

"A huge green fierce
snake bars the way";

Or, Building a Text Adventure Game in Python

Enter name:

> astrosilverio

> examine talk

what are we doing here

> **How does a text adventure work?**

- Why make a text adventure framework?
- How does a text adventure framework work?
- Let's make a framework!

> go north
Filch's Office

You are in a small, spotless room. A filing cabinet stands in the corner and a discontented cat hisses at you from the floor. The door is to the south.

> take cat
You can't take that.

> examine cat
Dust-colored cat with large lamplike eyes. Scrawny and very bad tempered.

>
I didn't understand any of that.

> examine cabinet
You pull open a drawer labeled 'Confiscated and Highly Dangerous.'

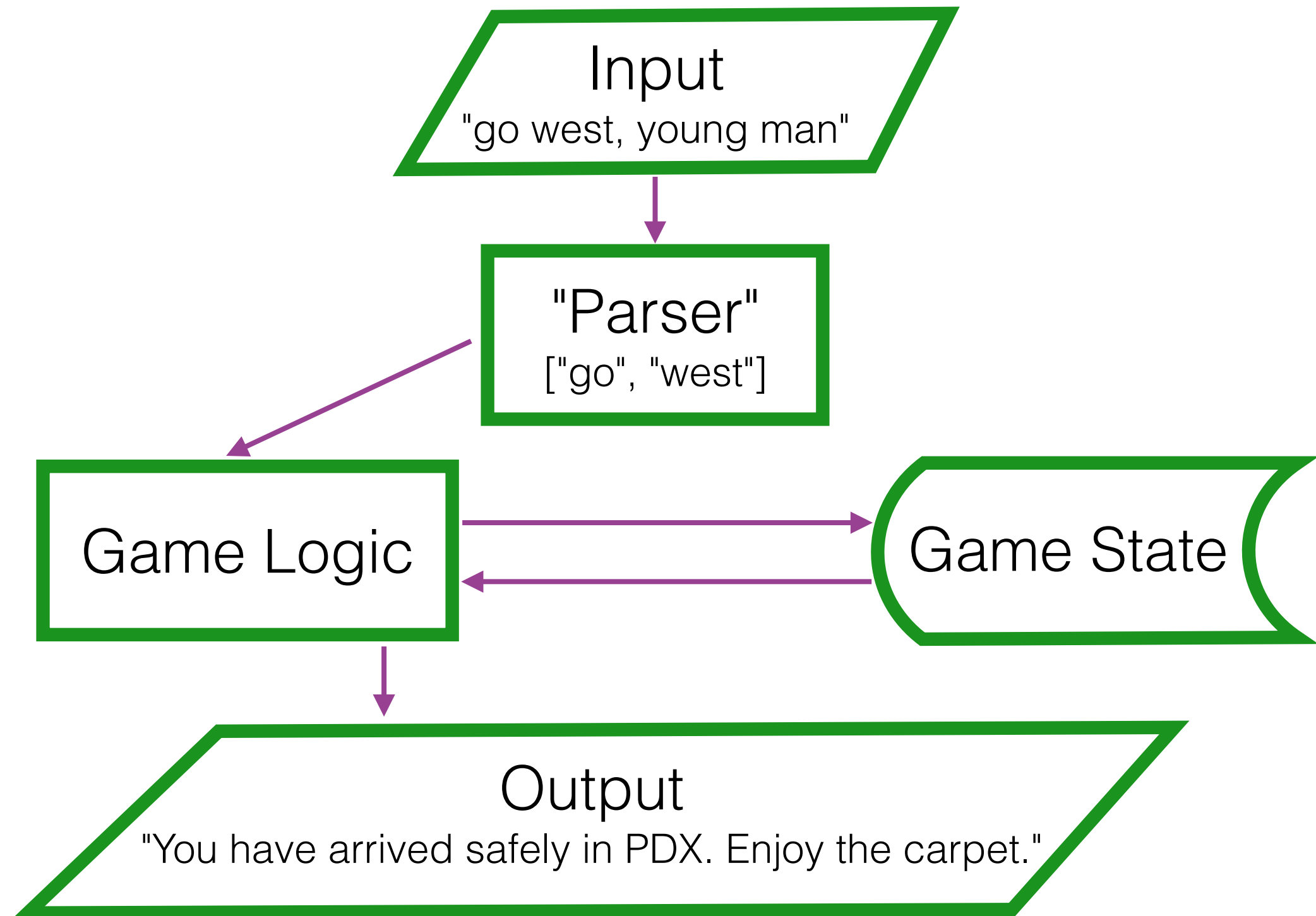
> look
Filch's Office

You are in a small, spotless room. A filing cabinet stands in the corner and a discontented cat hisses at you from the floor. The door is to the south.

A shimmering cloak lies crumpled in liquid-like folds.

> take cloak
> inventory
You are carrying:
wand
invisibility cloak

> █





Room

- has a description
- connects to other rooms
- has an inventory
- may have a password

Thing

- has a description
- may have an inventory
- may be alive
- may be edible
- ...

Player

- has a location
- has an inventory
- may have a House

`["examine", "cat"]`



`examine(cat)`

`["take", "cat"]`



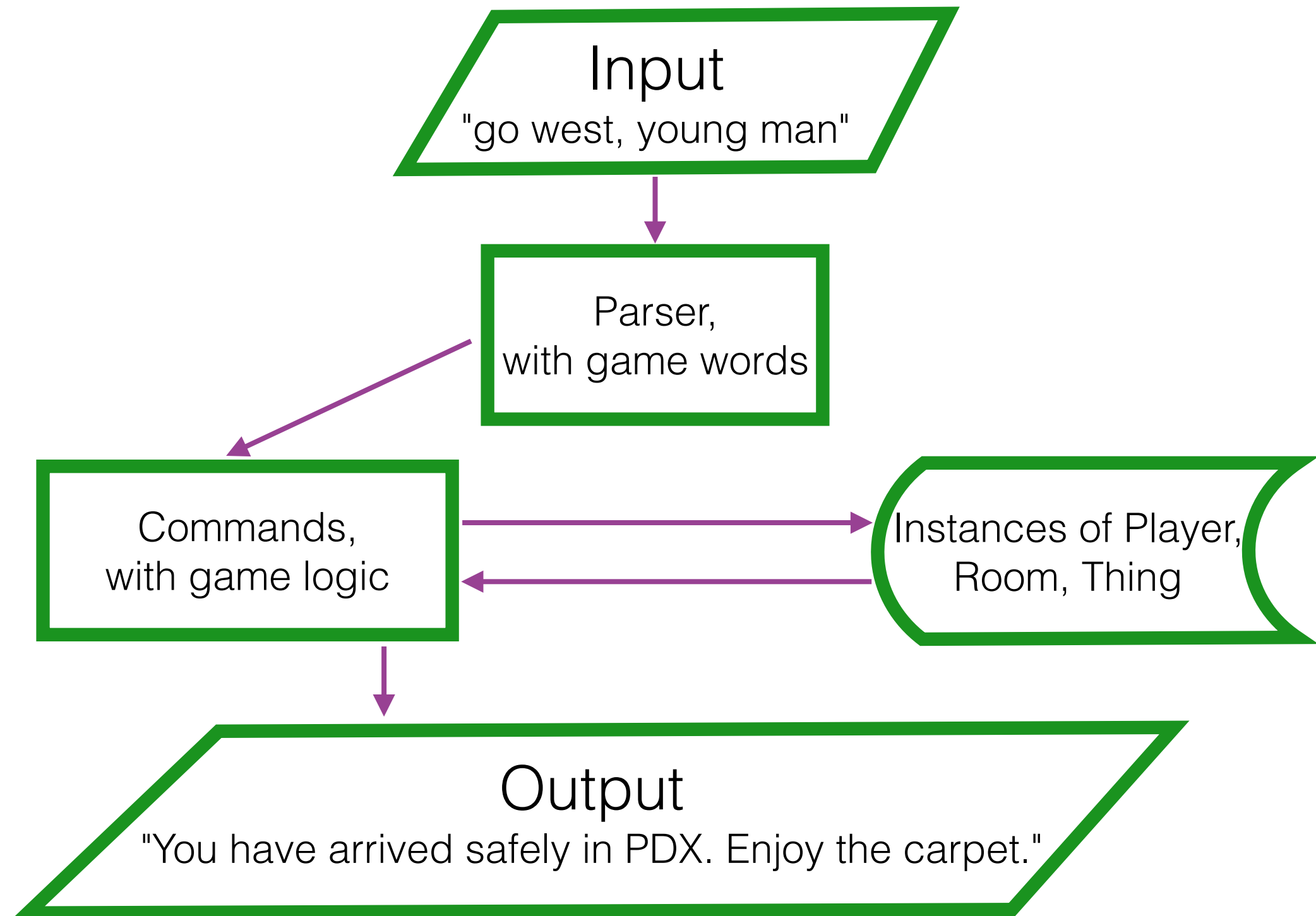
`take(cat)`

Command

```
def examine(thing):  
    return thing.description
```

```
def take(thing):  
    room.contents.remove(thing)  
    player.contents.add(thing)  
    return player.contents
```





> examine talk

what are we doing here

- How does a text adventure work?

> **Why make a text adventure framework?**

- How does a text adventure framework work?
- Let's make a framework!

```
In [1]: run(hogwarts.init)
```

You are in King's Cross on the platform between tracks 9 and 10. You're running slightly late for the Hogwarts Express. The train must be boarding by now, but you don't see any trains and the few other people on the platform seem completely relaxed.

There is a solid-looking brick pillar nearby.

```
> run at pillar
```

```
In [1]: run(firefly.init)
```

You are in a sort of spacecraft parking lot on Persephone.

The Nu Du Shen, a sleek Floating World Class cruise liner, gleams nearby. The Brutus, a simple but solid-looking Orion Cruiser, bustles with activity. The engines of a shiny Alliance corvette purr as they warm up.

There is a battered Firefly-class transport ship here.

```
> enter firefly
```

a multiplying sword

a case study

a multiplying sword

```
> examine sorting hat
```

```
It's tattered and torn and really incredibly dirty.
```

a multiplying sword

> what's my house

You are in Gryffindor House. You are brave and chivalrous, and possibly obnoxious. You probably have red hair.

> examine sorting hat

It's tattered and torn and really incredibly dirty.

A silvery sword falls out of the hat! Congratulations, you are a true Gryffindor!

a multiplying sword

```
> examine sorting hat
```

It's tattered and torn and really incredibly dirty.

A silvery sword falls out of the hat! Congratulations, you are a true Gryffindor!

```
> examine sorting hat
```

It's tattered and torn and really incredibly dirty.

A silvery sword falls out of the hat! Congratulations, you are a true Gryffindor!

```
> inventory
```

Sword of Gryffindor
Sword of Gryffindor

“Much of my work has come from being lazy. I didn’t like writing programs, and so...I started work on a programming system to make it easier to write programs.”

– John Backus, 1979, in Think, the IBM employee magazine

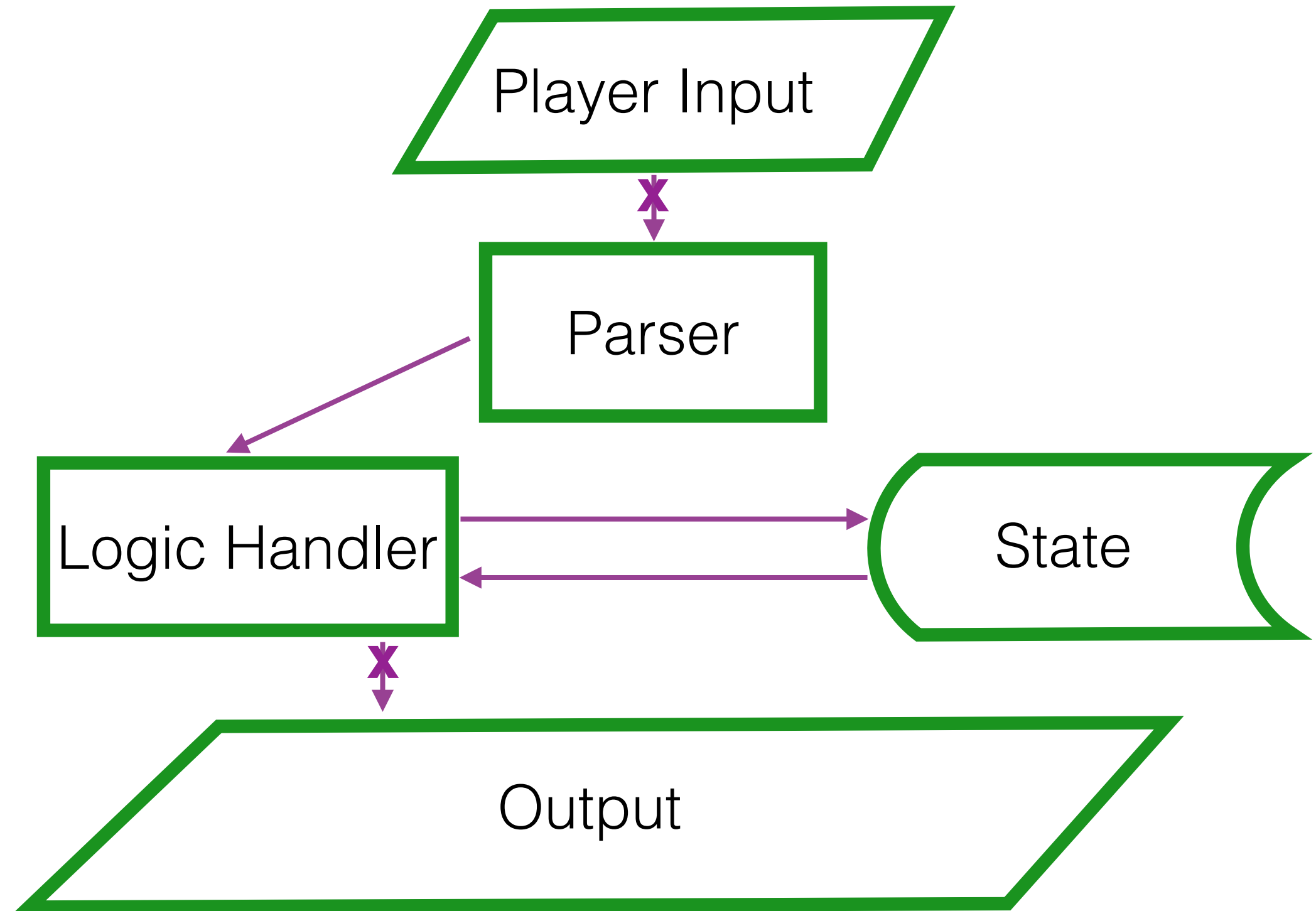
> examine talk

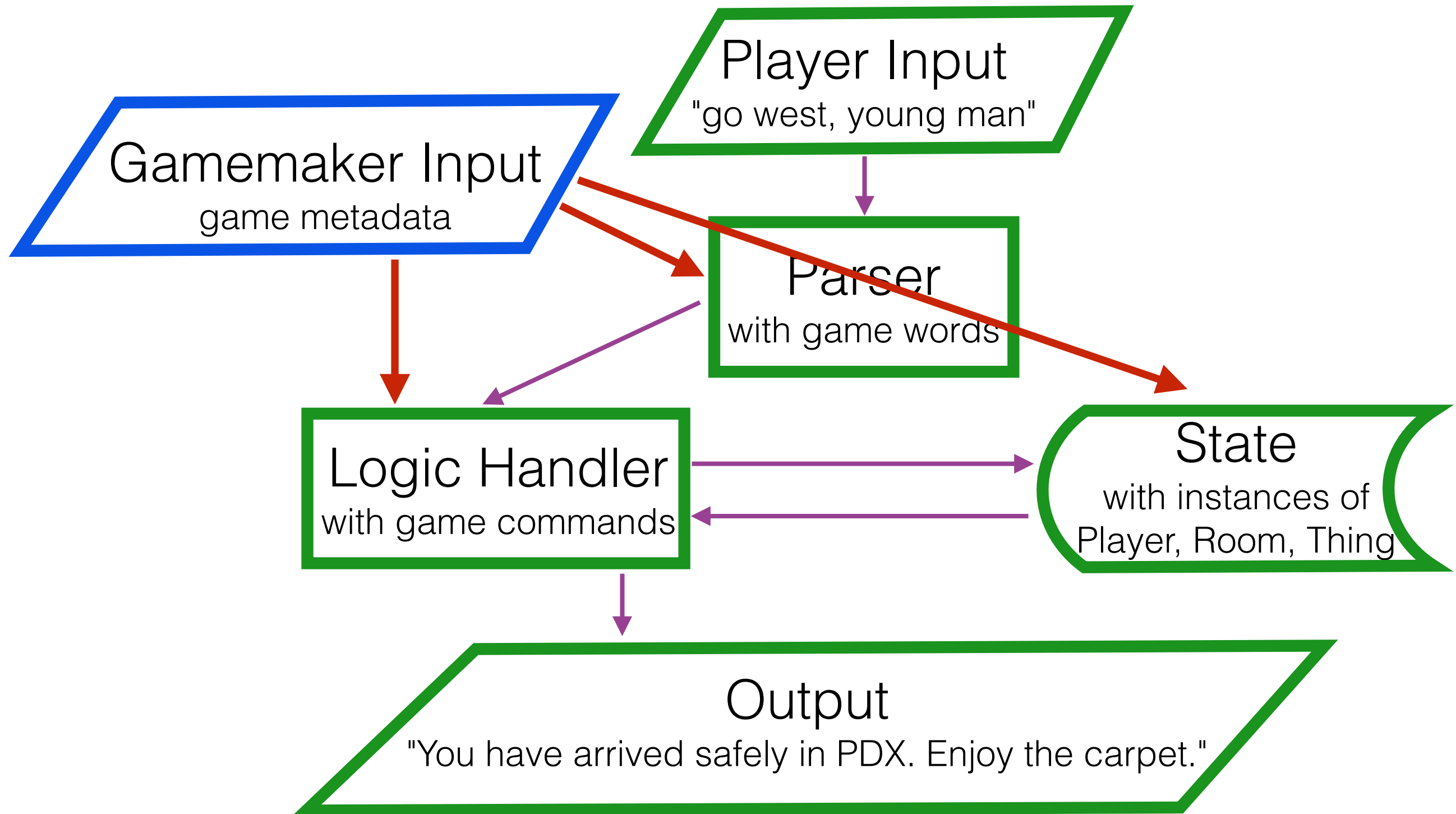
what are we doing here

- How does a text adventure work?
- Why make a text adventure framework?

> How does a text adventure framework work?

- Let's make a framework!





> examine talk

what are we doing here

- How does a text adventure work?
- Why make a text adventure framework?
- How does a text adventure framework work?

> Let's make a framework!

> activate translator

a brief word about abstracting parsing

> It has been coming on so gradually, that I hardly know when it began. But I believe I must date it from my first seeing his beautiful grounds at Pemberley.

> It has been coming on so gradually, that I hardly know when it began. But I believe I must **date** it from my first seeing his beautiful grounds at **Pemberley.**

> It has been coming on so gradually,
that I hardly know when it began. But I
believe I must **date** it from my first seeing
his beautiful grounds at **Pemberley**.



```
graph TD; A["date"] --> C["date"]; B["Pemberley"] --> C; C --> D["I'm sorry, you cannot date a location."];
```

["date", "Pemberley"]

"I'm sorry, you cannot date a location."

i'll do what i want

customizable commands

> define command

go('wand') => wat

go('south') => gotcha

> define command

```
go('wand') => wat  
go('south') => gotcha
```

```
# you are in antarctica  
go('south') => "can't"
```

```
# you are in vancouver wa  
go('south') => "welcome to pdx"
```


> execute command

- **syntax check** – “do these arguments make sense? I can’t exactly take east.”
- **logic check** – “can I do that? I can’t go west if there’s a brick wall in that direction.”
- **(optional) state change** – “ok looks like I can take that book for you, let me modify your inventory accordingly”
- **formulate response** – “I’ve moved you south, here is the description of the new room you’re in”

can i go now?

```
def is_a_direction(map, word):  
    if word not in map.directions:  
        raise CommandError("Where do you want me to go?")  
  
def path_exists(map, player, direction):  
    if direction not in player.location.paths:  
        raise CommandError("You can't go that way.")  
  
def move_player(map, player, direction):  
    new_location = player.location.paths[direction]  
    player.location = new_location  
  
go = Command(  
    name='go',  
    syntax=[is_a_direction],  
    rules=[path_exists],  
    state_changes=[move_player],  
    response=location_description)
```

oobleck!

```
def not_in_oobleck(map, player, direction):  
    if player.location.oobleck:  
        raise CommandError("You are stuck in oobleck!")  
  
go = Command(  
    name='go',  
    syntax=[is_a_direction],  
    rules=[path_exists, not_in_tractor_beam],  
    state_changes=[move_player],  
    response=location_description)
```

> examine talk

what are we doing here

- How does a text adventure work?
- How does a text adventure framework work?
- Let's make a framework!

> Let's do cool things with a framework!

> release basilisk

```
def turn_100(map, player):  
    if len(player.history) == 100:  
        return True  
  
def release_basilisk(map, player):  
    map['second floor bathroom'].inventory.add(basilisk)  
  
basilisk = Command(  
    rules=[turn_100],  
    state_changes=[release_basilisk])
```

> put on sorting hat

```
def turn_25(map, player):  
    if len(player.history) == 25:  
        return True
```

```
def sort_player(map, player):  
    house = <complex sorting algorithm>  
    player.house = house
```

```
sort = Command(  
    rules=[turn_25],  
    state_changes=[sort_player],  
    response=player_house)
```


> abracadabra!

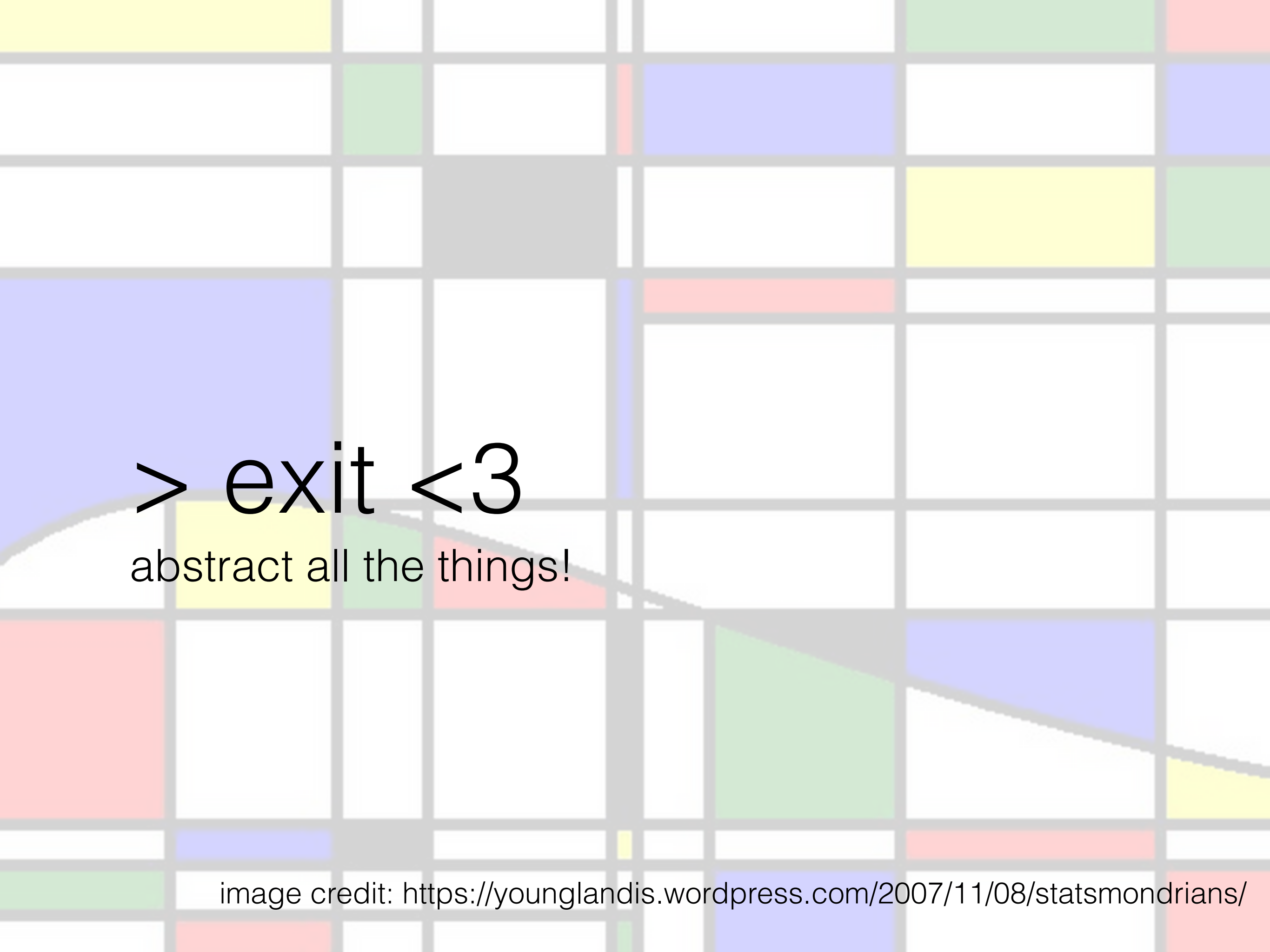
"I'm sorry, did you mean Expelliarmus?"

```
expelliarmus = Command(  
    name='expelliarmus',  
    syntax=[is_a_player],  
    rules=[carrying_wand],  
    state_changes=[disarm_player],  
    response=disarm_message)
```

> abracadabra!

"I'm sorry, did you mean Expelliarmus?"

```
def can_cast_spells(map, player):  
    player.spell_skill += 1  
    if not player.spell_skill > 5:  
        raise CommandError("Nothing happens.")  
  
expelliarmus = Command(  
    name='expelliarmus',  
    syntax=[is_a_player],  
    rules=[carrying_wand, can_cast_spells],  
    state_changes=[disarm_player],  
    response=disarm_message)
```



> exit <3
abstract all the things!

image credit: <https://younglandis.wordpress.com/2007/11/08/statsmondrians/>

if ye had the chance
to change yer state...

customizable stateful things

> make cat

entity

Alive

- can be asleep / awake
- can be hungry
- has a mood

component

Portable

- can be carried

component

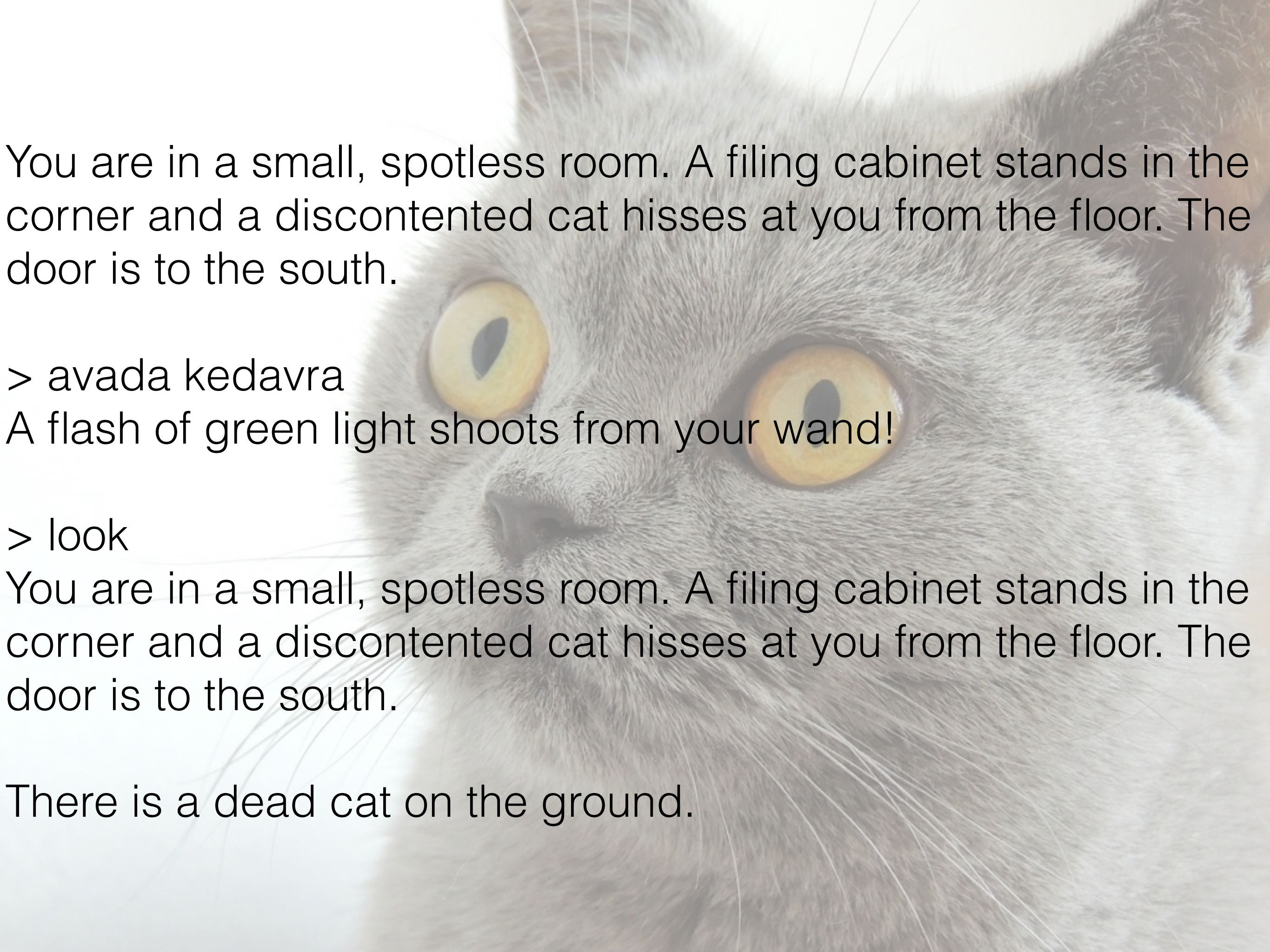
Container

- has inventory

component

an undead cat

a case study



You are in a small, spotless room. A filing cabinet stands in the corner and a discontented cat hisses at you from the floor. The door is to the south.

> avada kedavra

A flash of green light shoots from your wand!

> look

You are in a small, spotless room. A filing cabinet stands in the corner and a discontented cat hisses at you from the floor. The door is to the south.

There is a dead cat on the ground.

> ask forgiveness

```
def go(*args):  
    try:  
        current_location = player.location  
        new_location = current_location.paths[args[0]]  
        player.location = new_location  
    except KeyError:  
        return "I can't go that direction"  
    else:  
        return player.location.description
```


> ask more forgiveness

```
def go(*args):  
    try:  
        current_location = player.location  
        new_location = current_location.paths[args[0]]  
        player.change_location(new_location)  
    except KeyError:  
        return "I can't go that direction"  
    except OobleckError:  
        return "Your shoes are covered in oobleck."  
    else:  
        return player.location.description
```

> next

what's new for phase two

> accio broom

game-specific commands

retro

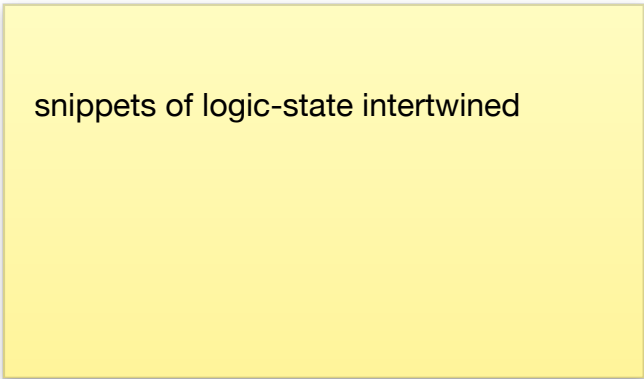
> examine hat

can we prevent having infinite swords?

design principles

> examine game

here is some gross code I wrote



snippets of logic-state intertwined

reimplement examine

```
def is_a_thing(map, word):  
    if word not in map.things:  
        raise CommandError("What do you want me to examine?")  
  
def is_gryffindor(map, player, direction):  
    if player.house != "Gryffindor":  
        return False  
  
def drop_sword(map, player, direction):  
    new_location = player.location.paths[direction]  
    player.location = new_location  
  
examine = Command(  
    name='examine',  
    syntax=[is_a_thing],  
    rules=[is_gryffindor],  
    state_changes=[drop_sword],  
    response=thing_description)
```


a multiplying sword

```
def examine(thing, player):  
    output = thing.description  
    if thing.is_sorting_hat:  
        distance = find_distance(sword)  
        if distance == -1 and sword not in player.inventory:  
            if player.house == "Gryffindor":  
                output = "A silver sword falls out of the hat."  
                player.add_inventory(sword)  
    return output
```

> help

Colossal Cave Adventure ▶ Score: 63 ▶ Turns: 13

~~The little bird attacks the green snake, and in an~~
astounding flurry drives the snake away.

Ok.

> SW

You are in a secret canyon which here runs E/W. It crosses
over a very tight canyon 15 feet below. If you go down you
may not be able to get back up.

> go west

You are in a secret canyon which exits to the north and east.
A huge green fierce dragon bars the way!
The dragon is sprawled out on a persian rug!

> kill dragon

With what? Your bare hands?

> yes

Congratulations! You have just vanquished a dragon with
your bare hands! (Unbelievable, isn't it?)

What's next? ■