CS 106AX Programming Methodologies in JavaScript and Python Autumn 2026

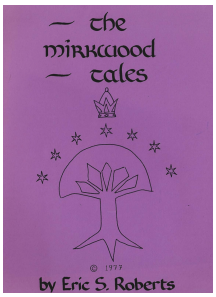
Lecture 20: A Life of Adventure!

Written by Ben Yan, from materials by Jerry Cain and Eric Roberts.

"Sorry! I don't want any adventures, thank you. Not Today." – Bilbo Baggins

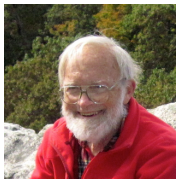
"I'm going on an adventure!" – Bilbo Baggins, after taking CS106AX

A Brief History of Time and Adventure



In early 1975, Professor Eric Roberts—the legendary Santa Claus of computer science education, a true gift to humanity, and the founding titan of CS198, which is Stanford's world-famous CS section leading program—embarked on a new and daring storytelling project called the ***Mirkwood Tales***. If that sounds familiar to you, you've done your share of fantasy reading! Mirkwood is the vast, dark, misty forest whose ominous shadows loom over the Middle-Earth universe of Tolkien's *Lord of the Rings* novel.

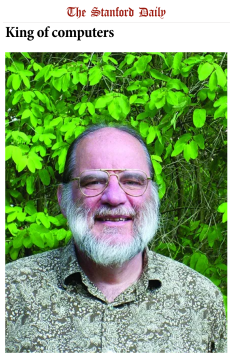
Eric's ***Mirkwood Tales*** was a complex and engaging role-playing adventure game inspired by *Dungeons & Dungeons*, and situated in Tolkien's Middle-Earth. Created when Eric was a graduate student at Harvard—why can't I be talented like that, fsck—the game became a regular fixture, even *obsession*, for friends and programmers in Cambridge who were working on the early Internet, called the ARPANET, at the time. The history of the Internet is thus deeply intertwined with not just the lore of Al Gore, but also that of Prof. Eric Roberts, who went on to become the greatest CS lecturer of all time until the arrival of Jerry Cain in 1997.



One was the regulars of Eric's game nights was **William Crowther**. He's like a side character who steals the show and gets his own brilliant spin-off, like Saul Goodman or Young Sheldon. Following a divorce ☹ from his wife Patricia, Crowther spent much of his spare time bringing a text-based adventure game to life, whose narrative was inspired by his years spelunking and mapping the world's largest cave system, the glittering *Mammoth Cave* in Kentucky. The game, developed on 700 lines of Fortran code, allowed a player to enter natural language inputs on a computer (simple commands such as **`EAST`**, **`WEST`**, **`TAKE WAND`**), and wander around a fantasy world rumored to sparkle with gold and ancient treasures. The program served also as the narrator.

And so ***Colossal Cave Adventure***, the first widely known interactive fiction game, was born. Two years later, in 1977, Don Woods, a Stanford grad student (I'm getting brutally mogged), released an expanded version of *Adventure*, adding magical items, spells, dragons, fantastic creatures, and more elaborate world-building.

By 1980, *Adventure* was ported to a wide variety of platforms, becoming an Internet and nerd culture sensation, and inspiring other adventure and role-playing games such as *Zork*, *Adventureland*, *Mystery House*, and Atari's graphical *Adventure* for video game consoles. Now, adventure games are everywhere, such as Minecraft!


The Stanford Daily
King of computers

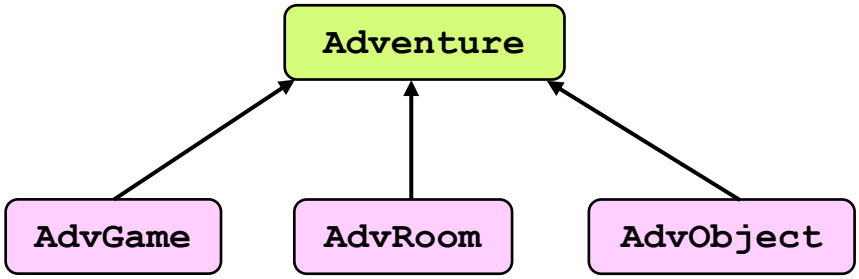
In 1984, Eric Roberts, then a professor at Wellesley College, expanded the game again, and used it as the basis for his first *Adventure Contest*, and later his famous *Adventure!* programming assignment. And as they say, the rest is history, though not all of it ☺, because **guess what you—my wonderful student—are going to have to implement yourself very very soon hehe**. We're so evil for this, but good luck!!

Classes in the Adventure Game

The Adventure game, which you'll create in Assignment 6 ☺, consists of three interrelated Python classes.

- **`AdvGame:`** Contains the code and data necessary for the user to play with the game.
- **`AdvRoom:`** Maintains the data structure for each room in the adventure world
- **`AdvObject:`** Maintains the data structure for each object that can be carried by the player.

We launch the game code by running the **`Adventure.py`** script, home to the main **`Adventure`** program.



Milestone #1

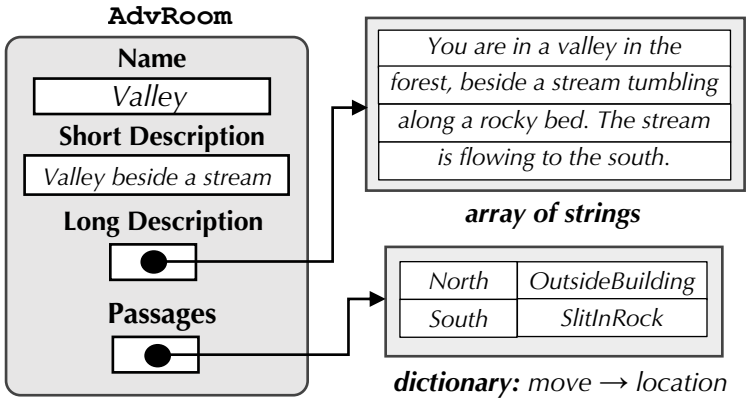
You'll first adapt the code from the ***Teaching Machine*** application so that it instead runs the game of *Adventure*, altering the class and method names as necessary. For example, instead of ***questions*** and ***possible answers*** we can choose from, we have ***locations / rooms*** and ***possible moves*** (e.g., **`EAST, NORTH, DOWN`**) we can choose from. All this data is stored in a text file, e.g., **`SmallRooms.txt`**.

```
OutsideBuilding
Outside building
You are standing at the end of a road before a small brick
building. A small stream flows out of the building and
down a gully to the south. A road runs up a small hill
to the west.
-----
WEST: EndOfRoad
UP: EndOfRoad
NORTH: InsideBuilding
IN: InsideBuilding
SOUTH: Valley
DOWN: Valley

EndOfRoad
End of road
You are at the end of a road at the top of a small hill.
You can see a small building in the valley to the east.
-----
EAST: OutsideBuilding
DOWN: OutsideBuilding
```

You'll implement the **AdvRoom** class, which stores information about a location or room in the game, e.g., ***Pirate Lair***, ***Valley***, ***Twisty Maze***, and its passages to other rooms, as read in from an excerpt of a text file.

```
Valley
Valley beside a stream
You are in a valley in the
forest, beside a stream tumbling
along a rocky bed. The stream
is flowing to the south.
-----
NORTH: OutsideBuilding
SOUTH: SlitInRock
```



Once this stepstone is completed, you should be able to wander around the surface geography of the game, which is bedizened with rivers, streams, villages, valleys, and surely enough, a colossal, mysterious cave ☺.

Adventure!

```
You are standing at the end of a road before a small brick
building. A small stream flows out of the building and
down a gully to the south. A road runs up a small hill
to the west.
> WEST
You are at the end of a road at the top of a small hill.
You can see a small building in the valley to the east.
> EAST
You are standing at the end of a road before a small brick
building. A small stream flows out of the building and
down a gully to the south. A road runs up a small hill
to the west.
>
```

Milestone #2

The next step is to implement the **setVisited** and **hasBeenVisited** methods in the AdvRoom class, which involve a **visited** flag for each room that captures whether we've been there before or not. This enables our game / narrator to print out the short description when you enter a previously visited room.

Adventure!

```
You are standing at the end of a road before a small brick
building. A small stream flows out of the building and
down a gully to the south. A road runs up a small hill
to the west.
> WEST
You are at the end of a road at the top of a small hill.
You can see a small building in the valley to the east.
> EAST
Outside building.
>
```

Milestone #3

You'll implement the **`QUIT`**, **`HELP`**, and **`LOOK`** commands, so that the player can end the game, see the built-in help message, and redisplay a room's long description (even after visiting that room), respectively.

Adventure!

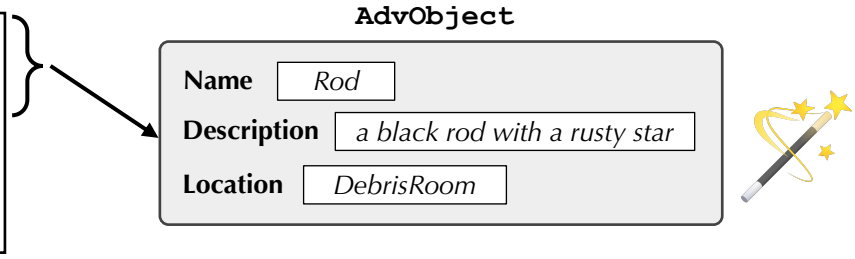
```
You are standing at the end of a road before a small brick
building. A small stream flows out of the building and
down a gully to the south. A road runs up a small hill
to the west.
> WEST
You are at the end of a road at the top of a small hill.
You can see a small building in the valley to the east.
> EAST
Outside building.
> LOOK
You are standing at the end of a road before a small brick
building. A small stream flows out of the building and
down a gully to the south. A road runs up a small hill
to the west.
> QUIT
```

Milestone #4

You've previously implemented the **`AdvRoom`** class, the template for objects that store information about a certain location or "room" of the Adventure game. Next, you'll implement the methods of the **`AdvObject`** class, **which makes it possible to keep track of the objects in a room**. You'll also modify the **`AdvGame`** class, to read in objects from a file, e.g., **`SmallRooms.txt`**, and put each object in its initial room.

```
ROD
a black rod with a rusty star
DebrisRoom

LAMP
a brightly shining brass lamp
BeneathGrate
```



We also change the code so that when we display a room, we also list out the objects inside. This milestone enables you to view—but not yet take—objects. For now, we ignore objects assigned to the **`"PLAYER"`** ☺.

Adventure!

```
Outside building.
> IN
You are inside a building, a well house for a large spring.
The exit door is to the south. There is another room to
the north, but the door is barred by a shimmering curtain.
There is a set of keys here.
>
```

Milestone #5

We implement the **TAKE**, **DROP**, and **INVENTORY** commands, and any code you need to recall what the player is carrying. This enables the user to take an object from the room they're currently in, remove an object from their possession and give it back to the room, and list all items held by the player, respectively.

Adventure!

```
You are inside a building, a well house for a large spring.
The exit door is to the south. There is another room to
the north, but the door is barred by a shimmering curtain.
There is a set of keys here.
> TAKE KEYS
Taken.
> TAKE GOLD
I don't see that here.
> INVENTORY
You are carrying:
a bottle of water
a set of keys
> DROP WATER
Dropped.
> DROP KEYS
Dropped.
> INVENTORY
You are empty-handed.
>
```

Milestone #6

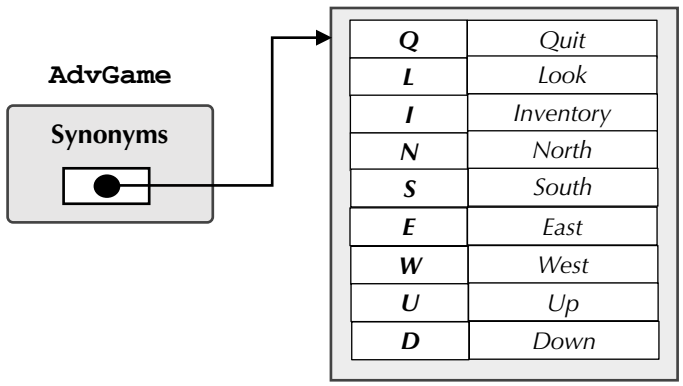
We implement synonym processing so that the player can use abbreviated forms of the direction verbs and alternative names for the objects, e.g., **I** for **INVENTORY**, **W** for **WEST**, **S** for **South**, **Q** for **Quit**, etc.

Adventure!

```
Welcome to Adventure
You are standing at the end of a road before a small brick
building. A small stream flows out of the building and
down a gully to the south. A road runs up a small hill
to the west.
> I
You are carrying:
a bottle of water
> DROP BOTTLE
Dropped.
> W
You are at the end of a road at the top of a small hill.
You can see a small building in the valley to the east.
> D
Outside building.
There is a bottle of water here.
>
```

Here's an example file, **`SmallSynonyms.txt`**, that stores synonyms, which we can read into a dictionary.

```
Q=QUIT
L=LOOK
I=INVENTORY
N=NORTH
S=SOUTH
E=EAST
W=WEST
U=UP
D=DOWN
```



dictionary: *abbreviation → full name*

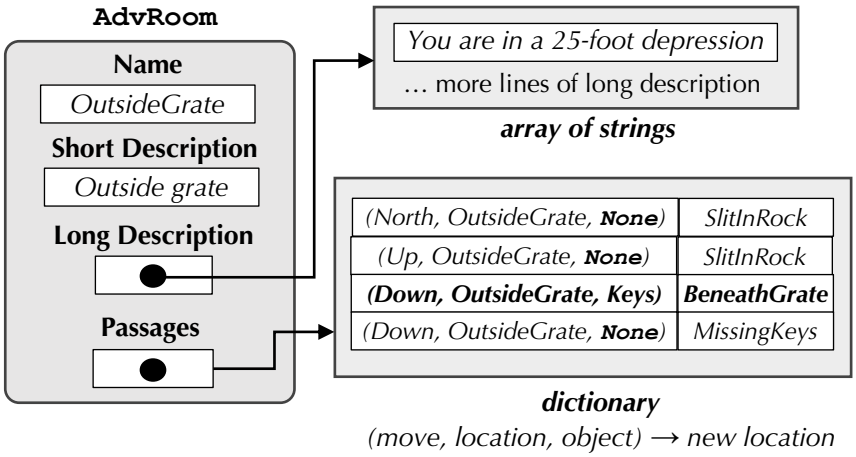
Milestone #7

We implement ***locked passages***, which are passages that require a particular object be held by the player to access it. For instance, in the below gameplay, the user needs ***a set of keys*** to go beneath the steel grate.

```
Adventure!
You are in a 20-foot depression floored with bare dirt.
Set into the dirt is a strong steel grate mounted in
concrete. A dry streambed leads into the depression from
the north.
> INVENTORY
You are carrying:
a bottle of water
a set of keys
> DOWN
You are in a small chamber beneath a 3x3 steel grate to
the surface. A low crawl over cobbles leads inward to
the west.
There is a brightly shining brass lamp here.
>
```

Enacting this change requires migrating the **`getNextRoom`** code from **`AdvRoom`** to **`AdvGame`**, so that it can see the objects the player carries. Here's how a locked passage requiring keys may appear in the rooms file.

```
OutsideGrate
Outside grate
You are in a 25-foot depression
… more lines of long description here …
-----
NORTH: SlitInRock
UP: SlitInRock
DOWN: BeneathGrate/KEYS
Down: MissingKeys
```



dictionary

(move, location, object) → new location

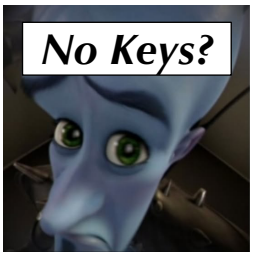
Milestone #8

> *"Nobody sails into hell and comes back"* – Odysseus
> *"You will."* – Circe

The final milestone, and the algorithmically most tricky and thrilling one, is to implement ***forced motion***, where a player is forced to move from a room even before reading a command. Forced motion is indicated by the special verb **FORCED**, as opposed to a movement verb the user inputs, e.g., **NORTH**, **DOWN**, or **SOUTH**.

It is crucial to ensure that your implementation of forced motion allows these passages to be locked.

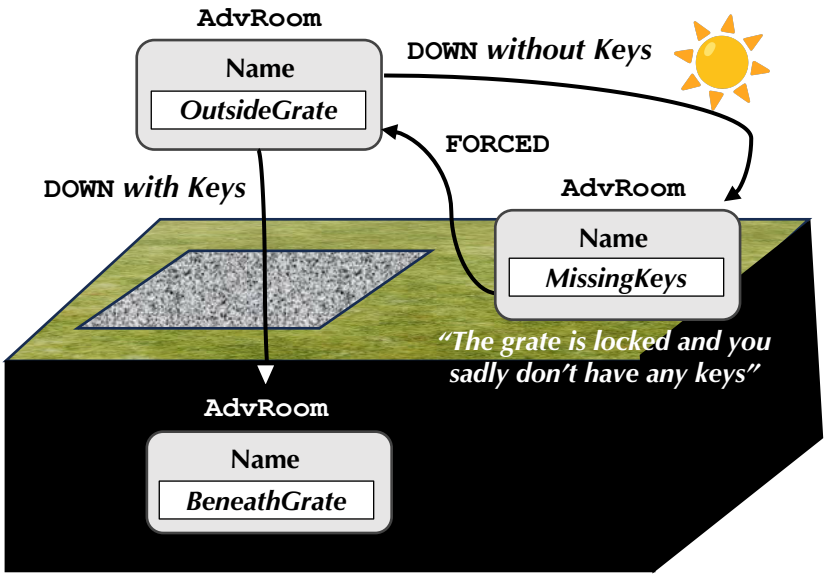
```
Adventure!
You are in a 20-foot depression floored with bare dirt.
Set into the dirt is a strong steel grate mounted in
concrete. A dry streambed leads into the depression from the north.
> INVENTORY
You are carrying:
a bottle of water
> DOWN
The grate is locked and you sadly don't have any keys.
Outside grate.
>
```



What's obscured in the gameplay above is **what truly happens when we attempt to move into a locked room**, e.g., from ***OutsideGrate*** to ***BeneathGrate***, without the needed items. When Crowther designed the original game, he simply created new, intermediate rooms, e.g., ***MissingKeys***, whose descriptions contained the messages he wanted to deliver. Of course, you wouldn't want a player to end up in one of these limbo rooms, so they would be automatically moved—via the **FORCED** motion—back to the ***OutsideGrate*** room.

```
OutsideGrate
Outside grate
You are in a 25-foot depression
floored with bare dirt. Set into
… more lines of long description here …
-----
NORTH: SlitInRock
UP: SlitInRock
DOWN: BeneathGrate/KEYS
Down: MissingKeys

MissingKeys
Above locked grate
The grate is locked and you
sadly don't have any keys.
-----
FORCED: OutsideGrate
```



The player had no idea they were in ***MissingKeys***, only that a message *The grate is locked...* was printed ☺.