

# Supplementary Material for FlairGPT: Repurposing LLMs for Interior Designs

Gabrielle Littlefair<sup>1</sup> , Niladri Shekhar Dutt<sup>1</sup> , and Niloy J. Mitra<sup>1,2</sup> 

<sup>1</sup>University College London

<sup>2</sup>Adobe Research

[flairgpt.github.io](https://flairgpt.github.io)

## Contents

1. Statistics For Experiments (page 1)
2. Token Cost Comparison with I-Design [S\*24] and Holodeck [YSW\*23]. (page 2)
3. User Study I Responses (page 3)
4. User Study II Responses (page 4)
5. Human Forms for User Studies and Human Drawn Layouts (page 6)
6. Blank Constraint Cost Functions (page 9)
7. Full example language output for “a bedroom that is 4m × 5m.” (page 17)

## 1. Statistics For Experiments

We define 5 types of errors that can occur throughout our method:

- Language Error: This type of error arises purely from the output of the LLM during the language generation phase. It includes incorrect object sizing, nonsensical constraints (e.g., “put the table lamp on the armchair”), or other errors in the initial LLM output.
- Cleaning Error: These errors occur during the cleaning phase. Examples include the unintended removal of constraints or the omission of crucial information from a constraint.
- Translation Error: This is the broadest category of errors and can occur at any point during the translation phase. It may involve matching a language constraint to a similar but suboptimal constraint (e.g., selecting “away from window” instead of “not blocking a window”), completely misinterpreting the constraint, missing applicable constraints that have matching functions, or using incorrect parameters. Translation errors are the most frequent type of error.
- Contradictory Constraint Error: This error occurs when two or more constraints are chosen that are mutually exclusive, making it impossible to satisfy all of them simultaneously within the solution.
- Optimization Error: An optimization error arises when an object is placed in a position that does not align with its constraints, and yet the optimization process fails to find a better solution throughout the optimization process.

While there are many places for errors to arise, they are not all critical. For example, the most common translation error that we have seen is choosing “ind\\_away\\_from” instead of “ind\\_not\\_block” which are similar constraints and will achieve the object not blocking the window. When incorrect types of parameters are used, the function returns 0 so that constraint is lost. This can occur when choosing the sides of an object (one of “left”, “right”, “front” or “back”) with the LLM choosing something like “longer side”. The most problematic errors are the contradictory constraint errors and

**Table 1:** Statistics for our experiments including: the number of primary ( $P$ ), secondary ( $S$ ), and tertiary ( $T$ ) objects per scene; the number of constraints before cleaning, after cleaning, and after translation (function calls); the number of errors including Language errors, Cleaning errors, Translation errors, and Optimization errors; and the time (minutes) for the Language and Translation phase combined, the Optimization phase, and the total time to generate each layout.

Prompt	Objects			Constraints			Errors				Time (mins)			
	P	S	T	Uncleaned	Cleaned	Function Calls	Language	Cleaning	Translation	Contradiction	Optimization	Language + Translation	Optimization	Total
"A bedroom that is 4m x 5m."	3	4	7	49	52	57	1	2	6	0	1	0.82	7.20	8.02
"A living room that is 4m x 4m."	2	3	10	43	45	48	1	2	7	1	1	1.16	7.60	8.76
"A sewing room."	3	5	11	59	62	70	0	1	11	2	1	1.06	12.71	13.76
"A small home gym."	3	5	7	52	48	53	1	0	6	1	0	1.56	14.45	16.01
"A small green boho dining room."	3	7	9	58	65	68	1	1	24	2	0	1.05	24.35	25.41
"A traditional living room."	3	5	10	64	73	72	0	2	7	1	3	1.37	8.17	9.54
"An office for a bestselling writer in New York who likes to write Fantasy books."	3	4	11	60	62	63	1	3	4	0	1	1.08	13.04	14.12
"A bedroom that is 5x5 for a young girl who likes to paint whilst looking out of her window."	3	5	8	62	62	61	0	3	16	1	1	1.03	6.97	8.00
"A bedroom for a vampire."	3	4	9	49	47	47	0	0	0	2	2	0.85	6.68	7.53
"A small workshop for a wizard."	3	6	10	65	64	65	0	0	6	1	1	1.24	10.85	12.08
"A kitchen for an ogre."	4	10	10	72	73	79	0	7	3	2	1	1.61	12.13	13.73
Mean values	3.00	5.27	9.27	57.55	59.36	62.09	0.36	1.91	8.27	1.18	1.09	1.17	11.29	12.45

the optimization errors. These are the most visible in the outputs, however these are also far less frequent than translation errors.

### 1.1. Success Rate with Varying Numbers of Objects

In our experiments, we found that the success was similar for a room with 5 primary and secondary objects ("A living room that is 4m x 4m") and a room with 14 objects ("A kitchen for an ogre"). The tertiary objects are optimized separately and rely on a maximum of one primary or secondary object, and so larger numbers should not interfere with the success too much. Our method was designed in such a way that there is not too much variation in the number of objects generated to prevent overcrowding.

### 2. Token Cost Comparison with I-Design [S\*24] and Holodeck [YSW\*23]

Method	Number of API Calls	Total Number of Tokens
I-Design [S*24]	$23 \pm 4.58$	$35295 \pm 5355$
Holodeck [YSW*23]	7	$6440 \pm 2007$
FlairGPT (Ours)	$20.66 \pm 1.53$	$22762 \pm 2179$

### 3. User Study 1 Responses

Q1: Is this a good choice for you?				Q2: Do you feel like the colors and materials used in this interior design are appropriate?		
Good				Bad		
1	1	2	2	2	1	Bottom
2	1	2	2	2	1	Top
3	1	2	2	2	1	Bottom
4	1	2	2	2	1	Top
5	1	2	2	2	1	Bottom
6	1	2	2	2	1	Top
7	1	2	2	2	1	Bottom
8	1	2	2	2	1	Top
9	1	2	2	2	1	Bottom
10	1	2	2	2	1	Top
11	1	2	2	2	1	Bottom
12	1	2	2	2	1	Top
13	1	2	2	2	1	Bottom
14	1	2	2	2	1	Top
15	1	2	2	2	1	Bottom
16	1	2	2	2	1	Top
17	1	2	2	2	1	Bottom
18	1	2	2	2	1	Top
19	1	2	2	2	1	Bottom
20	1	2	2	2	1	Top
21	1	2	2	2	1	Bottom
22	1	2	2	2	1	Top
23	1	2	2	2	1	Bottom
24	1	2	2	2	1	Top
25	1	2	2	2	1	Bottom
26	1	2	2	2	1	Top
27	1	2	2	2	1	Bottom
28	1	2	2	2	1	Top
29	1	2	2	2	1	Bottom
30	1	2	2	2	1	Top
31	1	2	2	2	1	Bottom
32	1	2	2	2	1	Top
33	1	2	2	2	1	Bottom
34	1	2	2	2	1	Top
35	1	2	2	2	1	Bottom
36	1	2	2	2	1	Top
37	1	2	2	2	1	Bottom
38	1	2	2	2	1	Top
39	1	2	2	2	1	Bottom
40	1	2	2	2	1	Top
41	1	2	2	2	1	Bottom
42	1	2	2	2	1	Top
43	1	2	2	2	1	Bottom
44	1	2	2	2	1	Top
45	1	2	2	2	1	Bottom
46	1	2	2	2	1	Top
47	1	2	2	2	1	Bottom
48	1	2	2	2	1	Top
49	1	2	2	2	1	Bottom
50	1	2	2	2	1	Top

## 4. User Study 2 Responses

(a) Object Type Ignoring the type, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	4	4	4	3	4	4	4	4	5	3	4	4	4	4	2	2	2
(b) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	5	2	3	2	4	2	3	4	4	3	5	1	4	2	4	4	3
(c) Object Style Ignoring the type, placement, shape, and size of the objects, please consider the style (colour, material, overall style e.g. minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	4	5	4	3	3	4	2	4	1	4	4	4	5	5	3	3	5
(d) Object Functionality Ignoring the type, size, style, and placement of the objects, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down? How much do you agree with the following statement: "I am able to easily use every object in the room."	3	2	4	4	5	3	5	4	3	3	3	2	5	3	4	3	3
(e) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is set out well and the objects are placed in a way that makes sense."	3	2	2	2	4	3	4	4	2	3	4	1	3	4	2	2	4
(f) Object Type Ignoring the type, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	5	5	5	5	5	5	4	4	5	4	5	4	5	4	4	4	4
(g) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	5	5	3	4	5	5	2	4	5	5	4	5	5	5	4	4	5
(h) Object Style Ignoring the type, size, functionality and placement of the objects, please consider the style (colour, material, overall style e.g. minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	4	5	4	5	4	4	2	4	4	5	4	5	5	5	2	4	3
(i) Object Functionality Ignoring the type, size, style, and placement of the objects, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down? How much do you agree with the following statement: "I am able to easily use every object in the room."	3	4	4	4	5	2	2	2	2	5	5	0	4	5	4	3	5
(j) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is set out well and the objects are placed in a way that makes sense."	4	4	3	5	4	2	4	2	2	4	3	4	4	4	3	3	5
(k) Object Type Ignoring the type, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	4	5	4	5	5	3	4	4	2	4	5	3	4	5	3	4	4
(l) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	5	5	5	5	5	5	3	4	5	5	5	5	5	5	5	5	5
(m) Object Style Ignoring the type, size, style, and placement of the objects, please consider the style (colour, material, overall style e.g. minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	4	5	4	2	5	4	5	4	5	5	4	5	5	5	5	5	5
(n) Object Functionality Ignoring the type, size, style, and placement of the objects, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down? How much do you agree with the following statement: "I am able to easily use every object in the room."	4	5	4	5	5	5	5	3	4	5	5	5	5	5	5	5	4
(o) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is set out well and the objects are placed in a way that makes sense."	3	4	4	5	5	5	5	4	3	4	3	4	5	5	5	4	5
(p) Object Type Ignoring the type, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	4	5	5	4	5	5	3	4	5	4	5	5	4	5	3	4	5
(q) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	5	5	5	5	5	5	3	4	5	5	5	5	5	5	5	5	5
(r) Object Style Ignoring the type, size, style, and placement of the objects, please consider the style (colour, material, overall style e.g. minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	4	5	5	5	5	5	2	4	4	5	5	5	5	5	3	5	4
(s) Object Functionality Ignoring the type, size, style, and placement of the objects, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down? How much do you agree with the following statement: "I am able to easily use every object in the room."	4	5	4	5	5	5	5	3	4	5	5	5	5	5	5	5	4
(t) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is set out well and the objects are placed in a way that makes sense."	3	4	3	5	4	5	3	4	5	5	5	4	5	5	3	5	2
(u) Object Type Ignoring the type, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	4	5	5	5	5	5	5	4	3	4	5	5	5	5	5	5	5
(v) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	4	5	3	2	3	2	3	2	3	4	3	4	5	5	2	4	5
(w) Object Style Ignoring the type, size, style, and placement of the objects, please consider the style (colour, material, overall style e.g. minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	5	5	4	4	5	5	3	3	5	5	5	4	4	5	5	4	5
(x) Object Functionality Ignoring the type, size, style, and placement of the objects, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down? How much do you agree with the following statement: "I am able to easily use every object in the room."	4	5	4	4	5	5	2	4	3	2	4	5	5	5	3	2	5
(y) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is set out well and the objects are placed in a way that makes sense."	4	5	4	4	5	5	3	3	2	2	2	5	5	5	5	5	4
(z) Object Type Ignoring the type, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	4	5	3	2	4	4	3	1	3	4	1	3	3	4	5	2	4
(aa) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	5	4	4	4	5	3	4	4	4	3	1	4	4	5	5	4	4
(bb) Object Style Ignoring the type, size, style, and placement of the objects, please consider the style (colour, material, overall style e.g. minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	4	4	4	4	5	3	5	2	4	3	2	4	5	5	3	2	5
(cc) Object Functionality Ignoring the type, size, style, and placement of the objects, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down? How much do you agree with the following statement: "I am able to easily use every object in the room."	4	5	4	5	5	5	3	3	2	2	2	5	5	5	5	5	4
(dd) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is set out well and the objects are placed in a way that makes sense."	4	3	4	4	5	3	1	4	2	1	0	5	5	5	3	2	5

(a) Object Type Ignoring the style, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	5	5	4	4	5	5	4	4	4	4	5	5	5	5	4	4	4	4	4
(b) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	4	5	4	4	5	5	4	4	4	4	5	5	5	5	4	5	3	2	2
(c) Object Style Ignoring the type, placement, shape, and size of the objects, please consider the style (colour, material, overall style e.g. minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	5	5	5	5	4	5	5	4	5	4	5	5	5	5	4	5	4	4	4
(d) Object Functionality Ignoring the type, size, style, and placement, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down?	4	5	4	5	5	5	4	4	4	4	5	5	5	5	4	4	4	4	3
(e) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is laid out well and the objects are placed in a way that makes sense."	4	5	3	5	5	5	4	4	3	5	4	4	5	4	4	4	4	4	4
(f) Object Type Ignoring the style, placement, shape, and size of the objects, please consider the type (e.g. bed or desk) of each object in the room. How much do you agree with this statement: "Given the description of the room, each object in the room should be there, and there are no objects in the room that should not be there."	5	5	4	5	5	4	4	3	5	5	5	4	5	5	3	4	4	4	4
(g) Object Size Ignoring the type, style, functionality, and placement of the objects, please consider the size of each object. How much do you agree with this statement: "The objects in the room are the correct size and in proportion."	3	4	2	4	5	2	3	2	4	5	5	3	5	4	4	4	4	4	3
(h) Object Style Ignoring the type, placement, shape, and size of the objects, please consider the style (colour, material, overall style e.g minimalist) of each object in the room. How much do you agree with this statement: "The style of the objects in the room is cohesive and matches the room description well."	4	5	4	5	5	5	4	3	5	4	4	5	5	5	4	5	4	4	4
(i) Object Functionality Ignoring the type, size, style, and placement, please consider the functionality of all the objects in the room. Can the wardrobe doors open? Can you walk to the sofa and sit down?	4	5	3	5	5	2	4	3	2	4	5	5	5	3	5	3	4	4	3
(j) Object Placement Ignoring the type, size, style, and functionality of the objects, please consider the placement of all the objects in the room. How much do you agree with the following statement: "The room is laid out well and the objects are placed in a way that makes sense."	4	4	3	5	5	3	4	3	2	5	5	4	5	3	3	4	4	3	2

## 5. Human Forms for User Studies and Human Drawn Layouts

**Example: "A living room."**

- Arrows indicate the front side of corresponding objects  
- Reference grid lines are spaced to be 1 meter apart

**Key**

- Socket
- Door
- Window

**M = Must Have, S = Should Have, N = Nice to Have**

1. Sofa (M)
2. Coffee Table (S)
3. TV Stand (S)
4. Armchair (S)
5. TV (M)
6. Painting (N)

**"A bedroom that is 4m x 5m."**

Please add up to 15 objects into the room. Label each object with a number and fill out the spaces below with the name of the object. Please follow the name with either M, S, or N.

M = Must Have  
S = Should Have  
N = Nice to Have

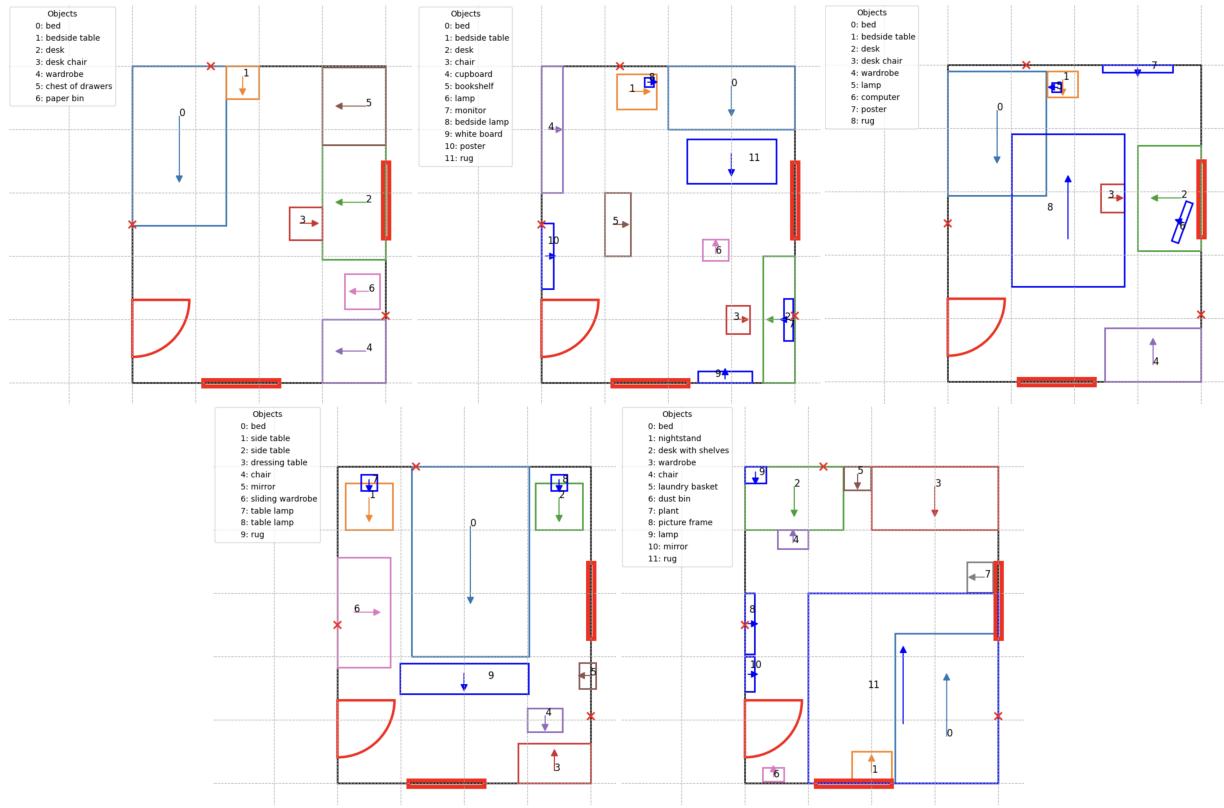
- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.
- 11.
- 12.
- 13.
- 14.
- 15.

**"An office for a bestselling writer in New York who likes to write Fantasy books."**

Please add up to 20 objects into the room. Label each object with a number and fill out the spaces below with the name of the object. Please follow the name with either M, S, or N.

M = Must Have  
S = Should Have  
N = Nice to Have

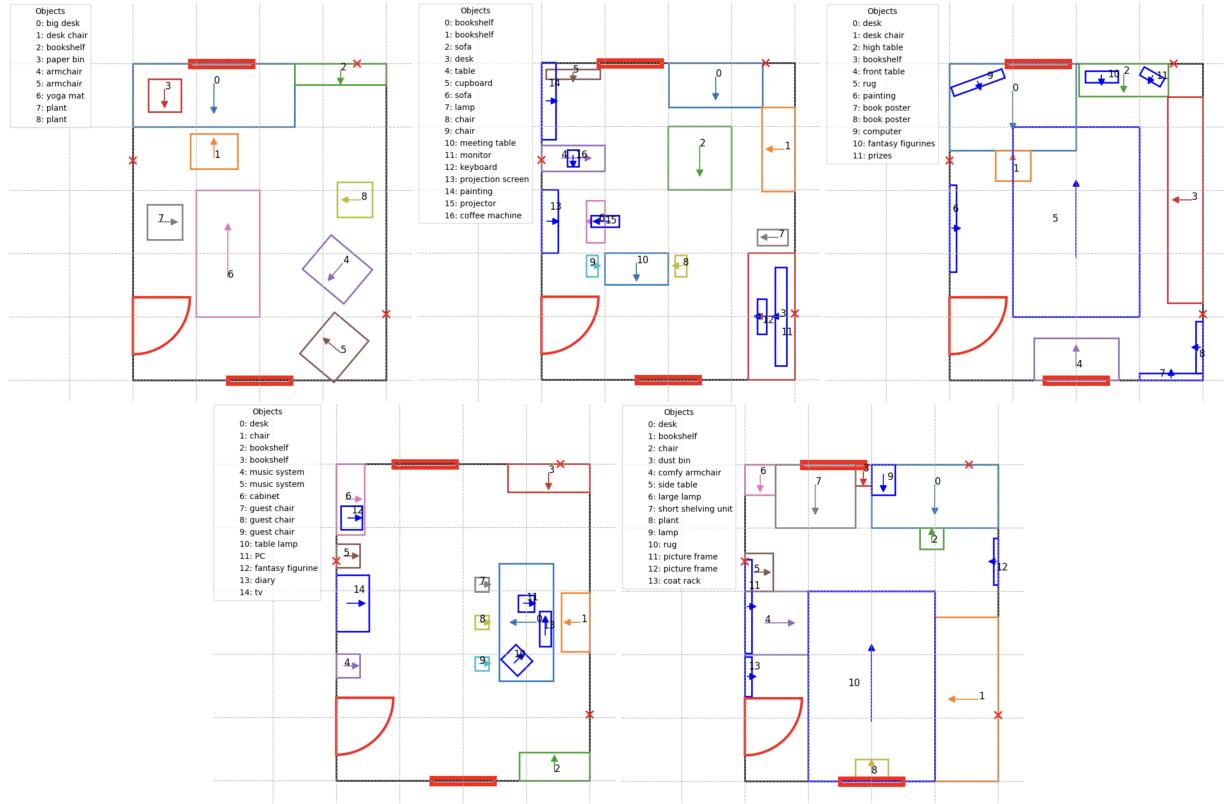
- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.
- 11.
- 12.
- 13.
- 14.
- 15.
- 16.
- 17.
- 18.
- 19.
- 20.



**Figure 1:** Layouts designed by 5 novice human designers for the prompt: “a bedroom that is 4m x 5m.”



**Figure 2:** Two layouts chosen from Figure 1, rendered in Blender [Ble18], using assets from BlenderKit [Ble24].



**Figure 3:** Layouts designed by 5 novice human designers for the prompt: “an office for a bestselling writer in New York who likes to write Fantasy books.”



**Figure 4:** Two layouts chosen from Figure 3, rendered in Blender [Ble18], using assets from BlenderKit [Ble24].

## 6. Blank Constraint Cost Functions

### 1 Blank Constraint Cost Functions for Individual Constraints

---

```

1  from Class_Structures import *
2  from shapely.geometry import Polygon
3
4  """Throughout, the sides of the objects are defined as follows:
5  # 'back' of the object would be the headboard of a bed, or the back of a chair
6  # 'front' of the object would be the foot of a bed, or the front of a wardrobe (the side with the doors)
7  # 'left' would be the left side of the object, when standing behind it
8  # 'right' would be the right side of the object, when standing behind it
9
10 def ind_next_to_wall(positions, room, object_index):
11     """This function ensures an object is next to a wall in a room. Specifically the back of the object.
12     Example constraint: "The tv should be against a wall."
13
14     Args:
15         positions: list of floats, x, y, theta values for all objects in the room
16         room: rectangular Room object
17         object_index: int, index of the object in the room's object list
18
19     """
20
21     return
22
23
24 def ind_near_wall(positions, room, object_index, side = 'back', max_dist = 0.5):
25     """This function ensures an object is near to a wall in a room (within a specific distance, NOT next to).
26     The specific side of the object will be used. If no side is given, the back of the object will be used.
27
28     Args:
29         positions: list of floats, x, y, theta values for all objects in the room
30         room: rectangular Room object
31         object_index: int, index of the object in the room's object list
32         side: string, one of 'back', 'front', 'left', 'right' defines which side of the object to check e.g back of bed
33         max_dist: float, maximum distance the object should be from the wall
34
35
36
37     return
38
39 def ind_close_to_fixed_object(positions, room, object_index, fixed_object_type, side = None, max_dist = 0.5):
40     """The function ind_close_to_fixed_object is used for 3 purposes:
41         1) an object should be next to a window (fixed_object_type = 'window')
42         2) an object should be next to a door (fixed_object_type = 'door')
43         3) an object should be next a socket (fixed_object_type = 'socket')
44
45     If side is given, the specific side of the object will be used.
46
47     Args:
48         positions: list of floats, x, y, theta values for all objects in the room
49         room: rectangular Room object
50         object_index: int, index of the object in the room's object list
51         fixed_object_type: string, type of fixed object to check. One of 'window', 'door', 'socket'
52         side: string, one of 'back' (for things like headboard of bed, or back of bookshelf), 'front'
53             (for things like foot of bed or front of bookshelf), 'left', 'right', defines which side of the object to check
54             e.g back of bed
55         max_dist: float, maximum distance between the object and the fixed object to be considered close to it.
56         Write this as a float e.g. 2.0.
57
58
59     return
60
61 def ind_away_from_fixed_object(positions, room, object_index, fixed_object_type, min_dist = 2.0):
62     """This function is used for 3 purposes:
63         1) an object should be away from a window (fixed_object_type = 'window')

```

```

64     2) an object should be away from a door (fixed_object_type = 'door')
65     3) an object should be away from a socket (fixed_object_type = 'socket')
66
67     Args:
68     positions: list of floats, x, y, theta values for all objects in the room
69     room: rectangular Room object
70     object_index: int, index of the object in the room's object list
71     fixed_object_type: string, type of fixed object to check. One of 'window', 'door', 'socket'
72     min_dist: float, minimum distance between the object and the fixed object to be considered away from it.
73     Write this as a float, e.g. 2.0.
74
75     """
76     return
77
78 def ind_accessible(positions, room, object_index, sides, min_dist = None):
79     """ This function ensures that an object is accessible from given sides. It can also ensure that
80         there is nothing too close to a given side of an object (e.g. if there needs to be clearance around something).
81         If no sides are given, the front side is used. If min_dist is given, then this function
82         will act as a clearance constraint. If you want all the sides to be accessible,
83         sides = ['front', 'back', 'left', 'right'].
84
85     Args:
86     positions: list of floats, x, y, theta values for all objects in the room
87     room: rectangular Room object
88     object_index: int, index of the object in the room's object list
89     sides: a list of strings, each one one of 'front', 'left', 'right', defines which side of the object to check
90     min_dist: float (optional), minimum distance clearance for the object on the sides given.
91     Write this as a float, e.g. 1.0.
92
93     """
94
95     return
96
96 def ind_central(positions, room, object_index, both = False):
97     """ This function ensures that an object is centrally placed in the room.
98     Args:
99     positions: list of floats, x, y, theta values for all objects in the room
100    room: rectangular Room object
101    object_index: int, index of the object in the room's object list
102    both: bool (optional), if True, then the object should be placed centrally in both x and y. For example for a bed,
103        but for a dining table, it should be True.
104
105
106     return
107
108
109 def ind_not_block_fixed_object(positions, room, object_index, fixed_object_type):
110
111     """ This function is used for 2 purposes:
112         1) an object does not block a window (fixed_object_type = 'window')
113         2) an object does not block a door (fixed_object_type = 'door')
114
115     Args:
116     positions: list of floats, x, y, theta values for all objects in the room
117     room: rectangular Room object
118     object_index: int, index of the object in the room's object list
119     fixed_object_type: string, type of fixed object to check. E.g one of 'window', 'door', 'plug'
120
121
122     return
123
124 def ind_under_window(positions, room, object_index):
125
126     """ This function ensures that the object will be placed underneath a window.
127     For example, you might want a desk or a dresser below (but not blocking) a window. You would NOT use this for any
128     objects that would be tall, for example a wardrobe or a fridge. Example constraint "The desk should be under
129     the window", "The desk should look out the window". Don't use with ind_not_block_fixed_object for a window

```

```

130     and the same object.
131
132     Args:
133         positions: list of floats, x, y, theta values for all objects in the room
134         room: rectangular Room object
135
136     return
137
138 def ind_facing_into_room(positions, room, object_index):
139     """ ind_facing_into_room is a function that ensures and object faces into the center of the room.
140     E.g. an armchair might face into the room.
141
142     Args:
143         positions: list of floats, x, y, theta values for all objects in the room
144         room: rectangular Room object
145         object_index: int, index of the object in the room's object list
146
147     """
148
149     return
150
151 def ind_in_region(positions, room, object_index, region_name, weight = 5.0):
152     """ This function ensures that an object is in a given region. This should NOT be used with the
153     optimize_primary_objects function.
154
155     Args:
156         positions: list of floats, x, y, theta values for all objects in the room
157         room: rectangular Room object
158         object_index: int, index of the object in the room's object list
159         region_name: string, name of the region for the object to be in
160         weight: float, weight of the constraint
161
162     """
163
164     return
165
166 def ind_not_against_wall(positions, room, object_index, min_dist = 0.5):
167
168     """ ind_not_against_wall is a function that ensures an object is not against a wall.
169     For example "the rug should not be touching the wall" or "the dining table should not be against the wall".
170
171     Args:
172         positions: list of floats, x, y, theta values for all objects in the room
173         room: rectangular Room object
174         object_index: int, index of the object in the room's object list
175         min_dist: float, minimum distance the object should be from the wall. Please write this as a float, e.g. 2.0.
176
177     """
178
179     return
180
181 def ind_in_corner(positions, room, object_index, side = 'back', max_dist = 0.5):
182     """ This function can be used to ensure that an object is placed into a corner.
183     The back of the object will always be placed closest to the corner.
184
185     Args:
186         positions: list of floats, x, y, theta values for all objects in the room
187         room: rectangular Room object
188         object_index: int, index of the object in the room's object list
189         max_dist: float, maximum distance the object should be from the wall
190
191     """
192
193     return

```

---

## 2 Blank Constraint Cost Functions for Inter-Object Constraints

---

```
1 import numpy as np
2 from Class_Structures import *
3 from shapely.geometry import Polygon
4
5     ### Throughout, the sides of the objects are defined as follows:
6     # 'top' or 'back' of the object would be the headboard of a bed, or the back of a chair
7     # 'front' or 'bottom' of the object would be the foot of a bed, or the front of a wardrobe (the side with the doors)
8     # 'left' would be the left side of the object, when standing behind it
9     # 'right' would be the right side of the object, when standing behind it
10
11
12 def io_next_to(positions, room, object1_index, object2_index, side1 = None, side2 = None):
13     """DO NOT USE THIS WITH io_surround, IT WILL BE REDUNDANT OR CONTRADICTORY.
14     This function ensures that two objects are next to each other in a room.
15     This should only be used when necessary e.g. for nightstands and a bed, or a desk and desk chair.
16     This should not be used for dining chairs around a table or similar relationship, for that use io_surround.
17     If side1 is given, the specific side of object1 will be used. If side2 is given,
18     the specific side of object2 will be used. E.g. the 'front' of the chair should be next to the 'front' of the desk.
19     If no side is given, then any of the sides will be used.
20
21     Args:
22         room: rectangular Room object
23         object1: Object object
24         object2: Object object
25         side1: string, one of 'top' or 'back', 'bottom' or 'front', 'left', 'right', defines which side of object1 to use
26         side2: string, one of 'top' or 'back', 'bottom' or 'front', 'left', 'right', defines which side of object2 to use
27     """
28
29     return
30
31 def io_away_from(positions, room, object1_index, object2_index, min_dist = 2.0):
32     """This function ensures that two objects are away from each other in a room.
33     For example, a bed should be away from a desk.
34
35     Args:
36         positions: list of floats, x, y, theta values for all objects in the room
37         room: rectangular Room object
38         object1_index: int, index of object1 in the room
39         object2_index: int, index of object2 in the room
40         min_dist: float, minimum distance between the two objects. Please write this as a float, e.g. 2.0.
41
42     """
43
44     return
45
46 def io_near(positions, room, object1_index, object2_index, max_dist = 3.0):
47     """This function ensures that two objects are within a certain distance to each other.
48     They are not necessarily next to each other, but they are close. This might be for a bookshelf
49     and an armchair, or a mirror and a wardrobe.
50
51     Args:
52         room: rectangular Room object
53         object1_index: Object object
54         object2_index: Object object
55         max_dist: furthest distance between the two objects. Please write this as a float, e.g. 3.0.
56
57     """
58
59 def io_parallel(positions, room, object1_index, object2_index, center_object_info = None, max_dist = 2.0):
60     """This function ensures that two objects have the same orientation in a room.
61     That is, that they are parallel to each other. It does not handle distance, so if
62     proximity is important, please combine this function with io_near, or io_next_to, or even io_between.
63
```

```

64     Args:
65     positions: list of floats, x, y, theta values for all objects in the room
66     room: rectangular Room object
67     object1_index: int, index of object1 in the room
68     object2_index: int, index of object2 in the room
69     """
70     return
71
72 def io_facing(positions, room, object1_index, object2_index, both = False):
73     """ This function ensures that object1 is facing object2 in a room.
74     If both is True, then object2 will also be facing object1.
75     For example, a sofa and tv should face each other, so in that instance both would be True.
76
77     Args:
78     positions: list of floats, x, y, theta values for all objects in the room
79     room: rectangular Room object
80     object1_index: int, index of object1 in the room
81     object2_index: int, index of object2 in the room
82     both: bool, if True, object2 will also be facing object1
83     """
84     return
85
86 def io_infront(positions, room, object1_index, object2_index, dist = 0.8, parallel = False):
87     """ This function ensures that object1 is in front of object2 (both moving_objects i.e. not windows or doors).
88     E.g a coffee table should be in front of a sofa.
89
90     Args:
91     positions: list of floats, x, y, theta values for all objects in the room
92     room: rectangular Room object
93     object1_index: int, Object object
94     object2_index: int, Object object
95     dist: float, desired distance between two objects. E.g. if its a sofa and a coffee table, the distance should
96     be around 0.8m, if its a sofa and a fireplace, the distance should be around 2m/2.5m.
97     parallel: bool, if True, object1 will be parallel to object2. This would be used for a coffee table in front
98     of a sofa, but not for a sofa in front of a fireplace.
99     """
100    return
101
102 def io_perp(positions, room, object1_index, object2_index, center_object_index = None):
103     """ This function ensures that two objects are aligned in a room perpendicularly.
104     If center is given, the objects will be aligned about that point. For example, a sofa and chair might be aligned
105     perpendicularly about a coffee table or a side table.
106
107     Args:
108
109     positions: list of floats, x, y, theta values for all objects in the room
110     room: rectangular Room object
111     object1_index: int, index of object1 in the room
112     object2_index: int, index of object2 in the room
113     center_object_index: int, index of object in room.moving_objects to be used as the pivot for the alignment
114     (e.g. a coffee table or a table)
115     """
116
117    return
118
119 def io_surround(positions, room, central_object_index, object_indices):
120     ## IF YOU USE THIS FUNCTON, DO NOT ALSO USE "IO_NEXT_TO" AS IT WILL BE REDUNDANT OR CONTRADICTORY.
121     """ This function ensures that central_object is surrounded by all the objects in object_indices.
122     This would be used for chairs around a dining table. This should NOT be combined with io_next_to,
123     as that would be redundant or
124     it would contradict.
125
126     Args:
127     positions: list of floats, x, y, theta values for all objects in the room
128     room: rectangular Room object
129

```

```

130     central_object_index: int, Object object
131     object_indices: list of ints, indices of Object objects
132     """
133
134     return
135
136 def io_not_facing(positions, room, object1_index, object2_index):
137     """ This function ensures that object1 is NOT facing object2 in a room.
138     For example, a bed should not face a mirror.
139     Args:
140         positions: list of floats, x, y, theta values for all objects in the room
141         room: rectangular Room object
142         object1_index: int, index of object1 in the room
143         object2_index: int, index of object2 in the room
144     """
145
146     return
147
148 def io_between(positions, room, object1_index, object2_index, object3_index):
149     """ This function ensures that object1 is in between the two objects object2 and object3.
150     This would be used for something like a side table being between two chairs, or maybe a bed being
151     between two nightstands. Or a nightstand going between two beds.
152
153     Args:
154         positions: list of floats, x, y, theta values for all objects in the room
155         room: rectangular Room object
156         object1_index: int, index of object1 in the room (** this is the object that will go
157             in between the other two objects)
158         object2_index: int, index of object2 in the room
159         object3_index: int, index of object3 in the room
160     """
161
162     return

```

---

### 3 Blank Constraint Cost Functions for Tertiary Constraints

```

1 ## All the Individual Object constraint functions are defined here
2 from Class_Structures import *
3 from shapely.geometry import Polygon, Point
4 from shapely import distance
5 from Individual import *
6 from Setup_Functions import *
7 from Global import *
8 from scipy.optimize import minimize
9
10
11 def rug_under_central(positions, room, rug_index, object_index):
12
13     """ This function ensures that the rug is placed under the central object.
14
15     Args:
16         positions: list of floats, x, y, theta values for all objects in the room
17         room: rectangular Room object
18         rug_index: int, index of the rug object in the room.tertiary_objects list
19         object_index: int, index of the central object in the room.moving_objects list
20     """
21
22     return
23
24 def rug_under_central_forward(positions, room, rug_index, object_index):
25
26     """ This function ensures that the rug is placed under the central object, oriented correctly,
27     and moved slightly forward. E.g. this would be used for a rug that is placed under a bed.
28

```

```

29     Args:
30     positions: list of floats, x, y, theta values for all objects in the room
31     room: rectangular Room object
32     rug_index: int, index of the rug object in the room.tertiary_objects list
33     object_index: int, index of the central object in the room.moving_objects list
34
35     """
36     return
37
38 def on_top_central(positions, room, tertiary_index, other_index):
39
40     """ This function ensures that the tertiary object is placed on top of the central object.
41     This would be used for placing a table lamp on top of a nightstand, for example.
42
43     Args:
44         positions: list of floats, x, y, theta values for all objects in the room
45         room: rectangular Room object
46         tertiary_index: int, index of the tertiary object in the room.tertiary_objects list
47         other_index: int, index of the central object in the room.moving_objects list
48
49     """
50     return
51
52 def on_top_corner(positions, room, tertiary_index, other_index, corner = 'tl'):
53
54     """ This function ensures that the tertiary object is placed on top of the central object, at a specific corner.
55     This would be used for placing a lamp on the top left corner of a desk, for example.
56
57     Args:
58         positions: list of floats, x, y, theta values for all objects in the room
59         room: rectangular Room object
60         tertiary_index: int, index of the tertiary object in the room.tertiary_objects list
61         other_index: int, index of the central object in the room.moving_objects list
62         corner: str, corner of the central object where the tertiary object should be placed.
63         Options are 'tl', 'tr', 'bl', 'br'
64
65     """
66     return
67
68 def on_wall_near(positions, room, tertiary_index, other_index):
69
70     """ This function ensures that the tertiary object is placed on the wall near the central object.
71     This would be used for placing a painting on the wall near a dining table, for example.
72
73     Args:
74         positions: list of floats, x, y, theta values for all objects in the room
75         room: rectangular Room object
76         tertiary_index: int, index of the tertiary object in the room.tertiary_objects list
77         other_index: int, index of the central object in the room.moving_objects list
78
79     """
80     return
81
82 def on_wall_in_region(positions, room, tertiary_index, region_name):
83
84     """ This function is used to place a tertiary object on the wall in a specific region of the room.
85     This would be used for placing a painting on the wall in the living room, for example.
86
87     Args:
88         positions: list of floats, x, y, theta values for all objects in the room
89         room: rectangular Room object
90         tertiary_index: int, index of the tertiary object in the room.tertiary_objects list
91         region_name: str, name of the region where the tertiary object should be placed
92         (e.g. 'living', 'dining', 'bedroom')
93
94     """
95     return
96
97 def center_ceiling(positions, room, tertiary_index):
98
99

```

```
95     """ This function ensures that the tertiary object is placed in the center of the ceiling.  
96     This would be used for placing a chandelier/ceiling light in the center of the room.  
97  
98     Args:  
99         positions: list of floats, x, y, theta values for all objects in the room  
100        room: rectangular Room object  
101        tertiary_index: int, index of the tertiary object in the room.tertiary_objects list  
102    """  
103    return  
104  
105 def ceiling_above(positions, room, tertiary_index, other_index):  
106     """ This function ensures that the tertiary object is placed on the ceiling above the central object.  
107     This would be used for placing a ceiling fan above a bed, or a chandelier above a dining table for example.  
108  
109     Args:  
110         positions: list of floats, x, y, theta values for all objects in the room  
111        room: rectangular Room object  
112        tertiary_index: int, index of the tertiary object in the room.tertiary_objects list  
113        other_index: int, index of the central object in the room.moving_objects list  
114    """  
115    return  
116
```

---

## 7. Full example for “A bedroom that is 4m x 5m.”

**Input:** “A bedroom that is 4m x 5m.”

### Extracting Parameters

**Prompt 1:** I am designing the layout for a bedroom that is 4m x 5m. Give me the dimensions of the room, as well as the number of windows, doors, and sockets. For each of these, tell me the dimensions in meters, which wall it should be on, as well as where on the wall it should be (a value between 0 and 1 determining how far along the wall it should be). Here is an example for a room and window set up:

{Room: width = 5, length = 6}, {Object type: window, width = 1, length = 0.1, wall = south, position = 3}. The windows and doors should all have length 0.1. Ensure that every object has dimensions, position, and wall. No other text please. Just the list with no commentary or explanation.

**Prompt 2:** The functions in the following file are used to create a room and out objects in it: + file\_contents1. The room is described as “a bedroom that is 4m x 5m”. For every element in this list, fixed\_objects, translate it into a function call using the file attached.

For example, given the room descriptor ‘a small kitchen’ as well as the list: {Room: width = 5, length = 6}, {Object type: window, Number: 1, width = 1, length = 0.1, wall = south, position = 0.3}, the output would be:

kitchen = create\_room(5, 6), create\_fixed\_object(kitchen, ‘window’, 1, 0.1, ‘south’, position = 0.3). Return only the python script.

No other text. Please ensure the correct number of inputs are given for each function.

#### **Output 1:**

```
room = create_room(4, 5)
create_fixed_object(room, 'door', 0.9, 0.1, 'west', 0.1)

create_fixed_object(room, 'window', 1.2, 0.1, 'south', 0.4)
create_fixed_object(room, 'window', 1.2, 0.1, 'east', 0.6)
create_fixed_object(room, 'socket', 0.2, 0.1, 'north', 0.3)
create_fixed_object(room, 'socket', 0.2, 0.1, 'west', 0.5)
create_fixed_object(room, 'socket', 0.2, 0.1, 'east', 0.2)
```

### Regions

**Prompt1:** I am designing the layout of a “a bedroom that is 4m x 5m”, that is 4m x 5m. Interior designers suggest that a good layout will have different regions in the room for different functions. What regions should I use, and how many regions (minimum is 1 region - small rooms, e.g. 3x3, 2x3, etc, should only have 1 region, small medium sized rooms will have 2, 3x4, 4x3, 4x4, medium will have 3, etc) should there be based on the description, as

well as the size of the room. Make sure to not add too many regions as this causes overcrowding (very few rooms will have more than 4 regions). Return this as a simple bullet list of regions to include. Return these in list of priority order. Do NOT add too many regions. Only return the list of regions. No other information please.

**Output:** ['sleeping', 'storage', 'work']

**Prompt 2:** Given the room name: room\_name , and the regions: regions, using the function region\_setup defined below, initialise each region. For example 'eating region' might output 'region\_setup(kitchen, 'eating', 0). Where possible the region name ('eating') should be only one word. No other text please, only the python script. Don't include the function in the output. The function is:

```
def region_setup(room_name, name, index):
```

"A function that initialises the regions in a room randomly.

Inputs:

room\_name: Room obj (this should be the name of the room)

name: str, the name of the region e.g 'sleeping' index: int, the index of the region (0, 1, 2, ...). First one must be 0, and the rest must be in order.

""

```
return
```

## Primary Objects, Secondary Objects

**Prompt 1:** The room is described as "a bedroom that is 4m x 5m". Now for each one of the regions: regions, what is the most important object to include (keep in mind the room description), and what are its dimensions (give width of back of object and length of side of object but not height) in meters. ONLY one object per region. Don't include any objects that go on the walls, e.g. wall art, and don't include any rugs/mats. The size of the room is 4m x 5m, bear this in mind when choosing the objects and the size of the objects (i.e. put small objects in small rooms). Give no other information please.

**Output:**

- Sleeping: Bed, 1.5m x 2m
- Storage: Wardrobe, 1.2m x 0.6m
- Work: Desk, 1.2m x 0.6m

**Prompt 2:** Given a room described as: "a bedroom that is 4m x 5m", with size: 4m x 5m with these regions within it: regions, and these primary objects already in it: primary\_objects, what other objects should be added in?

Give me 0-4 objects per region (depending on the size of the room - fewer objects for smaller rooms, and more objects for larger room) that should be added into the room (so make sure they are the most appropriate/necessary objects for their regions). Ensure to add objects that are necessary,e.g. if there's a desk, always add a desk chair. Try not to give objects that are redundant, e.g. if there is a tv stand in the room, you don't need a media console. Objects should be things like furniture or plants or other objects that sit on the floor.For each one,

make sure that they make sense to go into one of the regions, and tell me which region they should go into. There can be more than one of each object kind (eg. multiple dining chairs or multiple nightstands) so make sure to specify how many of each object. Make sure to also give the dimensions of every object in meters (width of the back of the object and length of the side of the object but NOT HEIGHT). Remove any objects that are not placed directly on the floor, e.g. table lamp, and remove any objects that are mounted on the wall (e.g. paintings). Don't include rugs. Only give objects that get placed onto the floor. Give no other text in the response. Only the list of objects.

**Output:**

- Sleeping:
  - Nightstand, 0.5m x 0.4m (2 items)
- Storage:
  - Dresser, 1.0m x 0.5m
- Work:
  - Desk Chair, 0.6m x 0.6m

## Primary Objects, Secondary Objects Constraints

**Prompt1:** For each of these objects: primary\_objects, tell me all of the considerations and constraints for its placement within the room described as “a bedroom that is 4m x 5m”, with size: 4m x 5m that depend only on fixed features in the room like walls, windows, doors, sockets, etc. and return these as a bullet list for each object. Include practical things like whether it should be against a wall, or which side should be accessible for use (most objects will need an accessible constraint e.g. front of wardrobe needs to be accessible - however very few objects need all of their sides to be accessible and very few need their back to be accessible.) or if its tall, maybe it shouldn't block a window, etc. Bear in mind the size of the room, for example if the room is large, then the front, left, and right of the bed should be accessible, but if the room is very small, maybe only one side of the bed needs to be accessible. Only give these constraints and considerations, no other information.

**Output:**

- ### Bed (1.5m x 2m)
  - Should be placed against a wall to maximize space.
  - Should not block access to windows.
  - Should not block access to doors.
  - Front side should be accessible.
  - Should allow for at least 60 cm of free space on one side for ease of getting in and out.
  - Avoid placing under low-hanging fixtures if the ceiling is not uniformly high.

### Wardrobe (1.2m x 0.6m)

- Should be placed against a wall for stability.
- Should not block access to windows.
- Should not block access to doors.
- Front side should be accessible for opening doors/drawers.
- Avoid placing it where it will block electrical sockets, unless not in use.

- Ideally placed in a corner to save space.

#### ### Desk (1.2m x 0.6m)

- Should be placed against a wall to maximize space.
- Should not block access to windows to ensure sufficient natural light.
- Should not block access to doors.
- Front side should be accessible for seating and working.
- Ideally positioned near electrical sockets for easy access to power supply.
- Should not block airflow from vents or radiators if present.

**Prompt 2:** Tell me all of the constraints and considerations between the objects in this list, primary\_objects, that depend only on each other. These objects are for different regions, so the constraints should only really be about them being away from each other or near each other, nothing more specific. For example, maybe a desk should not be close to a bed, etc. The room is described as "a bedroom that is 4m x 5m", with size 4m x 5m. Only give the constraints and considerations between objects, no other information.

#### Output:

1. The bed should not be close to the desk.
2. The bed should not be close to the wardrobe.
3. The desk should have enough space around it to allow for comfortable work.
4. The wardrobe should be accessible without obstruction by the bed or desk.
5. Ensure there is sufficient walking space between the bed, desk, and wardrobe.

#### **Prompt 3:**

For each of these objects, secondary\_objects, tell me all of the considerations and constraints for its placement within the room that depends only on fixed features in the room like walls, windows, doors, sockets, etc. (return these as a bullet list for each object). Include practical things like whether it should be against a wall and what side of the object (one of: left, right, back, front) or if it should be against a wall, or which side should be accessible for use (Most objects will have an accessible constraint - this would be something like the wardrobe needs to be accessible from the front so that the doors can open, or the bed needs to be accessible from the sides - but only give necessary accessible constraints. If an object can have its side against something, then don't say its side needs to be accessible e.g. for a nightstand, etc). The room is described as "a bedroom that is 4m x 5m". Don't include any constraints that depend on other objects. Only give these constraints and considerations, no other information.

#### Output:

- #### ### Sleeping:
- #### #### Nightstand (0.5m x 0.4m) (2 items)
- Can be placed against a wall.

#### ### Storage:

- #### #### Dresser (1.0m x 0.5m)
- Should be placed against a wall.
  - Front should be accessible for use.

### Work:

#### Desk Chair (0.6m x 0.6m)

- Should not block any doors or windows.
- Should be near a wall socket for power if needed (e.g., for a computer).
- Should have enough space around it to pull out and sit down (at least 0.6m of free space in front).

**Prompt 4:**

Tell me all of the constraints and considerations between the objects in this list, secondary\_objects, as well as in this list primary\_objects. For example, a desk chair should have its front against the front of the desk, or the left side of one of the nightstands should be against the right side of the bed, etc. Be specific with relationships between objects to include sides (one of: left, right, back, front) if appropriate, or minimum/maximum distances between objects etc. The room is described as "a bedroom that is 4m x 5m". Only give the constraints and considerations between objects, no other information.

**Output:**

1. \*\*Bed and Nightstands:\*\*

- The left side of one nightstand should be against the right side of the bed.
- The right side of the other nightstand should be against the left side of the bed.
- The nightstands should be positioned so they are aligned with the headboard of the bed.

2. \*\*Desk Chair and Desk:\*\*

- The front of the desk chair should be against the front of the desk.
- Ensure there is adequate space behind the desk chair for movement (minimum of 0.5m).

3. \*\*Dresser and Wardrobe:\*\*

- The dresser and wardrobe should be positioned in a way that allows for ease of access.
- Maintain at least 0.5m of space in front of both the dresser and wardrobe for access.
- The dresser and wardrobe should not be placed next to each other to avoid crowding.

4. \*\*Bed and Dresser:\*\*

- Maintain a minimum distance of 0.5m between the bed and the dresser for ease of movement and access.

5. \*\*Bed and Wardrobe:\*\*

- Maintain a minimum distance of 0.5m between the bed and the wardrobe for ease of movement and access.

6. \*\*Desk and Bed/Nightstands:\*\*

- The desk should be positioned so there is a minimum distance of 0.5m from the bed and nightstands to avoid obstruction.
- The desk should be placed along a wall opposite to or adjacent to the bed to avoid obstruction.

7. \*\*Desk and Other Storage (Dresser/Wardrobe):\*\*

- The desk should have a minimum distance of 0.5m from the dresser and wardrobe to allow for ease of movement and access to storage items.

**\*\*Note:\*\* Ensure that all furniture placement considers the need for adequate circulation space and ease of access to various parts of the room.**

**Prompt 5:**

Given this list of primary objects: primary\_objects, and this list of secondary objects: secondary\_objects, use the file attached to create the objects with the correct parameters. The room name is: room\_name the region names are: regions.

The room is already set up, only add in the objects using the 'create\_moving\_object' function. Ensure that each objects index is unique and that the indices begin from 0. The objects should be added in the correct regions. Add in all of the primary objects first. This is the file: file\_contents1. No extra text, only the function calls. Don't have 'python' at the start of the code. Do not define ANY functions, only call them.

**Output:**

```
create_moving_object(room_name, 'bed', 1.5, 2.0, 'sleeping', 0)
create_moving_object(room_name, 'wardrobe', 1.2, 0.6, 'storage', 1)
create_moving_object(room_name, 'desk', 1.2, 0.6, 'work', 2)
create_moving_object(room_name, 'nightstand', 0.5, 0.4, 'sleeping', 3)
create_moving_object(room_name, 'nightstand', 0.5, 0.4, 'sleeping', 4)
create_moving_object(room_name, 'dresser', 1.0, 0.5, 'storage', 5)
create_moving_object(room_name, 'desk chair', 0.6, 0.6, 'work', 6)
```

## Tertiary Objects + Positioning

**Prompt 1:** Given a room described as: “a bedroom that is 4m x 5m”, with size: 4m x 5m, with these regions within it: regions, and these objects already in it: primary\_objects + secondary\_objects, suggest any rugs, other decorations or objects (these are things that go on top of other objects) that should be placed in the room. For example, a tv to go on the tv unit, a painting above the bed, a table lamp on the nightstand, a chandelier over the dining table, a computer to go on the desk, etc. Don't give any furniture or objects that go on the floor (other than rugs). Everything should either go on the wall, on the ceiling, or on top of a piece of furniture. Don't include curtains. Tell me the dimensions of the objects (width and length not height - in meters). If it is a wall object, its length should be 0.1m. Tell me how they should be placed (with respect to one object. e.g. painting should be placed near the dining table or lamp should be placed on a nightstand). Don't give too many objects and don't include multi-object objects e.g. gallery walls or shelves.

**Output:**

Certainly! Here are a few additions that can enhance the room, along with their dimensions and placement suggestions:

### Sleeping Area:

1. \*\*Table Lamp\*\*
  - \*\*Dimensions\*\*: 0.3m x 0.3m
  - \*\*Placement\*\*: One on each nightstand.

2. **Painting**
  - **Dimensions**: 1.0m x 0.1m
  - **Placement**: Centered above the bed

### Storage Area:

3. **Mirror**
  - **Dimensions**: 1.2m x 0.1m
  - **Placement**: Hang it above the dresser.

### Work Area:

4. **Desk Organizer**
  - **Dimensions**: 0.4m x 0.2m
  - **Placement**: On the desk for holding pens, papers, etc.

5. **Computer Monitor**
  - **Dimensions**: 0.5m x 0.2m
  - **Placement**: Centered on the desk.

### General Decorations:

6. **Chandelier**
  - **Dimensions**: (Round) 0.8m diameter (Height should be adjusted according to ceiling height)
  - **Placement**: Centered in the room.

7. **Rug**
  - **Dimensions**: 2.0m x 3.0m
  - **Placement**: Under the bed, extending outward to provide a soft landing spot for feet.

These additions will make the room more functional and aesthetically pleasing without overcrowding it.

### **Prompt 2:**

Given this list of tertiary objects: `tertiary_objects`, use the function attached to create the objects with the correct parameters. The room name is: `room_name`. The room description is: "a bedroom that is 4m x 5m". The room is already set up, only add in the objects using the '`create_tertiary_object`' function. Ensure that each objects index is unique and that the indices begin from 0. The object names should be simple and short. The objects should be given the correct tertiary type (ceiling, floor, wall, or table). Wall objects should all have length 0.1m. Ensure that the parameters are set correctly. This is the function:  
```

```
def create_tertiary_object(room, name, width, length, tertiary, index):
    " A function that creates and places a tertiary object
    Inputs:
        room: Room for the object to be put in
        name: str, name of the object all lowercase. E.g. 'painting'
        width: float, width of the object (m)
        length: float, length of the object (m)
        tertiary: str, tertiary object type, one of "wall" (for objects that go on the wall e.g.
        painting),
```

"floor" (for objects that go on the floor e.g. rug), "ceiling" (for objects that go on the ceiling e.g. chandelier),

"table" (for objects that go on a surface e.g. lamp).

index: int, index of the object in the room's tertiary object list

""

No extra text, only the function calls. Don't have 'python' at the start of the code. Do not define ANY functions, only call them.

### **Output:**

```
create_tertiary_object(room_name 'lamp', 0.3, 0.3, 'table', 0)
create_tertiary_object(room_name 'lamp', 0.3, 0.3, 'table', 1)
create_tertiary_object(room_name 'painting', 1.0, 0.1, 'wall', 2)
create_tertiary_object(room_name 'mirror', 1.2, 0.1, 'wall', 3)
create_tertiary_object(room_name, 'organizer', 0.4, 0.2, 'table', 4)
create_tertiary_object(room_name, 'monitor', 0.5, 0.2, 'table', 5)
create_tertiary_object(room_name, 'chandelier', 0.8, 0.8, 'ceiling', 6)
create_tertiary_object(room_name, 'rug', 2.0, 3.0, 'floor', 7)
```

## Style

### **Prompt:**

Given the description of the room: " a bedroom that is 4m x 5m ", with size: 4m x 5m, with these objects within it: primary\_objects + secondary\_objects + tertiary\_objects, tell me the colours of the walls and whether there should be wallpaper/paint/some other wall material. Tell me the style and colour of the windows and doors. Tell me the material and colour of the floor, and for each object, one by one, describe its colour, style, material etc. Make sure that all of the objects are cohesive together and match the description of the room.

### **Output:**

Based on the dimensions provided (4m x 5m) and the list of objects, we'll create a cohesive and aesthetically pleasing bedroom design with thoughtful selections for wall treatments, flooring, and each object.

#### **### Walls:**

- \*\*Colour:\*\* A soft, neutral palette such as light grey or warm beige to provide a serene and relaxing environment.
- \*\*Treatment:\*\* High-quality paint for a clean, modern look. Avoid wallpaper to maintain simplicity and versatility.

#### **### Floor:**

- \*\*Material:\*\* Hardwood for durability and elegance.
- \*\*Colour:\*\* A medium oak finish that provides warmth and complements the neutral wall colour.

#### **## Door:**

- \*\*Colour:\*\* A matching dark walnut finish, consistent with the bed frame and nightstands.

- **Material:** Solid wood for durability and a high-quality feel.
- **Style:** Modern with a clean, flat panel design, featuring sleek silver hardware that ties in with the wardrobe handles. The door should maintain simplicity to blend seamlessly with the room's overall aesthetic.

#### ## Windows:

- **Colour and Material:** Aluminum or wood, finished in a white or light grey to match the Scandinavian-inspired wardrobe and dresser.
- **Style:** Casement windows with clear glass, providing a modern, sleek appearance while also offering excellent insulation.

---

#### ### Objects:

1. **Bed:**
  - **Size:** Queen-sized for proportionate fitting in the room.
  - **Colour:** Dark walnut frame with a soft grey upholstered headboard.
  - **Style:** Modern with clean lines and minimalistic design.
2. **Wardrobe:**
  - **Colour:** White with sleek silver handles.
  - **Style:** Scandinavian-inspired with a focus on functionality and simplicity.
3. **Desk:**
  - **Colour:** Light wood top with white metal legs.
  - **Style:** Minimalist, providing a clutter-free workspace.
4. **Nightstands (2):**
  - **Colour:** Matching the bed frame, dark walnut.
  - **Style:** Modern with a single drawer for a clean appearance.
5. **Dresser:**
  - **Colour:** White to match the wardrobe.
  - **Style:** Contemporary with straight lines and silver handles.
6. **Desk Chair:**
  - **Colour:** Light grey fabric with a chrome base.
  - **Style:** Ergonomic and modern.
7. **Lamp (2):**
  - **Colour:** White ceramic base with a light grey shade.
  - **Style:** Elegant and simple, fitting well with nightstands and dresser.
8. **Painting:**
  - **Colour:** A modern abstract piece with predominant shades of grey, blue, and white to add a pop of colour.
  - **Style:** Framed in a sleek black frame for a gallery-quality look.
9. **Mirror:**
  - **Colour:** Frameless or with a thin silver frame.

- **Style:** Large and full-length for functionality and room enlargement effect.

10. **Organizer:**

- **Colour:** Soft beige fabric bins with white wooden frame.
- **Style:** Practical and minimalistic.

11. **Monitor:**

- **Colour:** Black.
- **Style:** Sleek and modern, with a small footprint on the desk.

12. **Chandelier:**

- **Colour:** Polished chrome with clear crystal accents.
- **Style:** Modern and elegant, adding a touch of luxury to the room.

13. **Rug:**

- **Colour:** Light grey with subtle geometric patterns in white.
- **Style:** Plush and cozy, placed under the bed extending slightly beyond it.

---

### ### Cohesion and Harmony:

The selections ensure a cohesive design that blends modern, minimalist, and Scandinavian elements. The soft neutral walls and medium oak flooring create a warm and balanced foundation. Each piece of furniture and accessory has been chosen for its simplicity, elegance, and ability to create a tranquil, aesthetically pleasing bedroom.

## Constraint Cleaning

This prompt is used 4 times: for the primary individual constraints, for the primary inter-object constraints, for the secondary individual constraints, and the secondary inter-object constraints.

**Prompt 1:** I am going to give you a list of constraints. I want you to merge any similar constraints and also remove any duplicate constraints. Each constraint should be simplified, so that it is said in the most basic terms. I want you to remove any constraints that are contradictory e.g. should be near a window AND should be away from a window. If there is an "or" statement in the constraint, choose one of the options. Finally, each bullet constraint should only specify one constraint. For example if there is a bullet point that says an object should not block windows and doors, change this into 2 bullets - one for blocking doors, and one for blocking windows. Remove any constraints that have if statements, and any that involve height. If there are constraints for objects to be accessible from all sides that are unnecessary, change them to be which sides you think ARE necessary. If there are constraints with "or" in them, choose between the options. Return the list of cleaned constraints. Here is the list of constraints: constraints.

### Output 1:

### Cleaned Constraints:

#### #### Bed (1.5m x 2m)

- Should be placed against a wall.
- Should not block access to windows.
- Should not block access to doors.
- Front side should be accessible.
- Should allow for at least 60 cm of free space on one side.

#### #### Wardrobe (1.2m x 0.6m)

- Should be placed against a wall.
- Should not block access to windows.
- Should not block access to doors.
- Front side should be accessible.
- Avoid placing it where it will block electrical sockets.
- Ideally placed in a corner.

#### #### Desk (1.2m x 0.6m)

- Should be placed against a wall.
- Should not block access to windows.
- Should not block access to doors.
- Front side should be accessible.
- Ideally positioned near electrical sockets.
- Should not block airflow from vents.

### **Output 2:**

Here are the cleaned and simplified constraints based on your requirements:

- The bed should not be close to the desk.
- The bed should not be close to the wardrobe.
- The desk should have enough space around it to allow for comfortable work.
- The wardrobe should be accessible without obstruction by the bed.
- The wardrobe should be accessible without obstruction by the desk.
- Ensure there is sufficient walking space between the bed and the desk.
- Ensure there is sufficient walking space between the bed and the wardrobe.
- Ensure there is sufficient walking space between the desk and the wardrobe.

### **Output 3:**

Here is the cleaned list of constraints:

#### ### Sleeping:

#### #### Nightstand (0.5m x 0.4m) (2 items)

- Can be placed against a wall.

#### ### Storage:

#### #### Dresser (1.0m x 0.5m)

- Should be placed against a wall.
- Front should be accessible for use.

### Work:  
#### Desk Chair (0.6m x 0.6m)  
- Should not block any doors.  
- Should not block any windows.  
- Should have enough space around it to pull out and sit down.

#### **Output 4:**

\*\*Cleaned Constraints:\*\*

1. \*\*Bed and Nightstands:\*\*
  - One nightstand should be to the right of the bed.
  - One nightstand should be to the left of the bed.
  - The nightstands should be aligned with the headboard of the bed.
2. \*\*Desk Chair and Desk:\*\*
  - The front of the desk chair should be against the front of the desk.
  - Ensure there is adequate space behind the desk chair for movement.
3. \*\*Dresser and Wardrobe:\*\*
  - Maintain at least 0.5m of space in front of the dresser for access.
  - Maintain at least 0.5m of space in front of the wardrobe for access.
  - The dresser and wardrobe should not be placed next to each other to avoid crowding.
4. \*\*Bed and Dresser:\*\*
  - Maintain a minimum distance of 0.5m between the bed and the dresser for ease of movement.
5. \*\*Bed and Wardrobe:\*\*
  - Maintain a minimum distance of 0.5m between the bed and the wardrobe for ease of movement.
6. \*\*Desk and Bed/Nightstands:\*\*
  - The desk should be positioned so there is a minimum distance of 0.5m from the bed and nightstands.
  - The desk should be placed along a wall opposite the bed.
7. \*\*Desk and Other Storage (Dresser/Wardrobe):\*\*
  - The desk should have a minimum distance of 0.5m from the dresser.
  - The desk should have a minimum distance of 0.5m from the wardrobe.

\*\*General Note:\*\* Ensure that all furniture placement considers the need for adequate circulation space and ease of access to various parts of the room.

## Translation

#### **Prompt 1:**

Given the room\_name room\_name, as well as the primary objects: primary\_objects, their indices primary\_indices, and these constraints: primary\_individual\_cons, using the script attached, transform each constraint into a SINGLE function call that will ensure that the primary objects are set up correctly. Match each constraint to the closest function in the script file by using the docstrings. Ensure that every constraint is mapped to a function call, unless there is not one that matches. Here is the first script: file\_contents2. For the second script, I want you to transform each pairwise constraint into a function call. Here is the second script: file\_contents3, and here is the list of pairwise constraints: primary\_io\_cons. I want the output to begin with: def optimize\_primary\_objects(positions, room): \n output = 0, followed by each constraint function call added to the output, and then the output returned (everyline should begin with 'output +=' except for the final line). Go object by object, and then pairwise. No extra text please, only the functions and the output. If there is only 1 primary object, there should be no pairwise calls. Do not define ANY functions, only call them. ENSURE that all parameters match docstrings.

### **Output:**

```
def optimize_primary_objects(positions, room):
    output = 0
    output += check_and_call('ind_next_to_wall', positions, room, 0)
    output += check_and_call('ind_not_block_fixed_object', positions, room, 0, 'window')
    output += check_and_call('ind_not_block_fixed_object', positions, room, 0, 'door')
    output += check_and_call('ind_accessible', positions, room, 0, ['front'])
    output += check_and_call('ind_accessible', positions, room, 0, ['left', 'right'], 0.6)
    output += check_and_call('ind_next_to_wall', positions, room, 1)
    output += check_and_call('ind_not_block_fixed_object', positions, room, 1, 'window')
    output += check_and_call('ind_not_block_fixed_object', positions, room, 1, 'door')
    output += check_and_call('ind_accessible', positions, room, 1, ['front'])
    output += check_and_call('ind_away_from_fixed_object', positions, room, 1, 'socket')
    output += check_and_call('ind_in_corner', positions, room, 1)
    output += check_and_call('ind_next_to_wall', positions, room, 2)
    output += check_and_call('ind_not_block_fixed_object', positions, room, 2, 'window')
    output += check_and_call('ind_not_block_fixed_object', positions, room, 2, 'door')
    output += check_and_call('ind_accessible', positions, room, 2, ['front'])
    output += check_and_call('ind_close_to_fixed_object', positions, room, 2, 'socket')
    output += check_and_call('ind_away_from_fixed_object', positions, room, 2, 'vent')
    output += check_and_call('io_away_from', positions, room, 0, 2, 2.0)
    output += check_and_call('io_away_from', positions, room, 0, 1, 2.0)
    output += check_and_call('io_away_from', positions, room, 2, 1, 2.0)
    output += check_and_call('io_away_from', positions, room, 1, 0, 2.0)
    output += check_and_call('io_next_to', positions, room, 1, 0)
    output += check_and_call('io_near', positions, room, 2, 0, 3.0)
    output += check_and_call('io_near', positions, room, 1, 2, 3.0)

    output += wall_attraction(positions, room)
    output += in_bounds(positions, room)
    output += no_overlap(positions, room)
    output += aligned(positions, room)
    output += balanced(positions, room)
    return output
```

**Prompt 2:**

Given the primary objects: primary\_objects, and their indices: primary\_indices, as well as the secondary objects, along with their indices and their regions: [(s\_i, ind\_i, r\_i), ..], their individual constraints: secondary\_ind\_cons, and the region names: regions. Using the script attached, transform each constraint into a SINGLE function call that will ensure that the secondary objects are set up correctly. Match each constraint to the closest function in the script file by using the docstrings. Ensure that every constraint is mapped to a function call, unless there is not one that fits. Here is the first script: file\_contents2. For the second script, I want you to transform each pairwise constraint into a SINGLE function call. Here is the second script: file\_contents3, and here is the list of pairwise constraints: s\_io\_cons. I want the output to begin with: 'def optimize\_secondary\_objects(positions, room): \n output = 0\n', followed by each constraint function call added to the output, (each line should begin with 'output +='). Go object by object, and then pairwise. Also ensure to use ind\_in\_region for each object. Do not define ANY functions, only call them. No extra text please, only the functions and the output.

**Output:**

```
def optimize_secondary_objects(positions, room):
    output = 0

    output += check_and_call('ind_in_region', positions, room, 3, 'sleeping')
    output += check_and_call('ind_next_to_wall', positions, room, 3)
    output += check_and_call('ind_in_region', positions, room, 4, 'sleeping')
    output += check_and_call('ind_next_to_wall', positions, room, 4)
    output += check_and_call('ind_in_region', positions, room, 5, 'storage')
    output += check_and_call('ind_next_to_wall', positions, room, 5)
    output += check_and_call('ind_accessible', positions, room, 5, ['front'], min_dist = 0.5)
    output += check_and_call('ind_in_region', positions, room, 6, 'work')
    output += check_and_call('ind_not_block_fixed_object', positions, room, 6, 'door')
    output += check_and_call('ind_not_block_fixed_object', positions, room, 6, 'window')
    output += check_and_call('ind_accessible', positions, room, 6, ['front', 'back'], min_dist =
0.6)
    output += check_and_call('io_next_to', positions, room, 3, 0, side1='right')
    output += check_and_call('io_next_to', positions, room, 4, 0, side1='left')
```

```

output += check_and_call('io_parallel', positions, room, 3, 4, max_dist=0.5)
output += check_and_call('io_next_to', positions, room, 6, 2, side1='front', side2='front')
output += check_and_call('ind_accessible', positions, room, 6, ['back'], min_dist=0.6)
output += check_and_call('ind_accessible', positions, room, 5, ['front'], min_dist=0.5)
output += check_and_call('ind_accessible', positions, room, 1, ['front'], min_dist=0.5)
output += check_and_call('io_away_from', positions, room, 5, 1, min_dist=0.5)
output += check_and_call('io_away_from', positions, room, 5, 0, min_dist=0.5)
output += check_and_call('io_away_from', positions, room, 1, 0, min_dist=0.5)
output += check_and_call('io_away_from', positions, room, 2, 0, min_dist=0.5)
output += check_and_call('io_away_from', positions, room, 2, 4, min_dist=0.5)
output += check_and_call('ind_next_to_wall', positions, room, 2)
output += check_and_call('io_away_from', positions, room, 2, 5, min_dist=0.5)
output += check_and_call('io_away_from', positions, room, 2, 1, min_dist=0.5)

```

```

output += wall_attraction(positions, room)
output += in_bounds(positions, room)
output += no_overlap(positions, room)
output += aligned(positions, room)
return output

```

### **Prompt 3:**

Given all the objects in the room: primary\_objects + secondary\_objects, and their indices: primary\_indices + secondary\_indices, as well as the region names: regions, here are the tertiary objects and how they should be placed: tertiary\_output. These are there indices: tertiary\_indices. Using the script attached, transform each placement into a SINGLE function call that will ensure that the tertiary objects are placed correctly. If there is not a function that fits, then choose another. Here is the script: file\_contents4. I want the output to begin with: 'def optimize\_tertiary\_objects(positions, room): \n output = 0\n', followed by each function call added to the output, (each line should begin with 'output +='). Do not define ANY functions, only call them. No extra text please, only the functions and the output.

**Output:**

```
def optimize_ternary_objects(positions, room):  
    output = 0  
  
    output += check_and_call('on_top_central', positions, room, 0, 3)  
    output += check_and_call('on_top_central', positions, room, 1, 4)  
    output += check_and_call('on_wall_above_central', positions, room, 2, 0)  
    output += check_and_call('on_wall_above_central', positions, room, 3, 5)  
    output += check_and_call('on_top_central', positions, room, 4, 2)  
    output += check_and_call('on_top_central', positions, room, 5, 2)  
    output += check_and_call('center_ceiling', positions, room, 6)  
    output += check_and_call('rug_under_central_forward', positions, room, 7, 0)  
  
    output += check_and_call('t_valid', positions, room)  
    return output
```

## References

- [Ble18] BLENDER ONLINE COMMUNITY: *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL: <http://www.blender.org>. 7, 8
- [Ble24] BLENDERKIT CONTRIBUTORS: BlenderKit: Free 3D models, materials, brushes and add-ons directly in Blender. <https://www.blenderkit.com>, 2024. Accessed: 2024-09-01. 7, 8
- [YSW\*23] YANG Y., SUN F.-Y., WEIHS L., VANDERBILT E., HER-RASTI A., HAN W., WU J., HABER N., KRISHNA R., LIU L., CALLISON-BURCH C., YATSKAR M., KEMBHAVI A., CLARK C.: Holodeck: Language Guided Generation of 3D Embodied AI Environments, 2023. URL: <https://arxiv.org/abs/2312.09067>, doi:10.48550/ARXIV.2312.09067. 1, 2
- [S\*24] ÇELEN A., HAN G., SCHINDLER K., GOOL L. V., ARMENI I., OBUKHOV A., WANG X.: I-Design- Personalized LLM Interior Designer, 2024. arXiv:arXiv:2404.02838. 1, 2