COMP 102 — PROGRAMMING FUNDAMENTALS · FORMAN CHRISTIAN UNIVERSITY · FALL 2026

EA1: Try It Yourself

Ten small experiments with the game and K2

after the lecture, about an hour

Fakhir Shaheen · Sections A and B

Each experiment has three steps. **Predict**: write your answer on paper first. **Try**: do it on the computer. **Explain**: one sentence on why. A wrong prediction you can explain teaches you more than a right guess. You need the game with its X-ray, `fakhirshaheen.com/assets/courses/comp-102/assignments/hoshruba.html?xray`, and Thonny. The answers are at the back. For reading the handout itself, use `ea1_fex.pdf`.

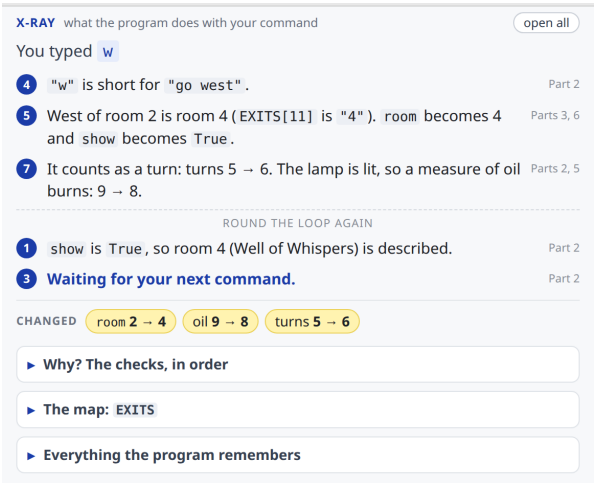
In the game

experiments 1 to 6

Reading the X-ray. It tells the story of your last command, in the order the program handled it.

- **The story**: one sentence per step of the loop, numbered as in the loop on page 4. After *round the loop again* comes the next pass, up to *Waiting for your next command.* The tag on the right names the Part of `ea1.pdf`.
- **Changed**: only what your command changed, old value → new value. ␣ is one space.
- **Three groups** that open with a click: *Why?* (the checks, in order), *The map* (this room's four exits; the one used is ringed) and *Everything the program remembers.*

Click an earlier command in the game to see its X-ray again. To start again: type `quit`, then press Enter at `>>`.



1 ▪ What costs a turn?

Type `Sara`, `bottle`, then `look`, `look`, `xyzzy`, `xyzzy`, `go west`. **Predict** the turns. **Try** it: step 7 of the story says whether each command counts. **Explain**: which commands were free, and why?

2 ▪ The oil clock

Start again. `n`, `n`, then `light lamp`, `look`, `look`, `look`, `douse lamp`, `look`. **Predict** the oil after each of the last six. **Try** it: read step 7 and *Changed* after each command. **Explain**: when exactly does oil burn?

3 ▪ Safe in the dark?

Start again. `n`, `n`, `w`, with the lamp out. Then `xyzzy`, `xyzzy`, `look`. **Predict** what happens after each of the last four. **Try** it: read step 2 of the story after each command. **Explain**: which commands bring the whispers closer?

4 ▪ Mark the garden

Start again. `n`, `e`, `n`, `take earring`, `e`, `drop earring`, `e`. Now walk back to the earring the shortest way, using the `EXITS` grid on page 4. **Predict** the room number after each of your moves. **Try** it: *Changed* shows `room`; open *Everything the program remembers* for `earring_room`. **Explain**: the three rooms look the same to you. How does the program tell them apart?

5 ▪ Who stops you?

Start again. `n`, `n`, `go north`. Then `s`, `e`, `e`, `e`, `go north`. **Predict** the reply to each `go north`. **Try** it, then open *Why?*: which check says yes? **Explain**: in both places the exit itself is not `-`. So what stops you?

6 ▪ Beat 77

The appendix of `ea1.pdf` scores 77. On paper, plan your own winning route: write every command, count the turns and the oil, and work out the score. Master Trickster needs 80. Then play it and compare. A measure of oil is worth 5 and a turn costs 2, so light the lamp only while you need it.

In K2

experiments 7 to 10

Open your `k2.py` from the lecture, or type it from the listing on page 2. Check it first: `n`, `n` ends with `Turns:  1.` Undo each change before the next experiment.

7 ▪ Watch the field loop

Click the line number 40, the `print` of the name: a red dot appears. *View* → *Variables*. *Run* → *Debug current script (faster)* (Shift+F5), then F8: it stops at the dot. **Predict** `field` and `name`. Then F8, type `n` in the Shell, and **Predict** them again when it stops. **Explain**: why does `field` end at the same number both times?

8 ▪ Add a camp, not code

Put a Camp Two between Base Camp and Camp Four. Change only `NAMES`, `DESCS`, `EXITS` and `SUMMIT`. **Predict** the new `EXITS` and `SUMMIT`. **Try** `n`, `n`, `n`: what are the last two lines? **Explain**: the loop did not change at all. Which rule of EA1 is this?

9 ▪ Break the index

Change `EXITS[room * 2 + d]` to `EXITS[room + d]`. **Predict** what `n`, `n`, `s` do from Base Camp. **Try** it. **Explain**: why `* 2` here, and `* 4` in EA1?

10 ▪ Count before you leave

Move the two lines `if understood:` and `turns = turns + 1` above `if won or leaving:`. **Predict** the last line after `n`, `n`. **Try** it. **Explain**: which rule of EA1 does the original order keep?

When your `ea1.py` and the game disagree.

1. Run your `ea1.py` (F5) and the game side by side. Type the same name, answer and commands into both.
2. Stop at the first line that differs, and read the game's X-ray at that moment.
3. Look at your own variables: a breakpoint on the first line inside your loop, Shift+F5, F8, *View* → *Variables*. Or print them between brackets for a moment, as the handout's Common Mistakes shows, and delete the `print` before you run `check.py`.

k2.py

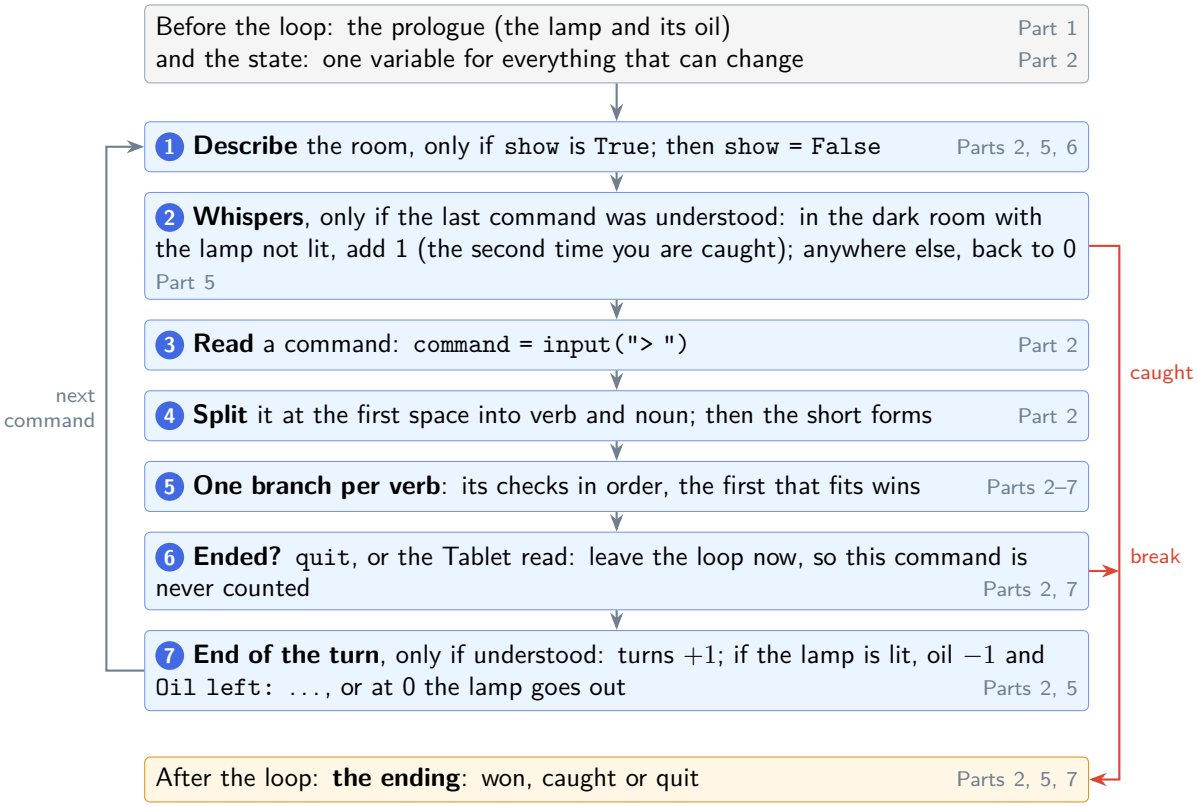
```
# The mountain: one name per camp, separated by |.  Camp 0 is the lowest.
NAMES = "Base Camp|Camp Four|Summit"
# One description per camp, separated by | in the same way.
DESCS = (
    "Tents on the Baltoro glacier. The ridge climbs north."
    + "|A ledge of blue ice. Base Camp is south, the summit north."
    + "|The top of K2. Clouds lie below you on every side."
)
# Two characters per camp, north then south: a camp number, or - for none.
EXITS = "1-" + "20" + "-1"
SUMMIT = 2

# What the climb remembers from one command to the next.
room = 0
show = True
turns = 0
won = False
leaving = False

# One pass of this loop is one command.
```

```
while True:
    # Describe the camp, but only after a move or a look.
    if show:
        # Walk along NAMES counting the | marks; keep the letters of camp room.
        field = 0
        name = ""
        for ch in NAMES:
            if ch == "|":
                field = field + 1
            elif field == room:
                name = name + ch
        print("--- " + name + " ---")
        # The description comes out of DESCS the same way.
        field = 0
        desc = ""
        for ch in DESCS:
            if ch == "|":
                field = field + 1
            elif field == room:
                desc = desc + ch
        print(desc)
        show = False

    # Read a command; n, s and l are short for north, south and look.
    command = input("> ")
    if command == "n":
        command = "north"
    elif command == "s":
        command = "south"
    elif command == "l":
        command = "look"

    # One branch per command.  The wind swallows any other word, free.
    understood = True
    if command == "look":
        show = True
    elif command == "quit":
        leaving = True
    elif command == "north" or command == "south":
        # Direction number: north 0, south 1.  Each camp has two exits.
        d = 0
        if command == "south":
            d = 1
        way = EXITS[room * 2 + d]
        if way == "-":
            print("You can't climb that way.")
        else:
            room = int(way)
            show = True
            if room == SUMMIT:
                won = True
    else:
        print("The wind takes your words.")
        understood = False

    # End of the pass: a win or a quit leaves at once, and is not counted.
    if won or leaving:
        break
    if understood:
        turns = turns + 1

# After the loop: the ending.
if won:
    print("You stand on the summit of K2.")
else:
    print("You climb down to the valley.")
print(f"Turns: {turns}.")
```

EA1 on one page

keep it open while you code

One pass of the loop



Check it with Game 1: `look`, `xyzzy`, `quit` ends with `Turns: 1`. Only `look` is counted: `xyzzy` is not understood, and `quit` leaves at 6, before 7.

The map as a grid

`EXITS` is this table written as one string. Room `r`, direction `d` (north 0, east 1, south 2, west 3) is `EXITS[r * 4 + d]`. The small number is that index.

r	Room	N (0)	E (1)	S (2)	W (3)
0	Umro's Camp	1 0	– 1	– 2	– 3
1	Gate of Illusions	2 4	6 5	0 6	– 7
2	Bridge of Fire	3 8	– 9	1 10	4 11
3	Dome of Light	– 12	– 13	2 14	– 15
4	Well of Whispers	– 16	2 17	– 18	– 19
5	Wizard's Tower	– 20	– 21	8 22	– 23
6	Garden of Mirrors	7 24	7 25	6 26	1 27
7	Garden of Mirrors	6 28	8 29	7 30	6 31
8	Garden of Mirrors	5 32	6 33	7 34	7 35

Every command

Checks run from the top; the first that fits prints its line and the rest are skipped. Every row is a turn unless it says **not a turn**.

When	It prints	Then
`look`, `l` Part 2		
1 always	—	`show = True`: described again at step 1
`quit` Part 2		

	When	It prints	Then
1	always	—	the game ends at step 6 **not a turn**
	`any other verb` Part 2		
1	always	`I don't understand that.`	**not a turn**
	`inventory, i` Part 4		
1	the bag is `"␣"`	`You carry nothing.`	
2	otherwise	`You carry:` and `bag[:-1]`	e.g. `You carry:  lamp key`
	`go dir, n e s w, north ...` Parts 3, 6		
1	the noun is not a direction	`I don't know that direction.`	**not a turn**
2	`EXITS[room * 4 + d]` is `-`	`You can't go that way.`	
3	in `JINN_ROOM`, going `JINN_DIR`, jinn still here	`The jinn rises in a column of flame.  "A gift, little trickster, or you shall not pass."`	
4	in `DOOR_ROOM`, going `DOOR_DIR`, door locked	`The iron door is locked.`	
5	otherwise	—	`room` = that exit, `show = True`
	`take item` Parts 4, 5		
1	no noun	`Take what?`	
2	in `DARK_ROOM`, lamp not lit	`It is too dark to see anything.`	
3	the item does not lie in this room	`There is no X here.`	X is the noun, whatever it is
4	the bag holds `BAG_LIMIT` items	`Your bag is full.`	
5	otherwise	`Taken.`	noun and a space added to the bag; its room is `-1`
	`drop item` Parts 4, 8		
1	no noun	`Drop what?`	
2	`"␣" + noun + "␣"` not in the bag	`You don't have that.`	
3	otherwise	`Dropped.`	cut out of the bag (Part 8); its room is this room
	`light lamp` Part 5		
1	the noun is not `lamp`	`You can't light that.`	
2	the lamp is not in the bag	`You don't have the lamp.`	
3	the oil is 0	`The lamp is empty.`	
4	the lamp is already lit	`It is already lit.`	
5	otherwise	`The lamp burns with a green flame.`	the lamp is lit
	`douse lamp` Part 5		
1	the noun is not `lamp`	`You can't douse that.`	
2	the lamp is not lit	`The lamp is not lit.`	
3	otherwise	`You douse the lamp.`	the lamp is out
	`give item` Part 6		
1	no noun	`Give what?`	
2	not in the bag	`You don't have that.`	
3	not in `JINN_ROOM`, or the jinn has gone	`There is nobody here to give it to.`	
4	the ring	`The jinn slips the ring onto a finger of flame and bows.  "Pass, trickster." It vanishes.`	ring cut out of the bag for good; the jinn has gone
5	the earring	`The jinn laughs.  "That is an earring, not a ring."`	
6	anything else	`The jinn wants none of that.`	

	When	It prints	Then
`unlock door` Part 6			
1	the noun is not `door`	`You can't unlock that.`	
2	not in `DOOR_ROOM`	`There is no door here.`	
3	no key in the bag	`You have no key.`	
4	the door is already open	`It is already open.`	
5	otherwise	`The key turns. The iron door swings open.`	the door is open
`read tablet` Part 7			
1	the noun is not `tablet`	`You can't read that.`	
2	not in `TABLET_ROOM`	`There is nothing to read here.`	
3	the lamp is not lit	`The letters are dark. They glow only in lamplight.`	
4	otherwise	`The letters blaze green. You read them aloud, and the tilism cracks like glass.` then `HOSHRUBA IS BROKEN.`	won **not a turn**
`step 7, the end of a turn` Parts 2, 5			
1	not understood	—	no turn, no oil
2	lamp lit, oil reaches 0	`Your lamp flickers and goes out.`	the lamp is out
3	lamp lit, oil left	`Oil left:` and the oil and a full stop	

Words

pass one trip round the `while` loop, steps 1 to 7. One pass per command.

command the line typed at the `>` prompt.

prompt the text inside `input(...)`. There are three.

verb, noun the command split at its first space: `go north` gives `go` and `north`. No space: the noun is `""`.

short form `n e s w`, `north` . . . , `i`, `l`: turned into the full verb and noun first.

transcript everything on the screen for one game. `check.py` compares transcripts.

turn a command the game understood, even one that fails.

understood `True`, except after `I don't understand that.` or `I don't know that direction.`

state every variable that can change while you play.

flag a Boolean that remembers whether something has happened, like `show`.

field a piece of `NAMES` or `DESCS` between two `|`. Room `r` is field `r`, from 0.

douse put out (a lamp). A **measure** is one unit of oil.

Answers

after you have tried

1 3: `look`, `look` and `go west`. The two `xyzzy` are free because the game does not understand them. `go west` fails, but the game understood it, so it counts.

2 9, 8, 7, 6, then 6 and 6. Oil burns at the end of each counted turn while the lamp is lit, starting with the turn that lights it. `douse lamp` puts it out before the end of its own turn.

3 After `w`: `The whispers come closer.` After each `xyzzy`: `I don't understand that.` and nothing more. A command the game does not understand never brings the whispers closer. After `look`: the Well again, then `Cold hands close on your shoulders...`, `THE TILISM HAS YOU.` and `Turns: 4.`

4 You drop the earring in room 8 and end up in room 6. Shortest way back: `n` (or `e`) to room 7, then `e` to room 8, where `You see: earring` and `earring_room` is 8. You see three identical rooms; the program sees the numbers 6, 7 and 8.

5 `The jinn rises in a column of flame. "A gift, little trickster, or you shall not pass."`, then `The iron door is locked.` The exits are `"3"` and `"5"`, so the `-` check says no; the jinn check and then the door check say yes. A way can be open on the map and still be barred.

6 One route that scores 93: `n`, `n`, `w`, `light lamp`, `take key`, `douse lamp`, `e`, `s`, `e`, `e`, `e`, `unlock door`, `n`, `take ring`, `s`, `e`, `w`, `n`, `give ring`, `n`, `light lamp`, `read tablet`. 21 turns and 7 measures left: $100 - 42 + 35 = 93$. It lights the lamp inside the Well: the whispers come once, and `light lamp` sends them back to 0.

7 `field` 2 and `name "Base Camp"`; then `field` 2 and `name "Camp Four"`. The loop always walks the whole of `NAMES`, so `field` ends at the number of `|` marks; `name` keeps only the letters of field `room`.

8 `EXITS = "1-" + "20" + "31" + "-2"` and `SUMMIT = 3`; `NAMES` and `DESCS` each get one more field. `n`, `n`, `n` ends with `You stand on the summit of K2.` and `Turns: 2.` The loop never names a camp: it reads the map. EA1's Rule 4 asks the same of your code, and `check.py` swaps in other tilisms to check it.

9 The first `n` reaches Camp Four (`EXITS[0]` happens to be right). The second prints `You can't climb that way.` because `EXITS[1]` is Base Camp's south. Then `s` reaches the summit and wins: `EXITS[2]` is Camp Four's north. Each camp owns 2 characters, so camp `r` starts at `r * 2`; in EA1 each room owns 4.

10 `Turns: 2.` instead of 1. The original order leaves the loop before counting, so a win or a quit is never a turn. It is EA1's rule too, the one behind Game 1's `Turns: 1.`