Extra Activity 1: Hoshruba

Fakhir Shaheen · Sections A and B

***Not graded**: no marks, no deadline, nothing to hand in*

Introduction

Hoshruba is the enchanted realm of the *Tilism-e-Hoshruba*, an epic printed in Lucknow between 1883 and 1893: forty-six volumes, the longest fantasy ever written. Its hero is not a warrior. He is Umro Ayyar, a trickster whose bag holds anything and whose wits beat every sorcerer. A *tilism* is a magic world built by wizards, and it can be broken only by finding and reading its Tablet. Umro has sent you, his newest apprentice, into a small tilism with a lamp and a riddle. **Your goal is to break the tilism**: get past the jinn, find the key to the wizard's door, reach the Tablet, and read it by lamplight. In as few turns as you can, because the game keeps score.

You will write a *text adventure*, the oldest kind of computer game. Will Crowther wrote the first one in 1976 by mapping a real cave into a program. There are no graphics. The game describes a place, you type a command, the program changes a few variables and describes the place again. That loop is the whole activity.

What you will practise

- A `while` loop that runs the whole game, and `for` loops that walk along a string (Lecture 7).
- `if`/`elif`/`else` chains that decide what a command does (Lecture 5).
- Strings as data: slicing, `in`, joining, and reading a map out of a string (Lecture 3).
- Boolean flags that remember what has already happened (Lecture 4).
- Checking your own program with `check.py`, as in the labs.

An extra activity

EA1 is an *extra activity*: it carries no marks, has no deadline, and nothing is handed in. It is practice, so it helps only if you write the code yourself. Code from a friend or from an AI tool gets the game working and teaches you nothing.

Getting Started

1. Download the EA1 starter kit from the course calendar and unzip it: `ea1.py` and `check.py`. **Keep both files in the same folder.** Open `ea1.py` in Thonny and read it. The top part is the `WORLD` block: the map of the tilism, as three strings and nine numbers. Below that, every part has a comment that tells you what to write. Part 1 is already started for you.
2. Open `check.py` and press F5, as in the labs. It needs no internet and never changes `ea1.py`, so run it after every part. It checks three of the rules below, then plays games against your program, part by part, and prints PASS or FAIL for each: the six games that the *Testing* notes of this handout name, and more that try the edge cases and other tilisms. Under the first game that fails it shows what went wrong: the command it had just typed, the line that was expected, and the line your program printed. Every

failed game is written out in full, as it would look on the screen, in `run_game_test_results.txt` beside `check.py`.

To play your own game, press F5 on `ea1.py` and type commands in the Shell.

Rule 1. Print every line exactly as shown. `check.py` ignores spaces at the end of a line. Everything else must match: capitals, full stops, quotation marks, and the number of lines, so print no blank lines.

Rule 2. Commands are lowercase words. A command the game does not know gets the answer `I don't understand that.` and costs nothing.

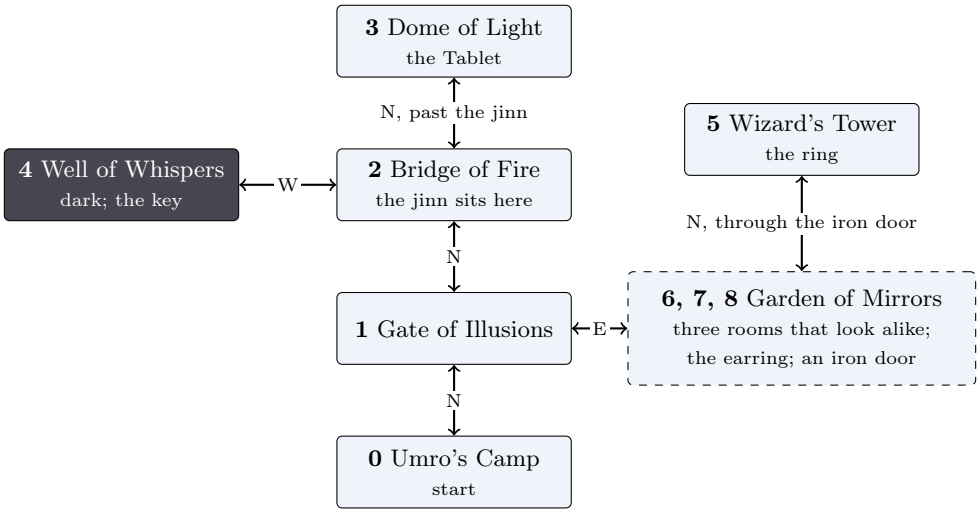
Rule 3. Use only what we did in Lectures 1 to 8: variables, `int()` and `str()`, arithmetic, strings with `+`, `*`, `[ ]`, `[ : ]`, `len()` and `in`, f-strings, `print()`, `input()`, comparisons, `and`/`or`/`not`, `if`/`elif`/`else`, `while` (with `break`), and `for` over a `range` or a string. Do NOT use `def`, lists, `import`, or anything with a dot like `.lower()`. Every block of code must start with a comment, as in A0. `check.py` checks both.

Rule 4. Do not edit the `WORLD` block. Your game must work in any tilism the block could describe: other rooms, other exits, and the jinn, the door and the well in other places. So your code must read `NAMES`, `DESCS`, `EXITS` and the nine constants, and never remember on its own that the jinn is in room 2. `check.py` checks both: one row says whether the `WORLD` block is untouched, and its last five games swap other tilisms into your file.

Tip. Part 1 needs only Lectures 3 to 5. Parts 2 to 8 need Lecture 7, on loops. The whole thing takes about ten hours, so work one part per sitting.

The Tilism

You start at Umro's camp, outside the Gate of Illusions. Beyond the gate, a jinn of fire sits on a bridge and will not let you cross to the Dome of Light, where the Tablet lies, unless you give it a gift. The gift it wants is a ring of red gold, locked in a wizard's tower at the far end of a garden of mirrors. The key to the tower lies at the bottom of the Well of Whispers, where it is always dark. Your lamp holds ten measures of oil, or eight if Umro's riddle beats you, and it burns one measure every turn it is lit. The letters of the Tablet glow only in lamplight, so keep some oil for the end.



How the tilism is stored

The `WORLD` block at the top of `ea1.py` is the map. Each string is written one room per line, with the room's number in a comment beside it, and the pieces joined with `+` inside a pair of brackets; Python reads the whole thing as one string. What it holds:

- `NAMES` holds the nine room names, separated by `|`. Room 0 is Umro's Camp, room 1 the Gate of Illusions, and so on. The name of room number `room` is the `room`-th field of `NAMES`, counting from 0. The starter shows how to pick out one field with a `for` loop.
- `DESCS` holds the nine descriptions in the same way.
- `EXITS` holds four characters for every room, in the order north, east, south, west: the number of the room that way, or `-` for no way. Room 1's four characters are `EXITS[4:8]`, which is `260-`: north leads to room 2, east to room 6, south to room 0, and there is no way west. In general, with direction numbers north 0, east 1, south 2 and west 3, the exit from room `r` in direction `d` is the character `EXITS[r * 4 + d]`.
- `KEY_ROOM`, `RING_ROOM` and `EARRING_ROOM` say where the loose items lie at the start. `DARK_ROOM` is the well. `JINN_ROOM` and `JINN_DIR` say where the jinn sits and which way it blocks. `DOOR_ROOM` and `DOOR_DIR` do the same for the iron door. `TABLET_ROOM` is where the Tablet lies.

The three garden rooms have the same description on purpose. Work out their exits from `EXITS`, or drop something in one of them to mark it, the way players did in 1977.

Item	Where	What it is for
`lamp`	in your bag from the start	the only light; `light lamp` and `douse lamp`
`key`	Well of Whispers	`unlock door` at the iron door
`ring`	Wizard's Tower	the jinn's gift: `give ring`
`earring`	Garden of Mirrors	nothing; a trap for careless code (Part 4)

Command	Short forms	What it does
`go north`, `go east`, `go south`, `go west`	`n e s w`, `north` ...	move
`look`	`l`	describe the room again
`inventory`	`i`	list what you carry
`take <item>`, `drop <item>`		pick up, put down, e.g. `take key`
`light lamp`, `douse lamp`		the lamp burns oil only while lit
`give <item>`		offer it to the jinn, e.g. `give ring`
`unlock door`		needs the key, at the door
`read tablet`		wins, by lamplight
`quit`		give up

A turn. Every command the game understands counts as one turn, even when it fails (`You can't go that way.` is a turn). A command the game does not understand is not a turn.

1 The Prologue

1. Print the banner: a rule of 30 equals signs (the starter makes it), `HOSHRUBA`, `An enchanted adventure`, then `A tale told by Umro Ayyar`, then the rule again.
2. Ask the player's name: `player = input("What is your name?  ")`.
3. Print Umro's greeting, with the name the player typed inside it: `Umro Ayyar looks you up and down.  "So you are Sara.  One riddle before you go."` where `Sara` is the name. Then print the riddle on its own line: `"What has a neck but no head?"` The double quotation marks are part of both lines, so put single quotes around the string in your `print()`. Then ask for the answer with `input("Your answer: ")`.
4. If the answer is exactly `bottle`, print `"Ha!  A bottle.  Take the big lamp."` and the oil is `BIG_LAMP`. Otherwise print `"No.  A bottle.  The small lamp, then."` and the oil is `SMALL_LAMP`. The quotation marks are part of the line.
5. Print `Your lamp holds 10 measures of oil.` with the oil of your lamp in place of `10`, then `You walk toward the green fire of the gate.`

Expected output of Part 1 when the player types `Sara` and `bottle`:

```
==============================
HOSHRUBA
An enchanted adventure
A tale told by Umro Ayyar
==============================
What is your name? Sara
Umro Ayyar looks you up and down. "So you are Sara. One riddle before you go."
"What has a neck but no head?"
Your answer: bottle
"Ha! A bottle. Take the big lamp."
Your lamp holds 10 measures of oil.
You walk toward the green fire of the gate.
```

The text inside `input(...)` is the *prompt*. Keep prompts inside `input(...)`, spelled exactly as shown, with the space at the end: `check.py` checks every prompt, and a prompt spelled differently fails the game.

Testing: the two Part 1 rows of `check.py` should say PASS. They check the prologue alone, so they pass before the loop is written.

2 The Loop and the Parser

The state. Before the loop, make one variable for everything that can change: the room you are in (`room`, starting at 0), the bag (`bag`, a string, see Part 4), where each loose item lies, whether the lamp is lit, the oil, the turn count, the whisper count, whether the jinn has gone, whether the door is open, whether to describe the room (`show`, starting `True`), whether the last command was understood, and the three ways the game can end. The starter makes the first six.

The loop. `while True:` runs until a `break`. One pass is one command.

1. **Describe the room**, but only if `show` is `True`: print `--` , the room's name and `--` (the starter does this), then the description (the same loop over `DESCS`). If the room is `JINN_ROOM` and the jinn has not gone, print `A jinn of fire sits here, watching you.` If any loose item lies in this room, print `You see:` followed by their names, in the order lamp, key, ring, earring, one space before each. Then set `show` to `False`.
2. **Read a command**: `command = input("> ")`. Split it at the first space: the part before is the *verb*, the part after is the *noun*. If there is no space, the verb is the whole command and the noun is `""`. Find the first space with a `for` loop over the indexes, and remember only the first one you meet.
3. **Short forms.** `n`, `e`, `s`, `w`, `north`, `east`, `south` and `west` on their own mean `go` with that direction. `i` means `inventory` and `l` means `look`. Change the verb and the noun before you look at them.
4. **One branch per verb.** `look` sets `show` back to `True`. `quit` ends the game. Any other verb, for now, prints `I don't understand that.` and does not count as a turn. The other verbs come one part at a time.
5. **End of the turn.** If the game has ended, `break`: the command that ends it is not counted. Otherwise, if the command was understood, add one to the turn count.

After the loop, print the ending. For `quit` it is `You leave the tilism unbroken.` and then `Turns:` with the count and a full stop.

Expected output of Parts 1 and 2 for the commands `look`, `xyzzy`, `quit` (Game 1 of `check.py`), after the prologue:

```
--- Umro's Camp ---
Tents, horses and woodsmoke. To the north, an arch of green fire marks the Gate of Illusions.
> look
--- Umro's Camp ---
Tents, horses and woodsmoke. To the north, an arch of green fire marks the Gate of Illusions.
> xyzzy
I don't understand that.
> quit
You leave the tilism unbroken.
Turns: 1.
```

Testing: run `check.py`. Game 1 and the other Part 2 rows should say PASS.

3 Moving

1. In the `go` branch, turn the noun into a direction number `d`: north 0, east 1, south 2, west 3. Any other noun prints `I don't know that direction.` and the command does not count as a turn.
2. Look up the exit: `EXITS[room * 4 + d]`. If it is `-`, print `You can't go that way.` Otherwise the new room number is that character turned into a number with `int()`, and `show` becomes `True` so the new room is described at the top of the next pass.

Expected output for `go north`, `go west`, `n`, `look`, `s` (from Game 2):

```
> go north
--- Gate of Illusions ---
Two stone lions guard the arch. A road runs north to a bridge, and a gap in the hedge opens east.
> go west
You can't go that way.
> n
--- Bridge of Fire ---
Black stone over a river of flame. A white dome lies north; to the west, steps go down into the dark.
A jinn of fire sits here, watching you.
> look
--- Bridge of Fire ---
Black stone over a river of flame. A white dome lies north; to the west, steps go down into the dark.
A jinn of fire sits here, watching you.
> s
--- Gate of Illusions ---
Two stone lions guard the arch. A road runs north to a bridge, and a gap in the hedge opens east.
```

Testing: Game 2 and the other Part 3 rows should say PASS. Game 2 also plays with the small lamp: check your prologue.

4 The Bag

The bag is one string with a space before and after every item: at the start `" lamp "`, later perhaps `" lamp key "`. The spaces matter. `"ring" in bag` is `True` when you carry an earring, because the word *earring* contains the word *ring*, and the jinn would take a ring you do not have. `" ring " in bag`, with the spaces, cannot be fooled. Each loose item also has a variable that says which room it lies in, or `-1` when it is in the bag.

1. `inventory`: print `You carry:` followed by the bag without its last character (`bag[:-1]`), so the line reads `You carry:  lamp key`. If the bag is just `" "`, print `You carry nothing.`
2. `take`: check, in this order, and print the first message that applies: no noun, `Take what?`; the room is dark and the lamp is not lit, `It is too dark to see anything.` (Part 5); the item does not lie in this room, `There is no X here.` with the noun in place of X, whatever it was; the bag already holds `BAG_LIMIT` items, `Your bag is full.`; otherwise add the noun and a space to the bag, set the item's room variable to `-1`, and print `Taken.` The bag holds one item fewer than it has spaces: count them with a `for` loop.
3. `drop`: no noun, `Drop what?`; the item is not in the bag, `You don't have that.`; otherwise cut it out of the bag with Umro's spell (the starter has it, and Part 8 is about it), set the item's room variable to this room, and print `Dropped.`

Expected output for `take earring`, `i`, `drop earring`, `i`, `take earring`, `take earring`, `take sword` in the garden room where the earring lies (from Game 3):

```
> take earring
Taken.
> i
You carry: lamp earring
> drop earring
Dropped.
> i
```

```
You carry: lamp
> take earring
Taken.
> take earring
There is no earring here.
> take sword
There is no sword here.
```

Testing: Game 3 and the other Part 4 rows should say PASS once Part 8 is also done; until then Game 3 fails at the `i` after `drop earring`.

5 Light and Dark

1. `light lamp`: if the noun is not `lamp`, `You can't light that.`; if the lamp is not in the bag, `You don't have the lamp.`; if the oil is 0, `The lamp is empty.`; if it is already lit, `It is already lit.`; otherwise it is lit now: `The lamp burns with a green flame.`
2. `douse lamp`: `You can't douse that.` for another noun; `The lamp is not lit.` if it is not; otherwise it is out: `You douse the lamp.`
3. **Oil.** At the end of every counted turn, if the lamp is lit, take one measure of oil. If that leaves 0, the lamp goes out (it is no longer lit) and you print `Your lamp flickers and goes out.` Otherwise print `Oil left:` and the number and a full stop.
4. **The dark room.** When you describe `DARK_ROOM` and the lamp is not lit, print `It is pitch dark. You hear whispers.` in place of the description and the items. Taking in the dark fails (Part 4).
5. **The whispers.** At the start of every turn, right after the room has been described: if the last command was understood and you are in the dark room without a lit lamp, add one to the whisper count. The first time, print `The whispers come closer.` The second time, print `Cold hands close on your shoulders. The whispers were the sorcerer's magic slaves.` and the game is over. Any understood command that leaves you out of the dark, or with the lamp lit, sets the count back to 0.

After the loop, that ending prints `THE TILISM HAS YOU.` and the `Turns:` line. Expected output of Game 4 after `n`, `n`, `light lamp`, `douse lamp`, `w`, `take key`:

```
> light lamp
The lamp burns with a green flame.
Oil left: 9.
> douse lamp
You douse the lamp.
> w
--- Well of Whispers ---
It is pitch dark. You hear whispers.
The whispers come closer.
> take key
It is too dark to see anything.
Cold hands close on your shoulders. The whispers were the sorcerer's magic slaves.
THE TILISM HAS YOU.
Turns: 6.
```

Testing: Game 4 and the other Part 5 rows should say PASS.

6 The Jinn and the Door

1. **The jinn bars the way.** In the `go` branch, after the `-` check: if you are in `JINN_ROOM`, the direction is `JINN_DIR` and the jinn has not gone, print `The jinn rises in a column of flame. "A gift, little trickster, or you shall not pass."` and stay where you are.
2. `give`: no noun, `Give what?`; not in the bag, `You don't have that.`; not in the jinn's room, or the jinn already gone, `There is nobody here to give it to.`; the ring: cut it out of the bag as in `drop` (it is gone for good), the jinn has gone, and print `The jinn slips the ring onto a finger of flame and bows. "Pass, trickster." It vanishes.`; the earring: `The jinn laughs. "That is an earring, not a ring."`; anything else: `The jinn wants none of that.`

3. **The door is locked.** Also in the `go` branch: in `DOOR_ROOM`, direction `DOOR_DIR`, door not open: `The iron door is locked.`
4. `unlock door`: another noun, `You can't unlock that.`; not in `DOOR_ROOM`, `There is no door here.`; no key in the bag, `You have no key.`; already open, `It is already open.`; otherwise the door is open now: `The key turns.  The iron door swings open.`

Once the jinn has gone, it is no longer mentioned when the bridge is described.

Testing: the three Part 6 rows should say PASS.

7 Breaking the Tilism

1. `read tablet`: another noun, `You can't read that.`; not in `TABLET_ROOM`, `There is nothing to read here.`; the lamp not lit, `The letters are dark.  They glow only in lamplight.`; otherwise print `The letters blaze green.  You read them aloud, and the tilism cracks like glass.` and `HOSHRUBA IS BROKEN.` and the game is won. Winning ends the game at once: that turn is not counted and burns no oil.
2. After the loop, for a win, print three lines. The first gives the turns and the oil, as in `Turns: 24. Oil left: 5.` The second is `Score:` and the score. The third is `Rank:` and the rank. The score is $100 - 2 \times \text{turns} + 5 \times \text{oil}$, and never below 0.

Score	Rank
80 or more	`Master Trickster`
60 to 79	`Trickster`
40 to 59	`Apprentice`
below 40	`Wanderer`

The end of the winning game in the appendix:

```
> read tablet
The letters blaze green. You read them aloud, and the tilism cracks like glass.
HOSHRUBA IS BROKEN.
Turns: 24. Oil left: 5.
Score: 77
Rank: Trickster
```

Testing: Game 5 should say PASS, and so should the other Part 7 rows: they read the Tablet in the wrong places and in the dark, and check the score and the rank on both sides of every boundary. The last five games, under Rule 4, play in tilisms you have not seen, with the same rules: if a room number is written into your code somewhere, Game 5 passes and some of them fail.

8 The Cursed Drop Spell

Umro wrote the `drop` spell in the starter for you, but a sorcerer has cursed it. Drop the key while you carry the lamp, and `inventory` says `You carry:  lampy`. The last letter of the thing you dropped stays behind, glued to the item before it. Find the mistake and fix it.

Hint. Take `bag = " lamp key "` and `noun = "key"`. Write the bag on paper with an index under every character. Where does `" key "` start? Which characters must go, so that `" lamp "` is what remains? Now compare with the slice in the spell.

Testing: the Part 8 row should say PASS, and so should Game 3 in Part 4.

When You Are Done

You are done when every row of `check.py` says PASS. Then your game prints exactly what the expected output in every part shows, uses only Lecture 1 to 8 tools with a comment on every block, and still works in a tilism

you have not seen (Rule 4).

- Keep the file name `ea1.py`: `check.py` looks for that name.
- Nothing is handed in: the game you wrote is yours to keep.

Common Mistakes

Each row is one thing that goes wrong for many students: what you see on the screen, why it happens, and what to do about it.

What you see	Why it happens	What to do
`go north` answers `I don't understand that.`, but `n` works.	The split at the first space is wrong: the verb still ends with the space, or the noun lost its first letter.	Print `"[" + verb + "]"` and `"[" + noun + "]"` for a moment and look at where the brackets fall. (Part 2)
The room is described after every command, or never again after the first time.	The `show` flag is never changed, or is changed in the wrong place.	Set `show = True` after a move and after `look`. Set `show = False` right after printing the description. (Part 2)
`IndexError: string index out of range` on the line that reads `EXITS`.	The index is wrong. It must be `room * 4 + d`, with `d` from 0 to 3.	Print `room` and `d` just before that line and check the arithmetic on paper. (Part 3)
`drop ring` drops the earring, or the jinn accepts the earring.	`"ring" in bag` is `True` when the bag holds `earring`, because `ring` is inside `earring`.	Test for `" ring "` with a space on both sides. That is why the bag is padded with spaces. (Part 4)
After `drop key`, `inventory` prints `You carry: lampy`.	The drop spell in the starter cuts one character too few, so a letter of the dropped item is left behind.	Write `" lamp key "` on paper, number the characters, and check where the slice should end. (Part 8)
`Oil left` is one too high or one too low.	The oil is burned at the wrong moment, or for a command that is not a turn.	Burn one measure at the end of each counted turn, after the command has been carried out, and only while the lamp is lit. `I don't understand that.` is not a turn. (Part 5)
`check.py` says your game asked for more input after `"> quit"`.	The game should have ended, but the loop kept running and asked for another command.	After `quit`, after reading the Tablet, and when the whispers catch you, leave the loop with `break`. (Part 7)
Every game in your tilism passes, but some of the Rule 4 games, in other tilisms, fail.	A room number is written into your code, for example `if room == 2` for the jinn. In another tilism the jinn sits elsewhere.	Use `JINN_ROOM`, `DOOR_ROOM`, `DARK_ROOM` and `TABLET_ROOM` from the `WORLD` block, never a number.
`check.py` says `your prompt is "» ", the handout's is "> "`.	A prompt is spelled differently from the handout: `"» "` instead of `"> "`, or a prompt that lost the space at its end.	Copy the three prompts from Parts 1 and 2: `"What is your name? "`, `"Your answer: "` and `"> "`, each with its space at the end.
`check.py` says `got ""`, or that your game printed a line too many.	A `print()` with nothing in it, or a line printed twice. Blank lines count.	Remove every `print()` that prints nothing, and print each line once.

Appendix: A Winning Game

The player types `Sara` and `bottle`. This is Game 5 of `check.py`.

```
==============================
HOSHRUBA
An enchanted adventure
A tale told by Umro Ayyar
==============================
What is your name? Sara
Umro Ayyar looks you up and down. "So you are Sara. One riddle before you go."
"What has a neck but no head?"
Your answer: bottle
"Ha! A bottle. Take the big lamp."
Your lamp holds 10 measures of oil.
You walk toward the green fire of the gate.
--- Umro's Camp ---
Tents, horses and woodsmoke. To the north, an arch of green fire marks the Gate of Illusions.
> go north
--- Gate of Illusions ---
Two stone lions guard the arch. A road runs north to a bridge, and a gap in the hedge opens east.
> go north
--- Bridge of Fire ---
Black stone over a river of flame. A white dome lies north; to the west, steps go down into the dark.
A jinn of fire sits here, watching you.
> light lamp
The lamp burns with a green flame.
Oil left: 9.
> go west
Oil left: 8.
--- Well of Whispers ---
The bottom of a dry well. Voices whisper from the walls.
You see: key
> take key
Taken.
Oil left: 7.
> go east
Oil left: 6.
--- Bridge of Fire ---
Black stone over a river of flame. A white dome lies north; to the west, steps go down into the dark.
A jinn of fire sits here, watching you.
> douse lamp
You douse the lamp.
> go south
--- Gate of Illusions ---
Two stone lions guard the arch. A road runs north to a bridge, and a gap in the hedge opens east.
> go east
--- Garden of Mirrors ---
Hedges of silver glass. Every path looks like every other path.
> go north
--- Garden of Mirrors ---
Hedges of silver glass. Every path looks like every other path.
You see: earring
> take earring
Taken.
> go east
--- Garden of Mirrors ---
Hedges of silver glass. Every path looks like every other path. An iron door stands in the north hedge.
> unlock door
The key turns. The iron door swings open.
> go north
--- Wizard's Tower ---
Shelves of bottled storms and sleeping birds. The wizard is away.
You see: ring
> take ring
Your bag is full.
> drop earring
Dropped.
> take ring
Taken.
> go south
--- Garden of Mirrors ---
Hedges of silver glass. Every path looks like every other path. An iron door stands in the north hedge.
> go east
--- Garden of Mirrors ---
```

```
Hedges of silver glass. Every path looks like every other path.
> go west
--- Gate of Illusions ---
Two stone lions guard the arch. A road runs north to a bridge, and a gap in the hedge opens east.
> go north
--- Bridge of Fire ---
Black stone over a river of flame. A white dome lies north; to the west, steps go down into the dark.
A jinn of fire sits here, watching you.
> give ring
The jinn slips the ring onto a finger of flame and bows. "Pass, trickster." It vanishes.
> go north
--- Dome of Light ---
A dome of white marble. On a pillar in the centre lies the Tablet of the Tilism.
> light lamp
The lamp burns with a green flame.
Oil left: 5.
> read tablet
The letters blaze green. You read them aloud, and the tilism cracks like glass.
HOSHRUBA IS BROKEN.
Turns: 24. Oil left: 5.
Score: 77
Rank: Trickster
```