Escape Room

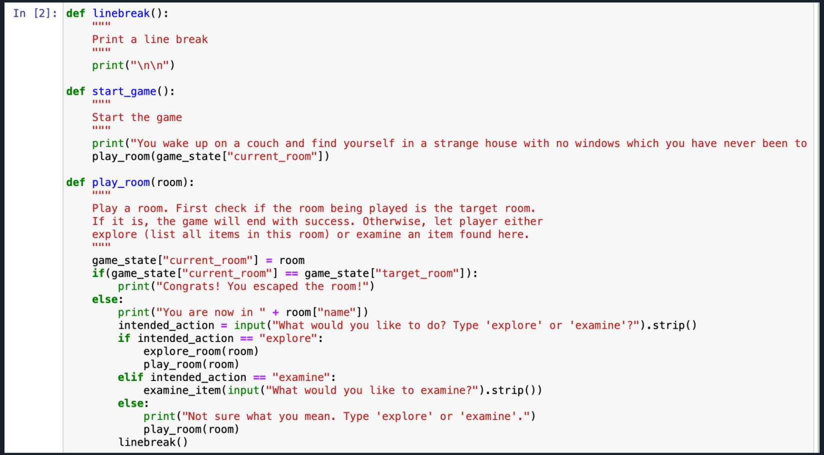
Juan Fran, Natalia, Javier

```python
couch = {
    "name": "couch",
    "type": "furniture",
}

door_a = {
    "name": "door a",
    "type": "door",
}

key_a = {
    "name": "key for door a",
    "type": "key",
    "target": door_a,
}

piano = {
    "name": "piano",
    "type": "furniture",
}

game_room = {
    "name": "game room",
    "type": "room",
}
```

Project Overview

Functions:

- Functions that includes:
  - A flow control with conditional statements to:
    - Start the game
    - Check the target room
    - The game will end with success
    - To explore or examine items
  - Iterations: for loop to:
    - Explore room
    - Connected rooms to the given door
    - Examine item (door or furniture) and collect keys.



```
In [2]: def linebreak():
            """
            Print a line break
            """
            print("\n\n")

        def start_game():
            """
            Start the game
            """
            print("You wake up on a couch and find yourself in a strange house with no windows which you have never been to
            play_room(game_state["current_room"])

        def play_room(room):
            """
            Play a room. First check if the room being played is the target room.
            If it is, the game will end with success. Otherwise, let player either
            explore (list all items in this room) or examine an item found here.
            """
            game_state["current_room"] = room
            if(game_state["current_room"] == game_state["target_room"]):
                print("Congrats! You escaped the room!")
            else:
                print("You are now in " + room["name"])
                intended_action = input("What would you like to do? Type 'explore' or 'examine'?").strip()
                if intended_action == "explore":
                    explore_room(room)
                    play_room(room)
                elif intended_action == "examine":
                    examine_item(input("What would you like to examine?").strip())
                else:
                    print("Not sure what you mean. Type 'explore' or 'examine'.")
                    play_room(room)
                linebreak()
```

Example: “linebreak”, “start_game”, “play_room”

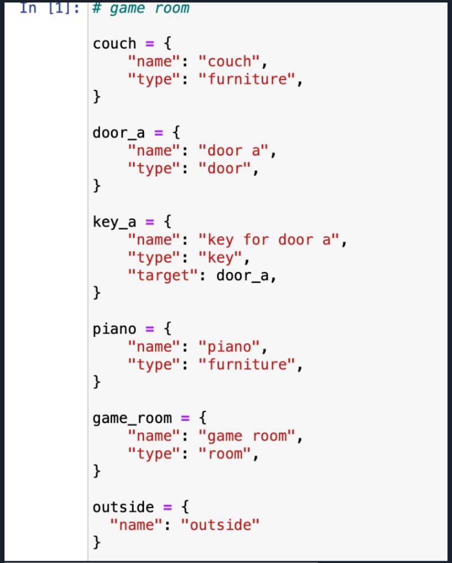
Functions

- Game starts once all cells are run
- Functions check where player starts and always checks in which room player is
- Functions offer player two inputs:
  - “Explore”: checks which room you are in and displays “examinable” items of that room
  - “Examine”: Triggers another input which asks which items of said room you want to examine.
- Win conditions is if “current rooms” = outside
- Player needs to find keys “examining” the items of rooms and using said keys for their corresponding doors

Technical Challenges: Dictionaries and functions

Dictionaries:

- Main technical challenges, define a notebook with:
  - Dictionaries that implement :
    - Virtual spaces in a cell to declare furniture, keys and doors, with:
      - keys: defining object name
      - values: defining tipology
      - Only difference in this dictionaries is for the keys, that will show a third key-value pair named "target" that execute with a loop (for) the possibility of open doors with their correct keys
    - The event that initialize game state

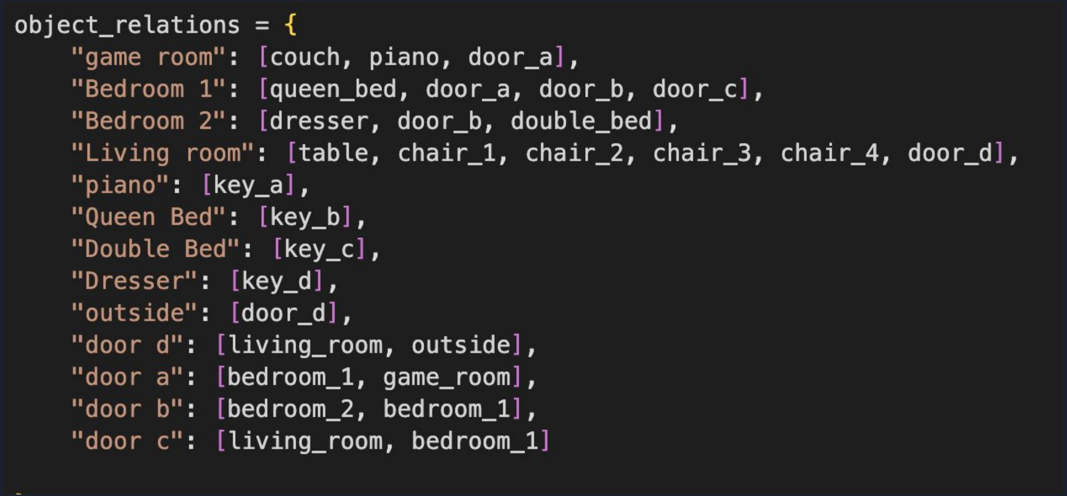


```
In [1]: # game room

        couch = {
            "name": "couch",
            "type": "furniture",
        }

        door_a = {
            "name": "door a",
            "type": "door",
        }

        key_a = {
            "name": "key for door a",
            "type": "key",
            "target": door_a,
        }

        piano = {
            "name": "piano",
            "type": "furniture",
        }

        game_room = {
            "name": "game room",
            "type": "room",
        }

        outside = {
          "name": "outside"
        }
```

Example: "game room"

Technical Challenges: Overcome the challenge

- Clone rooms with their connections: doors + keys
- Relate objects in order to found keys in objects, open doors and scape

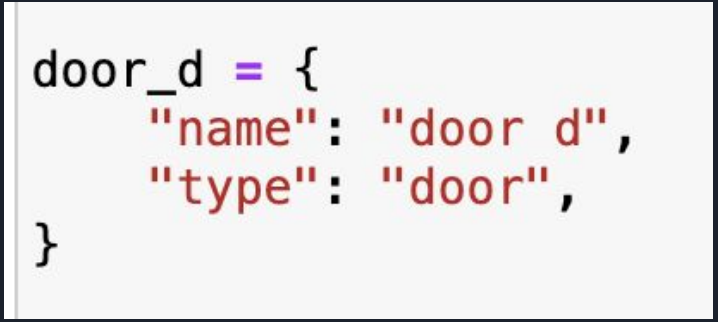


```
object_relations = {
    "game room": [couch, piano, door_a],
    "Bedroom 1": [queen_bed, door_a, door_b, door_c],
    "Bedroom 2": [dresser, door_b, double_bed],
    "Living room": [table, chair_1, chair_2, chair_3, chair_4, door_d],
    "piano": [key_a],
    "Queen Bed": [key_b],
    "Double Bed": [key_c],
    "Dresser": [key_d],
    "outside": [door_d],
    "door d": [living_room, outside],
    "door a": [bedroom_1, game_room],
    "door b": [bedroom_2, bedroom_1],
    "door c": [living_room, bedroom_1]
```

Big Mistake

Giving a wrong value to one of the dictionaries, labelling a door as a key.

This meant that everytime you examined the final door, the game interpreted it as a key and entered in a loop where you were stuck in the living room forever.



```
door_d = {
    "name": "door d",
    "type": "door",
}
```

door != key

To solve it, we changed the value associated to the key “type”, going from “key” to “door”.

After executing the code again, we were able to escape the room and the game ended.

Features and new overcomes

Features:

- Create new dictionaries with their unique keys and values.
- Create an alternative choice with a new winning condition.
- Create a new experience for the user
- Reward attention to detail

New overcomes:

- Find the new conditional to trigger the new ending
- Realise the importance of the return function
- The importance of #

Escape Room

Thank You!

Juan Fran, Natalia, Javier

Let's play a game!