to the start

> **Answer:** A. Losing the tag is expected, so `approach` turns slowly and looks again instead of giving up.

7. Put the hide and seek program in order. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `print("found 5")`
___ `set_cv("apriltag")`
___ `find(5)`
___ `approach(5, 14)`

> **Answer:**
>
> ```
> `set_cv("apriltag")`
> `find(5)`
> `print("found 5")`
> `approach(5, 14)`
> ```
>
> Switch on the detector first, then search, report, and servo in. Every camera program is search and approach in a row.

The task: hide and seek

Find marker 5, print `found 5`, then drive up and stop about 14 cm in front of it.

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

# camera: the tag detector
set_cv("apriltag")
print(apriltags())
```

The hint students can ask for: Marker 5 is somewhere around you but not in view. Spin until the camera sees it, print `found 5`, then drive up and stop about 14 cm in front of it.

A solution

```
from bugbot import *
connect()
def bearing_of(cx):
    return (cx - 160) * 120 / 320      # pixels to degrees, 120 degree view across 320
pixels
def find(tag_id):
    # spin until the tag is in view, then square up to it
    while True:
        tags = [t for t in apriltags() if t[0] == tag_id]
        if tags:
            break
        turn_right(40)
        wait(0.1)
    stop()
    wait(0.3)

def approach(tag_id, stop_at):
    while True:
        tags = [t for t in apriltags() if t[0] == tag_id]
        if not tags:
            turn_right(30)
            wait(0.1)
            continue
        tag = tags[0]
        cx, dist = tag[1], tag[3]
        if dist <= stop_at:
            break
        rot = bearing_of(cx) * 3
        speed = max(20, min(70, (dist - stop_at) * 3))
        drive(speed, 0, rot)
        wait(0.1)
    stop()
set_cv('apriltag')
find(5)
print('found 5')
approach(5, 14)
```

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