



GAMING

DEMO FIELD GUIDE

WALKTHROUGH + COMPLETE SOLUTIONS

THE HAWKINS GATE PROTOCOL

Explore every room. Restore every system.
Recover seven fragments. Close the gate.

08 CONNECTED ROOMS

09 PYTHON CHALLENGES

07 GATE FRAGMENTS

SOLUTIONS INCLUDED

Your mission starts here.

A complete route through the game, including the actions between the coding tasks.

CLOSE THE GATE

Seven corrupted systems stand between you and the shutdown protocol. Walk through the facility, restore each machine with Python, and recover its fragment.

Explore

Find the note, activation device and console.

Repair

Use the required item, then solve the task.

Recover

Collect the fragment to unlock the next door.

Spoilers inside. Each room includes a visual walkthrough and full solution. Use the answers for a guided demo or after trying the task.

JUMP TO A SECTION

Get ready: controls and inventory	03
The complete fragment chain	04
01 The Four Signal Machines	05
02 The Containment Symbol Printer	07
03 The Identity Scanner	09
04 The Hawkins Sorting Chamber	11
05 The Upside Down Grid Generator	13
06 Creature Number Analysis	16
07 The Synchronized Memory Tapes	18
08 The Gate Fragment Compiler	20
All answers at a glance	24
Demo checklist and troubleshooting	25
Source and verification notes	27

Move. Interact. Use. Run.

Use the on-screen prompts and move close to the object you are trying to operate.

START THE GAME

Open arcacademy.games and launch **The Hawkins Gate Protocol**. Choose an avatar if prompted. Continue through the opening screen with **Let's GO!** and follow the mission's start prompt.

W A S D / arrows	Move around the room
E	Interact, collect or use a door
I	Open your inventory
Esc	Close a dismissible panel
Ctrl / Cmd + Enter	Run code in the terminal
Space	Jump; no required jump puzzle

UNTIMED IN THIS PACKAGE

All eight rooms are configured without a countdown. Focus on the complete sequence, not on speed.

USING AN ITEM IS A SEPARATE ACTION

- 1 Approach the target**
Stand beside the socket, reader, lock or power panel—not beside the terminal.
- 2 Open inventory**
Press **I** and select the named item. For example: **Signal Fragment**.
- 3 Choose Use**
Click the inventory item's **Use** action while you are close to the correct target.
- 4 Confirm the response**
Look for the accepted/connected message or powered console. Then close inventory and move to the terminal.

IN THE PYTHON WORKSPACE

Read the task, edit the available lines, then choose **Run Python**. Use hints progressively. Keep the full solutions as a reference when some starter lines are locked.

Seven fragments. One connected journey.

Rooms 2-7 use the previous fragment to activate a system. Keep every fragment for the final chamber.

ROOM	USE THIS ITEM	ON THIS DEVICE	COMPLETE / RECOVER	CARRY TOTAL
01	Spare Fuse	Wall power panel	Signal Fragment	1 fragment
02	Signal Fragment	Purple floor emitter	Shape Fragment	2 fragments
03	Shape Fragment	Specimen pad	Identity Fragment	3 fragments
04	Identity Fragment	Badge reader	Sorting Fragment	4 fragments
05	Sorting Fragment	Red ritual socket	Matrix Fragment	5 fragments
06	Matrix Fragment	Analysis dais	Creature Fragment	6 fragments
07	Creature Fragment	Golden archive lock	Memory Fragment	7 fragments
08	All seven fragments	Seven matching sockets	Submit final protocol 9774	Gate closed

THE RULE THAT UNLOCKS THE GAME

Code succeeds → fragment becomes collectible → collect the fragment → the door unlocks.

Using a fragment in Rooms 2-7 keeps it in inventory. Room 8 consumes each fragment when its matching socket accepts it.

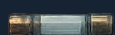
The Four Signal Machines

Variables, integer arithmetic and a four-digit protocol.



ROOM REFERENCE / NUMBERED MARKERS MATCH THE ACTIONS

USE TO ACTIVATE



Spare Fuse

COLLECT NEXT



Signal Fragment

PROTOCOL

3275

1 Collect the Spare Fuse

Walk to the red tool chest on the left. Press E to collect the fuse.

2 Restore the wall panel

Stand beside the panel above the chest. Press I, select Spare Fuse, then Use.

3 Repair the signal desk

Open the powered desk with E. Complete the transformations and run Python.

4 Collect the Signal Fragment

After success, approach the green crystal on the right and press E.

5 Leave through the door

Collection unlocks the right-hand containment door. Approach it, press E, then continue.

BEFORE YOU LEAVE

Signal Fragment in inventory; containment door unlocked. The fuse is consumed by the power panel.

Repair the code. Check the result.

The Four Signal Machines

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 a = 9
02 a = a - 4
03 a = a * 2
04 a = a // 3
05
06 b = 5
07 b = b + 7
08 b = b % 10
09
10 c = 8
11 c = c * 3
12 c = c - 17
13
14 d = 6
15 d = d + 5
16 d = d // 2
17
18 submit_code(f"{a}{b}{c}{d}")
```

EXPECTED PROTOCOL

3275

Machine A

$9 - 4 = 5$; $5 \times 2 = 10$; $10 // 3 = 3$.

Machine B

$5 + 7 = 12$; $12 \% 10 = 2$.

Machine C

$8 \times 3 = 24$; $24 - 17 = 7$.

Machine D

$6 + 5 = 11$; $11 // 2 = 5$.

KEY IDEA

Use integer division `//`, not `/`. The four values are joined as digits, not added.

After success: Machines A-D become tuned. The protocol display reads **3275**, and the Signal Fragment becomes collectible. Close the console, collect the fragment, then use the exit.

The Containment Symbol Printer

Lists, a for loop, polygon side counts and turning angles.



ROOM REFERENCE / NUMBERED MARKERS MATCH THE ACTIONS



USE TO ACTIVATE
Signal Fragment



COLLECT NEXT
Shape Fragment

PROTOCOL
4358

- 1 **Calibrate the emitter**
Use the **Signal Fragment** on the round purple floor emitter at the front-left: I → **Signal Fragment** → Use.
- 2 **Power the four projectors**
Open the projection console at the lower-right. Calculate the side counts and angles, then run Python.
- 3 **Collect the Shape Fragment**
Success stabilizes all four projectors. Collect the red crystal on the right with **E**.
- 4 **Take the containment door**
The door unlocks after collection. Press **E** at the right-hand exit and continue.

BEFORE YOU LEAVE

Keep both Signal and Shape Fragments. Using the Signal Fragment here does not consume it.

Calibrate all four projectors.

Full solution: side counts are transmitted; turning angles are checked too.

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 side_counts = [
02     12 // 3,
03     14 % 6 + 1,
04     2 * 3 - 1,
05     18 // 3 + 2,
06 ]
07
08 labels = ["A", "B", "C", "D"]
09 angles = []
10
11 for index in range(len(side_counts)):
12     sides = side_counts[index]
13     angle = 360 // sides
14     angles.append(angle)
15     set_resource(labels[index], sides)
```

PROJECTORS A → D

4 / 3 / 5 / 8

PROJECTOR	SIDES	ANGLE
A	4	90°
B	3	120°
C	5	72°
D	8	45°

BOTH LISTS MATTER

This task checks both **side_counts** and **angles**. It does not require **submit_code("4358")**.

Why it works: the expressions yield [4, 3, 5, 8]. For each projector, **360 // sides** yields its turning angle. **set_resource()** transmits the side count to the named projector.

Then: collect the Shape Fragment to open the exit.

The Identity Scanner

Convert text to integers before applying arithmetic.



ROOM REFERENCE / NUMBERED MARKERS MATCH THE ACTIONS



USE TO ACTIVATE

Shape Fragment



COLLECT NEXT

Identity Fragment

PROTOCOL

7341

1 Wake the checkpoint

Use the Shape Fragment on the specimen pad at the front-left: I → Shape Fragment → Use.

2 Convert the incoming data

Open the central checkpoint desk. Convert all four text values with `int()`, calculate and run Python.

3 Collect the Identity Fragment

After the badge is produced, take the card on the right-hand pedestal with E.

4 Proceed to sorting

Collection unlocks the containment door on the right. Use the door and continue.

BEFORE YOU LEAVE

Carry Signal, Shape and Identity Fragments. The new card is the key to the next room's badge reader.

Repair the code. Check the result.

The Identity Scanner

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 age = "12"
02 sector = "34"
03 power = "7"
04 signal = "81"
05
06 age = int(age)
07 sector = int(sector)
08 power = int(power)
09 signal = int(signal)
10
11 a = (age + 5) % 10
12 b = sector // 10
13 c = (power * 2) % 10
14 d = signal % 10
15
16 submit_code(f"{a}{b}{c}{d}")
```

EXPECTED PROTOCOL

7341

A / age

$\text{int}("12") \rightarrow 12; (12 + 5) \% 10 = 7.$

B / sector

$\text{int}("34") \rightarrow 34; 34 // 10 = 3.$

C / power

$\text{int}("7") \rightarrow 7; (7 \times 2) \% 10 = 4.$

D / signal

$\text{int}("81") \rightarrow 81; 81 \% 10 = 1.$

KEY IDEA

The quotes mean text. Convert every received value with `int()` before doing arithmetic.

After success: The badge printer receives **7341**, and the Identity Fragment becomes collectible. Close the console, collect the fragment, then use the exit.

The Hawkins Sorting Chamber

Functions, comparisons, temporary variables and ordered swaps.



ROOM REFERENCE / NUMBERED MARKERS MATCH THE ACTIONS



USE TO ACTIVATE

Identity Fragment



COLLECT NEXT

Sorting Fragment

PROTOCOL

5462

1 Authorize the badge reader

Use the Identity Fragment on the small reader in the lower-left: I → Identity Fragment → Use.

2 Repair the sorting function

Open the console on the central platform. Complete all three compare-and-swap steps, then run Python.

3 Collect the Sorting Fragment

Success opens the extraction chute and activates the specimen reward. Collect it on the right with E.

4 Leave for the matrix room

The containment door unlocks when the fragment is collected. Use the right-hand door.

BEFORE YOU LEAVE

Four fragments collected. Keep the Identity Fragment as well as the newly recovered Sorting Fragment.

Return the middle value.

Order three values with comparisons and swaps. Do not replace the exercise with sorted().

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 def middle_value(a, b, c):
02     if a > b:
03         temp = a
04         a = b
05         b = temp
06
07     if b > c:
08         temp = b
09         b = c
10         c = temp
11
12     if a > b:
13         temp = a
14         a = b
15         b = temp
16
17     return b
18 digits = [
19     middle_value(8, 2, 5),
20     middle_value(1, 9, 4),
21     middle_value(7, 3, 6),
22     middle_value(2, 8, 1),
23 ]
24
25 submit_code("".join(str(value) for value in digits))
```

EXPECTED PROTOCOL

5462

INPUT GROUP	ORDERED	MIDDLE
8, 2, 5	2, 5, 8	5
1, 9, 4	1, 4, 9	4
7, 3, 6	3, 6, 7	6
2, 8, 1	1, 2, 8	2

WHY THREE COMPARISONS?

The last **if a > b** is necessary. Swapping b and c can disturb the first pair, so compare a and b once more.

After success: the chute opens and the Sorting Fragment becomes collectible. Collect it on the right to unlock the door.

The Upside Down Grid Generator

Two connected tasks: build a 4 × 4 framework, then decode a matrix.



ROOM REFERENCE / NUMBERED MARKERS MATCH THE ACTIONS



USE TO ACTIVATE

Sorting Fragment



COLLECT NEXT

Matrix Fragment

PROTOCOL

7145

- 1 **Open the matrix lock**
Use the **Sorting Fragment** on the red ritual socket at the left.
- 2 **Build at the left console**
Run Task 5A to place all **16** cells. This activates the right-hand decoder.
- 3 **Decode at the right console**
Move to the second console. Run Task 5B to generate the values and extract the marked cells.
- 4 **Collect the Matrix Fragment**
After the decoder succeeds, collect the green matrix tile on the raised rear pedestal.
- 5 **Use the containment door**
Collection—not merely building the grid—unlocks the door on the right.

CHECK

Both terminals completed, Matrix Fragment collected, five fragments in inventory.

Task 5A: build the framework.

Use the left console first. Success activates the decoder on the right.

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 for row in range(4):
02     for column in range(4):
03         place_block(column, row)
```

ONE CALL FOR EACH CELL

`place_block(column, row)` receives the column first. Both loops visit 0, 1, 2 and 3, so the program places $4 \times 4 = 16$ cells.

WHAT CHANGES IN THE ROOM

The floor grid becomes a constructed framework. The right-hand extraction console powers on, and the marked-cell clue is revealed.

You are not finished with Room 5. Close the build console, walk to the right console, and continue with Task 5B on the next page.

EVERY CELL IS REQUIRED

	0	1	2	3
0	0,0	1,0	2,0	3,0
1	0,1	1,1	2,1	3,1
2	0,2	1,2	2,2	3,2
3	0,3	1,3	2,3	3,3

Columns →

Rows increase downward.

This diagram uses zero-based coordinates.

BUILD TARGET

16 / 16

Task 5B: generate and decode.

Use the right console after the framework is complete.

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 grid = []
02
03 for row in range(1, 5):
04     line = []
05
06     for column in range(1, 5):
07         value = (row * column + row + column) % 10
08         line.append(value)
09
10     grid.append(line)
11
12 digits = [
13     grid[0][2],
14     grid[2][1],
15     grid[3][3],
16     grid[2][2],
17 ]
18
19 submit_code("".join(str(value) for value in digits))
```

EXPECTED PROTOCOL

7145

GENERATED MATRIX

	0	1	2	3
0	3	5	7	9
1	5	8	1	4
2	7	1	5	9
3	9	4	9	4

Read in this order:

[0][2] = 7 / [2][1] = 1

[3][3] = 4 / [2][2] = 5

The highlighted cells are a lookup, not a left-to-right reading order.

Then: collect the Matrix Fragment from the raised rear pedestal. Collection unlocks the right-hand door. Generator values are 1-4; list indices are 0-3.

Creature Number Analysis

A while loop, digit extraction, parity, accumulation and reversal.



ROOM REFERENCE / NUMBERED MARKERS MATCH THE ACTIONS



USE TO ACTIVATE

Matrix Fragment



COLLECT NEXT

Creature Fragment

PROTOCOL

8346

1 Activate the analysis dais

Use the **Matrix Fragment** on the round green analysis dais at the front-left.

2 Analyze every digit

Open the console below the tank. Process the number **6843** with a while loop and run Python.

3 Collect the Creature Fragment

After all four gauges match, collect the teal specimen fragment on the right with **E**.

4 Enter the archive

Collection unlocks the right-hand door. Use it and continue.

BEFORE YOU LEAVE

Six fragments collected. The Creature Fragment will open the archive lock in Room 7.

Analyze the digits of 6843.

The program sends four instrument readings. No `submit_code()` call is needed here.

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 signal = 6843
02
03 even_sum = 0
04 odd_sum = 0
05 digit_count = 0
06 reversed_signal = 0
07
08 while signal > 0:
09     digit = signal % 10
10
11     if digit % 2 == 0:
12         even_sum += digit
13     else:
14         odd_sum += digit
15
16     reversed_signal = reversed_signal * 10 + digit
17     digit_count += 1
18     signal //= 10
19
20 set_resource("EVEN", even_sum % 10)
21 set_resource("ODD", odd_sum % 10)
22 set_resource("COUNT", digit_count)
23 set_resource("REVERSE", reversed_signal % 10)
```

EVEN / ODD / COUNT / REVERSE

8 / 3 / 4 / 6**EVEN → 8**

$6 + 8 + 4 = 18$; keep the last digit: $18 \% 10 = 8$.

ODD → 3

Only 3 is odd. Its sum is 3.

COUNT → 4

Four digits were processed.

REVERSE → 6

The full reversed number is 3486; $3486 \% 10 = 6$.

KEEP THE LOOP MOVING

`signal //= 10` removes the last digit each time. Without it, the loop never ends.

After success: protocol 8346 is recorded. Collect the Creature Fragment from the right-hand containment pedestal to unlock the exit.

The Synchronized Memory Tapes

Parallel lists, pop, append, reverse and lookup by index.



ROOM REFERENCE / NUMBERED MARKERS MATCH THE ACTIONS



USE TO ACTIVATE

Creature Fragment



COLLECT NEXT

Memory Fragment

PROTOCOL

7647

- 1 **Open the golden lock**
Use the **Creature Fragment** on the golden archive lock in the lower-left.
- 2 **Repair both tapes together**
Open the reading desk. Remove, append and reverse both lists in sync, then run Python.
- 3 **Collect the Memory Fragment**
After the archive is restored, collect the orange-red crystal on the right with **E**.
- 4 **Go to the final chamber**
Collection opens the door. Check that all **seven** fragments are in inventory before continuing.

BEFORE YOU LEAVE

Signal, Shape, Identity, Sorting, Matrix, Creature and Memory Fragments all present.

Keep the two tapes synchronized.

Apply each mutation to both lists, preserving the relationship between a signal and its level.

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 signals = ["bat", "vine", "gate", "shadow", "spore"]
02 levels = [4, 7, 2, 9, 5]
03
04 signals.pop(2)
05 levels.pop(2)
06
07 signals.append("echo")
08 levels.append(6)
09
10 signals.reverse()
11 levels.reverse()
12
13 first_digit = levels[signals.index("vine")]
14 second_digit = levels[signals.index("echo")]
15 third_digit = levels[signals.index("bat")]
16 fourth_digit = levels[len(signals) - 2]
17
18 submit_code(
19     f"{first_digit}{second_digit}{third_digit}{fourth_digit}"
20 )
```

EXPECTED PROTOCOL

7647

INDEX	SIGNAL	LEVEL
0	echo	6
1	spore	5
2	shadow	9
3	vine	7
4	bat	4

Lookup order:

vine → 7 / echo → 6

bat → 4 / index 3 → 7

Watch the pairing: pop index 2 from both lists, append echo and 6, then reverse both lists.

Then: collect the Memory Fragment. Check all seven fragments in inventory before using the door to the Gate Chamber.

The Gate Fragment Compiler

Insert the seven physical fragments before attempting the final coding task.



SEVEN NAMED SOCKETS / FINAL CONSOLE / GATE



Signal



Shape



Identity



Sorting



Matrix



Creature



Memory

- 1 Insert every fragment**
 Use each inventory fragment at the socket with the same name. Repeat **I → select fragment → Use** seven times. The next page identifies every socket.
- 2 Activate the compiler altar**
 When all seven sockets are filled, the front-center console activates. Open it with **E** and repair the five bugs.
- 3 Run the shutdown protocol**
 The repaired program submits **9774**. Read the acceptance message and continue through the shutdown sequence to the completion screen.

FINAL-ROOM EXCEPTION

No eighth fragment is collected. A correct program closes the gate and completes the game.

Match the name. Fill all seven sockets.

Suggested left-to-right route. The authored logic requires all seven matches, not one special insertion order.



1

2

3

4

5

6

7



SIGNAL

FROM ROOM 01



SHAPE

FROM ROOM 02



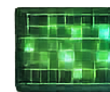
IDENTITY

FROM ROOM 03



SORTING

FROM ROOM 04



MATRIX

FROM ROOM 05



CREATURE

FROM ROOM 06



MEMORY

FROM ROOM 07



CONFIRM EACH INSERTION

The game reports “[Name] Fragment connected.” An accepted fragment is removed from inventory. When all seven are connected, the compiler powers on. The round central fixture is **not an eighth named socket**.

Repair the five compiler bugs.

Complete reference program. The seven room codes are already the inputs to this final task.

PYTHON / COMPLETE REFERENCE SOLUTION

RUN IN ARC

```
01 room_codes = [  
02     3275,  
03     4358,  
04     7341,  
05     5462,  
06     7145,  
07     8346,  
08     7647  
09 ]  
10  
11 fragments = []  
12  
13 for code in room_codes:  
14     middle = (code // 100) % 10  
15     last = code % 10  
16  
17     fragment = (middle * 2 + last) % 10  
18     fragments.append(fragment)  
19  
20 order = [6, 4, 2, 1]  
21 final_code = ""  
22  
23 for i in range(len(order)):  
24     final_code += str(fragments[order[i]])  
25  
26 submit_code(final_code)
```

FINAL SHUTDOWN PROTOCOL

9774

1 Integer division

Use `code // 100`, not `code / 100`.

2 Last digit

Use `code % 10`, not `code // 10`.

3 One-digit fragment

Apply `% 10` to the fragment formula.

4 Include index zero

Use `range(len(order))`, not a range starting at 1.

5 Concatenate text

Convert each selected fragment with `str()` before adding it to `final_code`.

Next page: the complete calculation behind the final answer.

Why the shutdown code is 9774.

A complete trace from the seven room protocols to the four-character answer.

$$\text{fragment} = (\text{hundreds_digit} \times 2 + \text{last_digit}) \% 10$$

INDEX	ROOM FRAGMENT	ROOM CODE	HUNDREDS	LAST	RESULT
0	Signal	3275	2	5	9
1	Shape	4358	3	8	4
2	Identity	7341	3	1	7
3	Sorting	5462	4	2	0
4	Matrix	7145	1	5	7
5	Creature	8346	3	6	2
6	Memory	7647	6	7	9

SELECT

order =
[6, 4, 2, 1]

[6] 9

[4] 7

[2] 7

[1] 4

SHUTDOWN ACCEPTED

9774 is submitted as text. The gate enters its closing state, the acceptance message appears, and the game moves to **The Hawkins Gate Is Closed** completion sequence.

All answers at a glance.

These are result checks. Follow the full code on the solution pages rather than bypassing the learning tasks.

ROOM	TASK	EXPECTED RESULT	NEXT OUTCOME	PAGE
01	Signal machines	3275	Signal Fragment	06
02	Polygon projectors	Sides 4, 3, 5, 8; angles 90, 120, 72, 45	Shape Fragment	08
03	Identity conversion	7341	Identity Fragment	10
04	Middle-value sorting	5462	Sorting Fragment	12
05A	Build matrix	16 cells; columns and rows 0-3	Activates second console	14
05B	Decode matrix	7145	Matrix Fragment	15
06	Creature analysis	EVEN 8 / ODD 3 / COUNT 4 / REVERSE 6	Creature Fragment	17
07	Memory tapes	7647	Memory Fragment	19
08	Fragment compiler	9774	Closes the gate	22

DO NOT SKIP THE PHYSICAL ACTIONS

First activate the room with the required inventory item. After coding, **collect the room's fragment to unlock its door**. In Room 5, complete both consoles. In Room 8, insert all seven fragments before opening the compiler.

Run the whole demo—not just the answers.

Suggested presentation sequence. This is a facilitation plan, not a timed requirement.

BEFORE THE AUDIENCE ARRIVES

Start at Room 1

Use a fresh demo run, not a resumed late-room state.

Check the first item handoff

Collect the Spare Fuse and use it on the wall panel.

Check Python once

Open a console, allow Python to load, and run the actual solution.

Check the experience

Use landscape, readable zoom and clear audio. Keep this guide on a second window or screen.

Rehearse the final sockets

Confirm the seven inventory-use targets work in the build you will present.

THE DEMONSTRATION STORY

- 1 **Room 1: show the complete loop**
Collect → use → code → visible repair → collect → door. Make the separation between solving and collecting explicit.
- 2 **Rooms 2-4: make the code visible**
Point to projectors, scanner inputs and specimen groups. Explain how Python is repairing the room, not answering an isolated worksheet.
- 3 **Room 5: show a dependency**
The first program builds the framework and unlocks the second console. Then the matrix values reveal the next fragment.
- 4 **Rooms 6-7: demonstrate reasoning**
Trace one digit through the while loop; then show why both memory lists must be edited together.
- 5 **Room 8: bring everything together**
Show all seven fragments, insert each one, repair the five bugs, and let the gate-closing completion sequence finish.

Stuck? Check the missing action.

A correct code result is only one part of a room's progression.

01 The console is dark or locked.

The activation step is still missing. Use the required incoming item on the socket, badge reader, lock or power panel first.

02 I pressed E on a socket and nothing changed.

E inspects or interacts with the object. To insert an item, stand beside the target, open inventory with I, select the matching item and choose Use.

03 My code passed, but the exit is still locked.

Close the console and collect the newly enabled fragment. Its collection event unlocks the door. Then walk to the exit and interact.

04 The Room 5 decoder is unavailable.

Complete Task 5A at the left console. Building all 16 cells activates the right-hand console. Task 5B is a separate program.

05 Room 8 still will not enable the compiler.

Confirm all seven named sockets accepted the corresponding fragments. Filling only one socket or merely carrying the seven items is not enough.

06 The program errors or never finishes.

Check indentation, spelling, quotes and the highlighted operators. Room 6 must shrink signal inside the while loop. A Python error is not the same as a wrong numeric answer.

Recovery tip: close any open dialog before moving. Use the room's white field-note paper or floating blue guide for direction. Reset replaces the starter code; it is not a full room restart.

Built from the supplied game.

Gameplay instructions follow the exported room configuration, challenges and action flows.

SOURCE BASIS

Game package

The-Hawkins-Gate-Protocol---Stranger-Things.arc-game(1).zip
Exported 9 September 2026. Room names, item targets, inventory behavior, terminal outcomes and door transitions were taken from **content/game.json** and its corresponding task/flow records.

Answer document

Game_Answers_Hawkings_gate.docx. All nine supplied solution programs were cross-checked with the package's task solutions.

Visual material

All eight supplied room images are included. Numbered markers and diagrams were added for this guide. Fragment art and the cover environment come from the game package.

ARC identity

The original ARC horizontal logo is reproduced from the existing ARC brand asset embedded in an earlier ARC template. Its recorded source is arcacademy.co/arc_logo/logos/arc-logo-horizontal.svg. The gaming-site logo could not be downloaded directly.

LOCAL PYTHON CHECKS PASSED

9 / 9

Each reference solution was executed in a local Python capture harness. Captured codes, resource values and grid coordinates were compared with the package's expected values; the projector assertions were also executed.

VERIFICATION BOUNDARY

This verifies the reference code and traces the authored progression. It is **not** a fresh end-to-end browser playtest or a guarantee that every live media asset and interaction target works.

Before a live demonstration, rehearse a fresh run in the actual published build—especially the item-use targets and the seven final sockets.