PRESENTATION 15 · LEARNING WITH PYTHON

One of each

One iron, one silver, one gold, and never a second of the same: check what you carry before collecting.

TheBotCup 2026 · AiBotCup Simulator



RECAP · PRESENTATION 14

LoadedObjects counts everything

LoadedObjects < 3

collect while not full

But…

iron + iron + silver is also 3

It counts objects, not **which metal**. Today that matters.

PART 1

The mission: Session 15

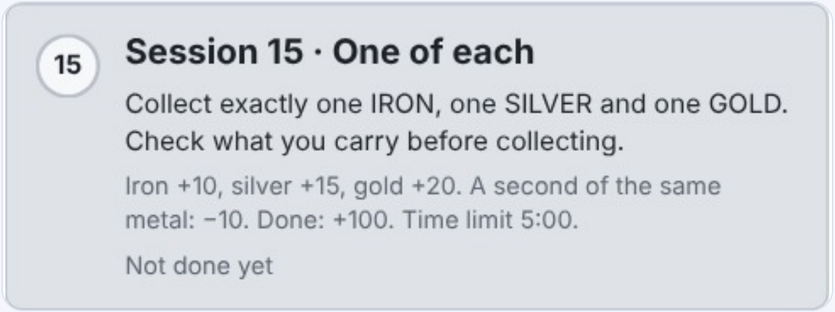
Exactly one of each metal

SESSION 15 · ONE OF EACH

One IRON, one SILVER, one GOLD

Iron **+10**, silver **+15**, gold **+20**. A second of the same metal: **−10**. Done: **+100**.

Time limit **5:00**. Code challenge: at most **45** lines.





THE FIELD

Plenty of every metal

START at the bottom centre.

Many irons, silvers and golds.

The robot will drive over metals it **must not** take.

PART 2

One counter per metal

LoadedIron, LoadedSilver, LoadedGold

THE IDEA · BUILT-IN VARIABLES

The robot counts each metal

LoadedIron

irons on board

LoadedSilver

silvers on board

LoadedGold

golds on board

Like LoadedObjects: nothing to create, they count only once a metal is **really on board**, and a deposit sets them back to 0.

HOW TO USE THEM

Each collect checks its own metal

Collect branches	Also test	Means
IronLeft, IronRight	LoadedIron < 1	no iron yet
SilverLeft, SilverRight	LoadedSilver < 1	no silver yet
GoldLeft, GoldRight	LoadedGold < 1	no gold yet

Once a metal is on board, its two branches **switch off**: the robot drives over that metal.

PART 3

Step by step in Python

Eight branches

STEP 1 · THE WHOLE PROGRAM

Six careful collects, two drivers

Branch	Condition	Also	Action
1 IronLeft	left iron	LoadedIron < 1	collect()
2 IronRight	right iron	LoadedIron < 1	collect()
3 SilverLeft	left silver	LoadedSilver < 1	collect()
4 SilverRight	right silver	LoadedSilver < 1	collect()
5 GoldLeft	left gold	LoadedGold < 1	collect()
6 GoldRight	right gold	LoadedGold < 1	collect()
7 Ahead	US_Front ≤ 20	—	−2 / 2 · Duration 700
8 Forward	the else	—	4.5 / 4.5

STEP 2 · LINES 3–10

The tools we know

```
3  def iron(r, g, b): return r <= 60 and g <= 60 and b <= 60
4  def silver(r, g, b): return 90 <= r <= 135 and 92 <= g <= 137 and 98 <= b <= 142
5  def gold(r, g, b): return r >= 195 and 135 <= g <= 200 and b <= 80
6
7  def collect():
8      WheelLeft = 0; WheelRight = 0
9      LED_1 = 2; LED_R = 255; LED_G = 0; LED_B = 0     # flashing red
10     Duration = 3200                                  # hold 3.2 seconds
```

iron(), silver(), gold(): the colour ranges, written once.

collect(): stop, flash red, hold 3.2 s.

STEP 3 · LINES 12–21

Iron and silver: only if we have none

```
12  def game1():
13      LED_1 = 0
14      if LoadedIron < 1 and iron(CSLeft_R, CSLeft_G, CSLeft_B):           # IronLeft
15          collect()
16      elif LoadedIron < 1 and iron(CSRight_R, CSRight_G, CSRight_B):      # IronRight
17          collect()
18      elif LoadedSilver < 1 and silver(CSLeft_R, CSLeft_G, CSLeft_B):     # SilverLeft
19          collect()
20      elif LoadedSilver < 1 and silver(CSRight_R, CSRight_G, CSRight_B):  # SilverRight
21          collect()
```

LoadedIron < 1 and iron(...) → collect(). No iron yet, and iron under the sensor.

The same with **LoadedSilver < 1** for silver, left and right.

STEP 4 · LINES 22–30

Gold, then the two drivers

```
22      elif LoadedGold < 1 and gold(CSLeft_R, CSLeft_G, CSLeft_B):          # GoldLeft
23          collect()
24      elif LoadedGold < 1 and gold(CSRight_R, CSRight_G, CSRight_B):       # GoldRight
25          collect()
26      elif US_Front <= 20:                                                  # Ahead
27          WheelLeft = -2; WheelRight = 2
28          Duration = 700
29      else:                                                                 # Forward
30          WheelLeft = 4.5; WheelRight = 4.5
```

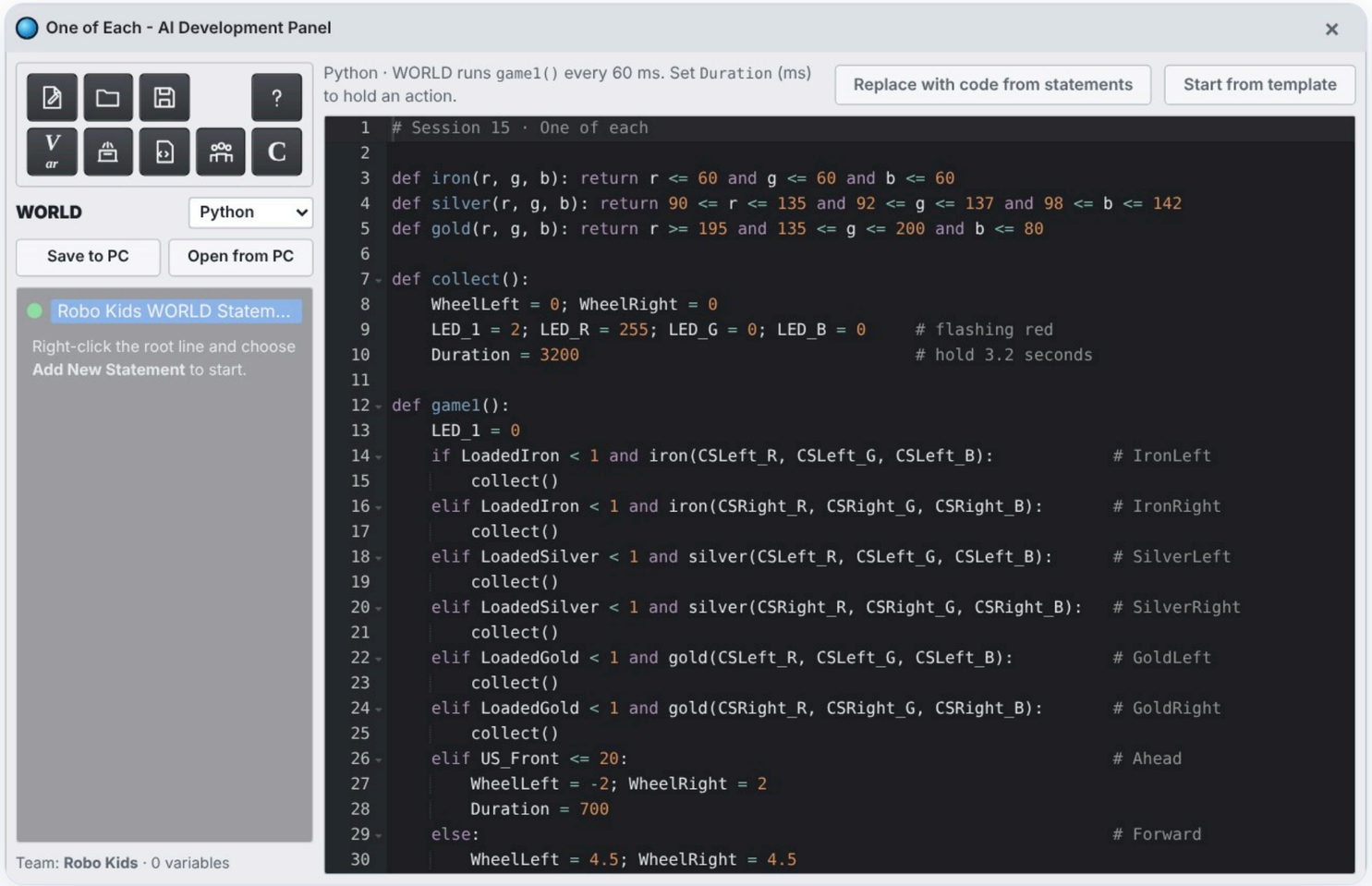
LoadedGold < 1 and gold(...) → collect().

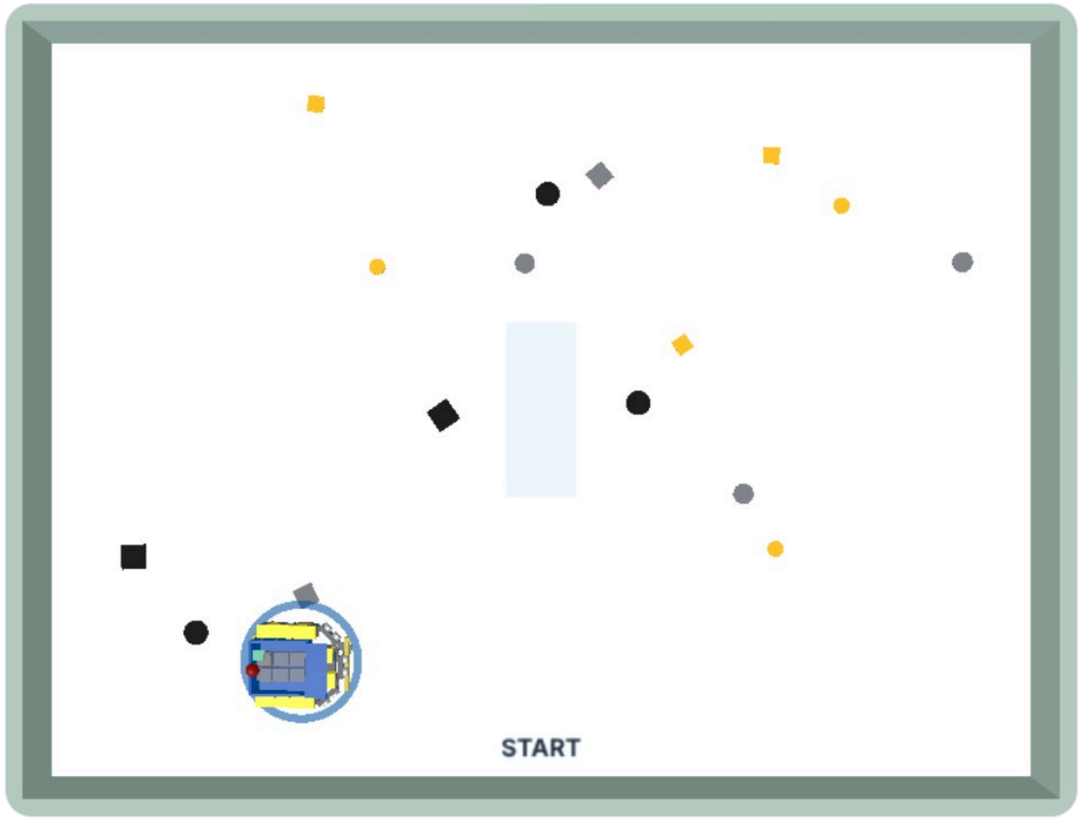
Ahead: −2 / 2 for 700 ms. **Forward**: the else.

STEP 5 · BUILD AND LOAD

The whole program, built

1. WORLD menu: **Python**.
2. Type the program, name it **One of Each**.
3. Press **Build**, then **OK**.
4. Press **Load** on the BLUE side.





Colour Sensors

Left R	109	(CSLeft_R)
G	113	(CSLeft_G)
B	116	(CSLeft_B)

left sees this colour

Right R	225	(CSRight_R)
G	232	(CSRight_G)
B	237	(CSRight_B)

right sees this colour

Robot

Wheels L **0.0** · R **0.0** · LED **2**

Load **1**/6 · Fe 1 Ag 0 Au 0

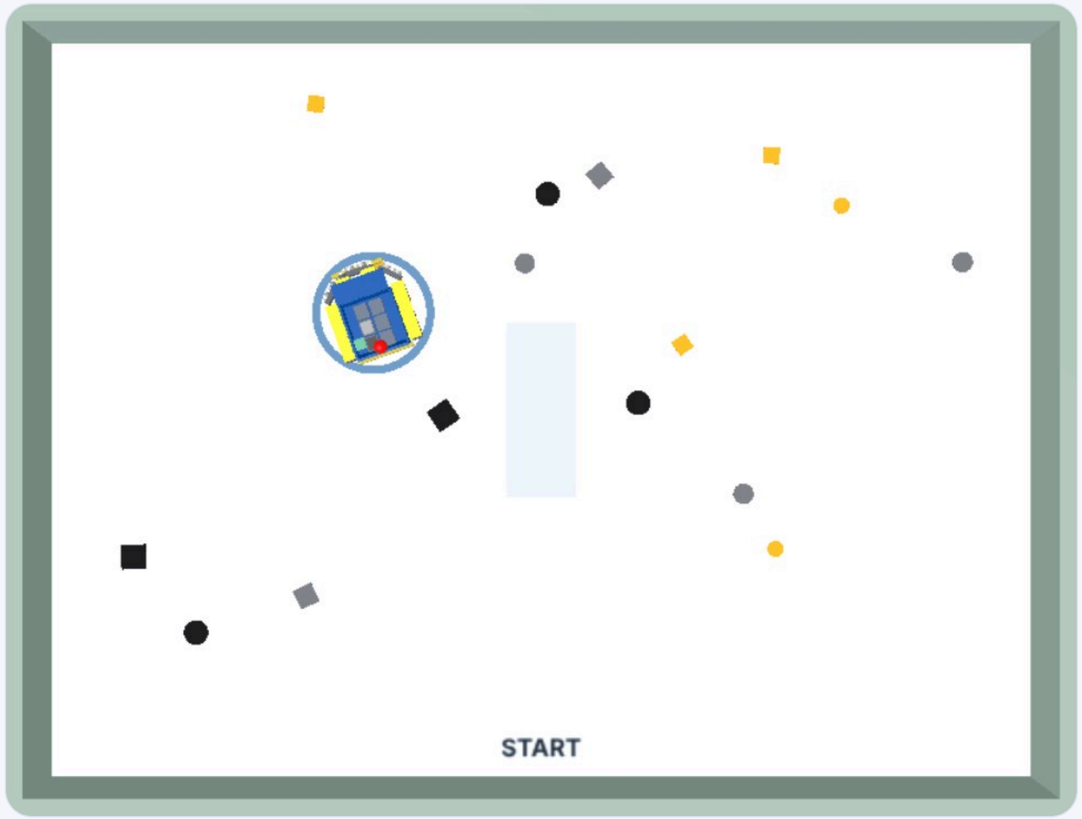
On **collecting 1.6 / 3 s**

STEP 6 · RUN · 15.0 S

Iron done, now silver

Load: **Fe 1**. Left sees silver, LoadedSilver is 0: **SilverLeft**.

Iron branches are now false.



Colour Sensors

Left R	225	(CSLeft_R)
G	233	(CSLeft_G)
B	240	(CSLeft_B)

left sees this colour

Right R	220	(CSRight_R)
G	171	(CSRight_G)
B	35	(CSRight_B)

right sees this colour

Robot

Wheels L **0.0** · R **0.0** · LED **2**

Load **2**/6 · Fe 1 Ag 1 Au 0

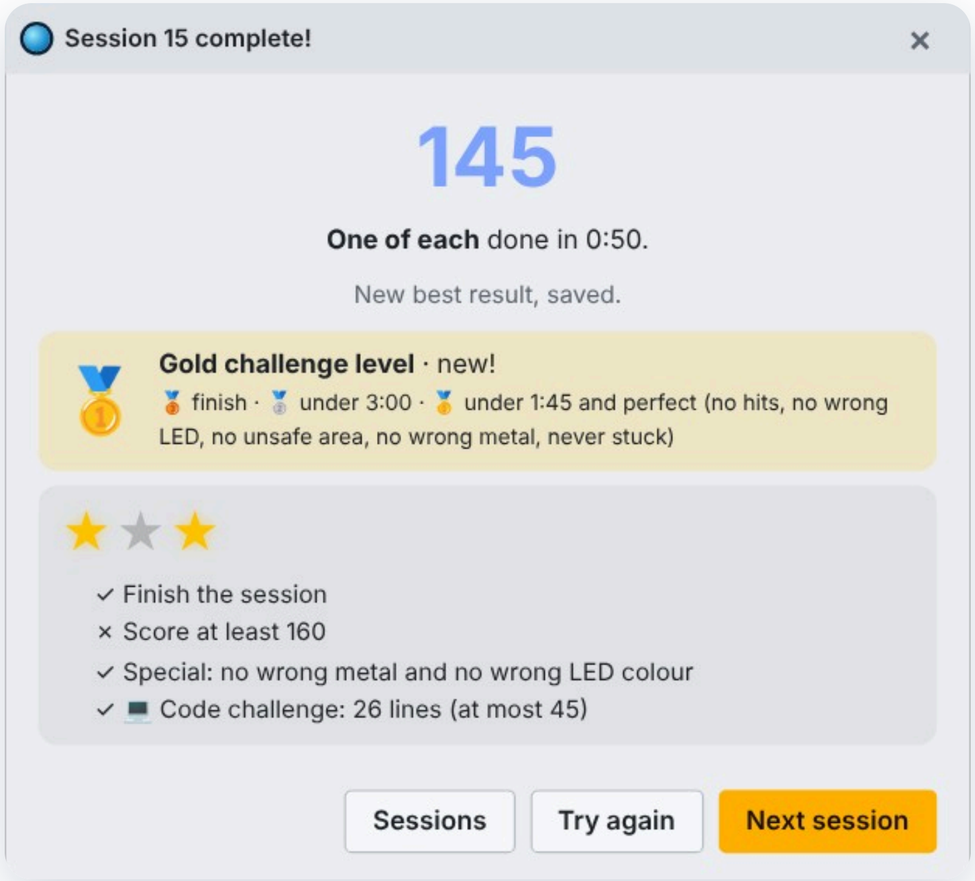
On **collecting 1.8 / 3 s**

STEP 7 · RUN · 49.1 S

The last one: gold

Load: **Fe 1 · Ag 1**. Right sees gold: **GoldRight**.

On the way it passed irons and silvers **without stopping**.



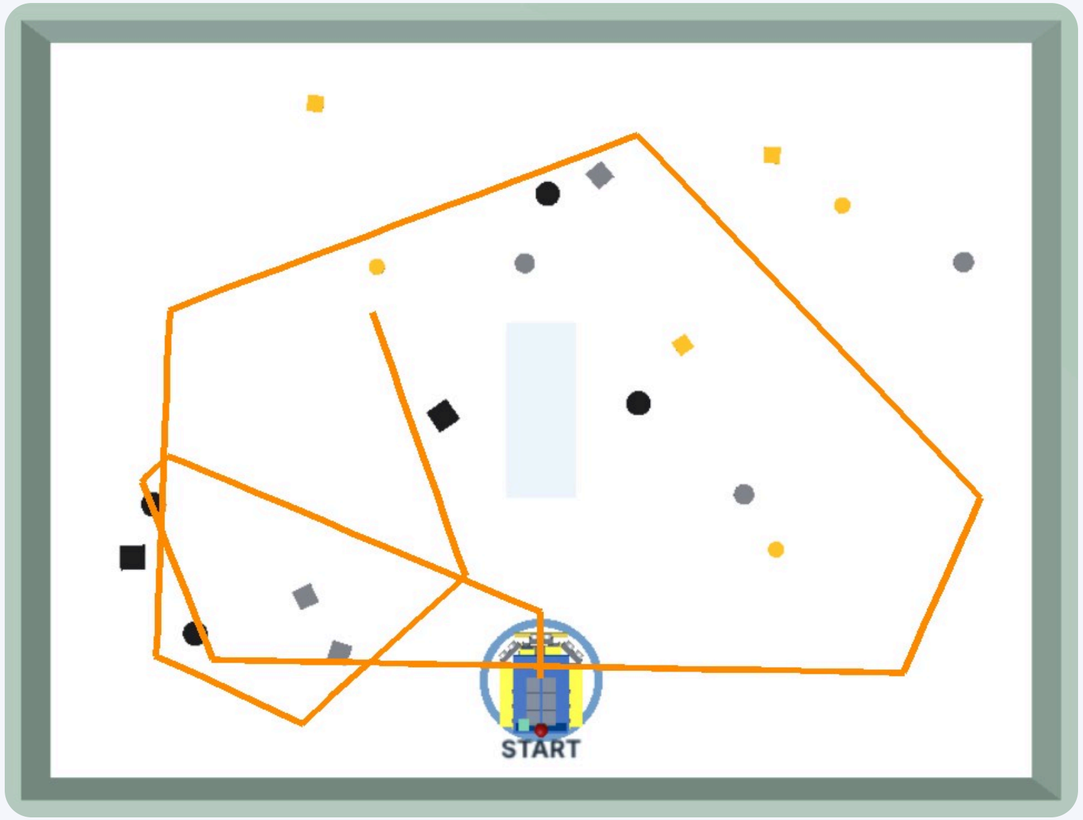
RESULT · TESTED

145 points in 0:50 · Gold

Score · 10 + 15 + 20 + 100, no −10.

Special · no wrong metal, no wrong LED. The “score 160” star was not reached.

Code challenge · at most 45 lines. We used 26.



THE PATH · RECORDED

Driving over, on purpose

The line crosses metals the robot **already had**.

Three counters, three switches: each metal turns its own branches **off**.

DID NOT WORK?

Check first, then tune carefully

What you see	What to do
A second iron, −10	Both iron branches need **LoadedIron < 1 and**
Never collects silver	Silver branches check **LoadedSilver**, not LoadedIron
Build error: unknown name	Spelling and capitals: **LoadedGold** exactly
Old program still on the robot	**Build** again, then **Load** on BLUE

NEXT TIME

Presentation 16 · Around the flags

A route around obstacles using **position and compass**.