

---

**donjuan**

***Release 12-20-2020***

**Tom McClintock, Jamie McClintock, James Devore**

**Jan 12, 2021**



**CONTENTS:**

<b>1</b>	<b>DonJuan</b>	<b>3</b>
<b>2</b>	<b>Developing</b>	<b>5</b>
<b>3</b>	<b>API Reference</b>	<b>7</b>
<b>4</b>	<b>Indices and tables</b>	<b>27</b>
	<b>Python Module Index</b>	<b>29</b>
	<b>Index</b>	<b>31</b>



This package is a rebuild of the [donjon](#) dungeon generator, with all the benefits that come from using modern tools. In *donjuan*, all parts of the dungeon are objects that can be subclassed and serialized (saved to disk) on their own. This includes things like walls, doors, and rooms, as well as algorithms for actual dungeon generation, renderers for different image formats, and exporters for saving map files for different VTT applications.



## DONJUAN

A translation/rebuild of the original [donjon dungeon generator](#). This package aims to deconstruct the original script into extendable parts, and provide all pieces for customization for different purposes and not just the map image. For example, when complete this package will automatically generate walls, doors, and light sources for use in [Foundry Virtual Tabletop](#).

You can find the [documentation here](#).

## 1.1 Installation

Install donjuan with pip:

```
pip install donjuan
```

You can find the package details [here on PyPI](#).

You can also install donjuan using the `setup.py` file. To do so, you must first clone or download this repository and install the requirements.

Assuming you have [git](#), you can do:

```
git clone https://github.com/tmcclintock/donjuan
cd donjuan
pip install -r requirements.txt
```

Then you can install with:

```
python setup.py install
```

If you have [conda](#) you can install the requirements using the `environment.yml` file before installing:

```
conda env create -f environment.yml
conda activate donjuan
python setup.py install
```

To run the test suite, you must have [pytest](#) installed. You can run the tests with:

```
pytest
```

which can be done from the root of the repository. To run all tests, including those with graphical outputs, run with the `runslow` flag:

```
pytest --runslow
```

Please report any issues you encounter on our [issue page](#). Doing so will help make donjuan even better!

## 1.2 Contributing

To contribute to donjuan please see the [developing page](#).



## DEVELOPING

Developing *donjuan* should follow best practices. This includes documentation, testing, and using composability.

Composability is the design concept that complex objects use more simple objects that are responsible for very specific tasks. For example, a *Dungeon* is composed of rooms and passages that are fundamentally made up of *Cell* objects that individually have properties, such as *Door* objects and *Faces*.

If you want to implement a new piece of logic and you are not sure where it goes, create a new class for it and use that class as appropriate.

### 2.1 Contributing

To contribute, please start by forking the repository and creating a branch for the feature you would like to work on. Once you clone your feature branch, install the developer environment, activate it and install *donjuan* in editable mode:

```
conda env create -f environment_dev.yml
conda activate djdev
pip install -e .
```

In this setup, any changes you make to the source code of *donjuan* will be seen immediately by your *djdev* environment. Furthermore, you will not have to reinstall *donjuan* if you switch branches.

Once your development environment is activated, please install [pre-commit](#) before committing any changes:

```
pre-commit install
```

This way, any time you perform a *git commit* command, the *black*, *flake8*, and *isort* packages will run to clean up your code. If you see a “failure” message, then simply *git reset* and then fix any issues the *pre-commit* packages are giving you. If *pre-commit* is being too annoying, then you can always recreate your conda environment without installing it.

Once your code is ready, commit and push it to your branch and issue a Pull Request on GitHub.

### 2.2 Testing data and a sample dungeon

Tests should be written in the *tests/* directory, so that running *pytest* at the root finds and runs all tests found there. Tests should be functions attached to classes that subclass *unittest.TestCase*. This allows you to write *setUp* and *tearDown* functions that run before and after all tests. A sample dungeon is found in *tests/fixtures/dummy\_dungeon.json*, where the *filled* attribute of a five-by-five dungeon have been saved in a 2D array. To load in this dungeon in memory simply run:

```
import json
fp = "/path/to/dummy_dungeon.json"
with open(fp, "r") as file:
    dungeon_json = json.load(file)
dungeon_array = dungeon_json["dungeon"] # the actual array
```

Then, to turn this into an actual *Dungeon* object, instantiate a *Dungeon* that is five by five and assign the *filled* values according to the values in the read array.

```
dungeon = Dungeon(n_rows=5, n_cols=5)

for i in range(5):
    for j in range(5):
        dungeon.grid.cells[i][j].filled = bool(dungeon_array[i][j])
```

## 2.3 Docstrings

Docstrings are automatically rendered in the online documentation using Napoleon sphinx. Docstrings follow the [Google style](#). All classes and non-trivial functions and properties should be documented. Function signatures should be *typed*. Module-level docstrings at the top of *.py* files are not required.

## API REFERENCE

This page contains auto-generated API reference documentation<sup>1</sup>.

### 3.1 donjuan

#### 3.1.1 Submodules

`donjuan.cell`

#### Module Contents

#### Classes

<code>Cell</code>	The cell represents a single ‘unit’ in the map. The attributes on the cell
<code>SquareCell</code>	A cell for a square grid.
<code>HexCell</code>	A cell for a hexagonal grid.

**class** `donjuan.cell.Cell` (*filled: bool = False, door\_space: Optional[DoorSpace] = None, contents: Optional[List[Any]] = None, coordinates: Optional[Tuple[int, int]] = None, space: Optional[‘Space’] = None, edges: Optional[List[‘Edge’]] = None*)

Bases: `abc.ABC`

The cell represents a single ‘unit’ in the map. The attributes on the cell define what exists in that unit. This includes the terrain, doors, walls, lights, other room details etc.

**set\_coordinates** (*self, y: int, x: int*) → None

**set\_edges** (*self, edges: List[‘Edge’]*) → None

**set\_space** (*self, space: Type[‘Space’]*) → None

**set\_x** (*self, x: int*) → None

**set\_y** (*self, y: int*) → None

**property coordinates** (*self*) → Tuple[int, int]

**property edges** (*self*) → List[‘Edge’]

---

<sup>1</sup> Created with `sphinx-autoapi`

**property** **space** (*self*) → Optional[Type['Space']]  
Space this cell is a part of.

**property** **x** (*self*) → int

**property** **y** (*self*) → int

**property** **n\_sides** (*self*) → int

**class** donjuan.cell.**SquareCell** (*filled: bool = False, door\_space: Optional[DoorSpace] = None, contents: Optional[List[Any]] = None, coordinates: Optional[Tuple[int, int]] = None*)

Bases: [donjuan.cell.Cell](#)

A cell for a square grid.

#### Parameters

- **filled** (*bool, optional*) – flag indicating whether the cell is filled (default `False`)
- **door\_space** (*Optional[DoorSpace]*) – kind of doorway in this cell
- **contents** (*Optional[List[Any]]*) – things in this cell

**\_n\_sides** = 4

**class** donjuan.cell.**HexCell** (*filled: bool = False, door\_space: Optional[DoorSpace] = None, contents: Optional[List[Any]] = None, coordinates: Optional[Tuple[int, int]] = None*)

Bases: [donjuan.cell.Cell](#)

A cell for a hexagonal grid.

#### Parameters

- **filled** (*bool, optional*) – flag indicating whether the cell is filled (default `False`)
- **door\_space** (*Optional[DoorSpace]*) – kind of doorway in this cell
- **contents** (*Optional[List[Any]]*) – things in this cell

**\_n\_sides** = 6

## donjuan.door\_space

Door ways that connect rooms to hallways and rooms to rooms.

## Module Contents

### Classes

---

<a href="#">DoorSpace</a>	Abstract base class for different kinds of doors. Door spaces can have many
<a href="#">Archway</a>	An archway to walk through.
<a href="#">Door</a>	A generic door.
<a href="#">Portcullis</a>	You can look but can't touch!

---

**class** donjuan.door\_space.**DoorSpace** (*secret: bool, locked: bool, closed: bool, jammed: bool, blocked: bool, broken: bool, material: str, name: str*)

Bases: `abc.ABC`

Abstract base class for different kinds of doors. Door spaces can have many properties, like if they are locked or blocked etc. To facilitate this logic in the generative process, these are encompassed in the attributes of a `DoorSpace`.

```
__slots__ = ['locked', 'closed', 'jammed', 'blocked', 'secret', 'broken', 'material',
__str__(self)
    Return str(self).
```

```
class donjuan.door_space.Archway(material: str = 'stone', blocked: bool = False, broken: bool =
                                False, secret: bool = False)
    Bases: donjuan.door_space.DoorSpace
```

An archway to walk through.

```
class donjuan.door_space.Door(secret: bool = False, locked: bool = False, closed: bool = True,
                              jammed: bool = False, blocked: bool = False, broken: bool =
                              False, material: str = 'wood')
    Bases: donjuan.door_space.DoorSpace
```

A generic door.

```
class donjuan.door_space.Portcullis(locked: bool = False, closed: bool = True, jammed: bool
                                     = False, broken: bool = False, material: str = 'metal')
    Bases: donjuan.door_space.DoorSpace
```

You can look but can't touch!

## donjuan.dungeon

### Module Contents

#### Classes

---

#### *Dungeon*

---

```
class donjuan.dungeon.Dungeon(n_rows: Optional[int] = 5, n_cols: Optional[int] = 5, grid: Op-
                              tional[Grid] = None, rooms: Optional[Dict[str, Room]] = None,
                              hallways: Optional[Dict[str, Hallway]] = None, randomizers:
                              Optional[List['Randomizer']] = None)
```

```
    add_room(self, room: Room) → None
```

```
    add_hallway(self, hallway: Hallway) → None
```

```
    property grid(self) → Grid
```

```
    property hallways(self) → Dict[str, Hallway]
```

```
    property n_cols(self) → int
```

```
    property n_rows(self) → int
```

```
    property randomizers(self) → List['Randomizer']
```

```
    property rooms(self) → Dict[str, Room]
```

```
    randomize(self) → None
```

For each item in `randomizers`, run the `Randomizer.randomize_dungeon()` method on this dungeon.

**emplace\_rooms** (*self*) → None

Replace the cells in the *grid* with the cells of the *rooms*.

**emplace\_space** (*self*, *space*: *Space*) → None

Replace the cells in the *grid* with the cells of the *Space*, and automatically link them with the *Edge*'s in the *Grid*.

**Parameters** *space* (*Space*) – room to emplace in the *grid*

`donjuan.dungeon_randomizer`

## Module Contents

### Classes

---

<i>DungeonRandomizer</i>	Randomize a dungeon by first creating rooms and then applying
--------------------------	---

---

```
class donjuan.dungeon_randomizer.DungeonRandomizer (room_entrance_randomizer:
                                                    Optional[Randomizer] = None,
                                                    room_size_randomizer: Optional[Randomizer] = None,
                                                    room_name_randomizer: Optional[Randomizer] = None,
                                                    room_position_randomizer: Optional[Randomizer] = None,
                                                    max_num_rooms: Optional[int] = None, max_room_attempts: int =
                                                    100)
```

Bases: *donjuan.randomizer.Randomizer*

Randomize a dungeon by first creating rooms and then applying room size, name, and position randomizers to sequentially generated rooms.

#### Parameters

- **room\_entrance\_randomizer** (*Optional[Randomizer]*) – randomizer for the entrances of a room. If None then default to a RoomEntrancesRandomizer.
- **room\_size\_randomizer** (*Optional[Randomizer]*) – randomizer for the room size. It must have a 'max\_size' attribute. If None then default to a RoomSizeRandomizer.
- **room\_name\_randomizer** (*Optional[Randomizer]*) – randomizer for the room name. If None default to a AlphaNumRoomName.
- **room\_position\_randomizer** (*Optional[Randomizer]*) – randomizer for the room position. If None default to a RoomPositionRandomizer.
- **max\_num\_rooms** (*Optional[int]*) – maximum number of rooms to draw, if ``None`` then default to the max\_room\_attempts. See *DungeonRoomRandomizer.get\_number\_of\_rooms()* for details.
- **max\_room\_attempts** (*int, optional*) – default is 100. Maximum number of attempts to generate rooms.

**get\_number\_of\_rooms** (*self*, *dungeon\_n\_rows*: int, *dungeon\_n\_cols*: int) → int

Randomly determine the number of rooms based on the size of the incoming grid or the `max_num_rooms` attribute, whichever is less.

#### Parameters

- **dungeon\_n\_rows** (*int*) – number of rows
- **dungeon\_n\_cols** (*int*) – number of columns

**randomize\_dungeon** (*self*, *dungeon*: *Dungeon*) → None

Randomly put rooms in the dungeon.

**Parameters** **dungeon** (*Dungeon*) – dungeon to randomize the rooms of

`donjuan.edge`

## Module Contents

### Classes

<i>Edge</i>	An edge sits between two <code>Cell</code> 's.
-------------	--

**class** `donjuan.edge.Edge` (*cell1*: *Optional[Cell]* = None, *cell2*: *Optional[Cell]* = None, *has\_door*: *bool* = False)

An edge sits between two `Cell`'s.

#### Parameters

- **cell1** (*Optional[Cell]*) – cell on one side of the edge
- **cell2** (*Optional[Cell]*) – cell on the other side of the edge
- **has\_door** (*bool*, *optional*) – default False, indicates whether this object has a door

**property** `cell1` (*self*) → *Optional[Cell]*

**property** `cell2` (*self*) → *Optional[Cell]*

**set\_cell1** (*self*, *cell*: *Cell*) → None

**set\_cell2** (*self*, *cell*: *Cell*) → None

**property** `is_wall` (*self*) → *bool*

`donjuan.face`

## Module Contents

### Classes

<i>Face</i>	Abstract base class for the geometric face of a
<i>BareFace</i>	A face with nothing on it.
<i>DoorFace</i>	A face with a door on it.

continues on next page

Table 6 – continued from previous page

<i>Faces</i>	A collection of faces of a cell.
<i>SquareFaces</i>	Four faces surrounding a square cell.
<i>HexFaces</i>	Six faces surrounding a hexagonal cell.

**class** donjuan.face.**Face** (*direction: int = 0*)

Bases: abc.ABC

Abstract base class for the geometric face of a *Cell*.

**Parameters** *direction* (*int, optional*) – represents the outer direction of the face (default 0)

**class** donjuan.face.**BareFace** (*direction: int = 0*)

Bases: *donjuan.face.Face*

A face with nothing on it.

**class** donjuan.face.**DoorFace** (*door\_space: DoorSpace, direction: int = 0*)

Bases: *donjuan.face.Face*

A face with a door on it.

**class** donjuan.face.**Faces** (*faces: List[Face]*)

A collection of faces of a cell.

**\_\_init\_faces** (*self*) → None

**property faces** (*self*) → List[*Face*]

**\_\_len\_\_** (*self*) → int

**\_\_getitem\_\_** (*self, key*)

**class** donjuan.face.**SquareFaces** (*faces: Optional[List[Face]] = None*)

Bases: *donjuan.face.Faces*

Four faces surrounding a square cell.

**class** donjuan.face.**HexFaces** (*faces: Optional[List[Face]] = None*)

Bases: *donjuan.face.Faces*

Six faces surrounding a hexagonal cell.

**donjuan.grid**

## Module Contents

### Classes

<i>Grid</i>	Abstract base class for a grid of cells. The underlying grid can either
<i>SquareGrid</i>	Rectangular grid of square cells. In a square grid, the cell positions are
<i>HexGrid</i>	Rectangular grid of hexagonal cells. In a hex grid, the cell positions

**class** donjuan.grid.**Grid** (*n\_rows: int, n\_cols: int, cells: Optional[List[List[Cell]]] = None, edges: Optional[List[List[List[Edge]]] = None*)



Bases: `abc.ABC`

Abstract base class for a grid of cells. The underlying grid can either be square or hexagonal.

**get\_filled\_grid**(*self*) → List[List[bool]]

Obtain a 2D array of boolean values representing the `filled` state of the cells attached to the grid.

**property n\_rows**(*self*) → int

**property n\_cols**(*self*) → int

**property cells**(*self*) → List[List[Cell]]

**property edges**(*self*) → List[List[List[Edge]]]

**reset\_cell\_coordinates**(*self*) → None

Helper function that sets the coordinates of the cells in the grid to their index values.

**check\_edges**(*self*, *edges*: Optional[List[List[List[Edge]]]]) → None

Check the dimensions of the *edges*.

**init\_edges**(*self*) → List[List[List[Edge]]]

**link\_edges\_to\_cells**(*self*) → None

For an *Edge*, the `Edge.cell1` always points to either the left or upper *Cell*. The `Edge.cell2` always points to the right or the bottom.

**abstract link\_cells\_to\_edges**(*self*) → None

**class** `donjuan.grid.SquareGrid`(*n\_rows*: int, *n\_cols*: int, *cells*: Optional[List[List[SquareCell]]] = None)

Bases: `donjuan.grid.Grid`

Rectangular grid of square cells. In a square grid, the cell positions are integers.

**cell\_type**

**link\_cells\_to\_edges**(*self*) → None

**class** `donjuan.grid.HexGrid`(*n\_rows*: int, *n\_cols*: int, *cells*: Optional[List[List[HexCell]]] = None)

Bases: `donjuan.grid.Grid`

Rectangular grid of hexagonal cells. In a hex grid, the cell positions are integers, with odd rows being “offset” by half a cell size when rendered.

**cell\_type**

**link\_cells\_to\_edges**(*self*) → None

## `donjuan.hallway`

### Module Contents

#### Classes

---

*Hallway*

A hallway in a dungeon. It has a start and end cell.

---

**class** `donjuan.hallway.Hallway`(*ordered\_cells*: Optional[List[Cell]] = None, *name*: Union[int, str] = "")

Bases: `donjuan.space.Space`

A hallway in a dungeon. It has a start and end cell.

**Parameters**

- **ordered\_cells** (*Optional[Sequence[Cell]]*) – ordered list of cells, where the order defines the path of the hallway. If *None* defaults to an empty list.
- **name** (*Union[int, str], optional*) – defaults to ‘’, the name of the hallway

**property ordered\_cells** (*self*) → List[Cell]

Cells that make up the path of the hallway. Does not contain extra cells that may be associated with this object (i.e. those off of the “path”). For the set of all cells, use `cells`.

**property end\_cell** (*self*) → Cell

**property start\_cell** (*self*) → Cell

**append\_ordered\_cell\_list** (*self, cells: List[Cell]*) → None

Append cells in order to the hallway. To add a cell to the hallway without including it in the hallway's path, use `add_cells()`.

**Parameters cells** – (List[Cell]): cells to append to the hallway

**get\_coordinate\_path** (*self*) → List[Tuple[int, int]]

Get the coordinates of the cells that make up this path this hallway takes. Does not contain coordinates of extra cells on this object.

**Returns** coordinates of the hallway path

`donjuan.randomizer`

**Module Contents****Classes**

---

<code>Randomizer</code>	Class for randomizing features of a dungeon.
<code>RandomFilled</code>	Randomly set the <code>filled</code> attribute of cells.

---

**class** `donjuan.randomizer.Randomizer`

Class for randomizing features of a dungeon.

**randomize\_cell** (*self, cell: Cell*) → None

Randomize properties of the *Cell*

**randomize\_dungeon** (*self, dungeon: Dungeon*) → None

Randomize properties of the *Dungeon*

**randomize\_grid** (*self, grid: Grid*) → None

Randomize properties of the *Grid*

**randomize\_hallway** (*self, hallway: Hallway*) → None

Randomize properties of the *Hallway*

**randomize\_room\_entrances** (*self, room: Room, \*args*) → None

Randomize the entrances of the *Room*

**randomize\_room\_size** (*self, room: Room, \*args*) → None

Randomize the size of the *Room*

**randomize\_room\_name** (*self, room: Room, \*args*) → None

Randomize the name of a *Room*

**randomize\_room\_position** (*self*, *room*: *Room*, \**args*) → None  
Randomize the position of a *Room*

**classmethod seed** (*cls*, *seed*: *Optional[int]* = None) → None

**Parameters** **seed** (*Optional[int]*) – seed passed to `random.seed()`

**class** `donjuan.randomizer.RandomFilled`  
Bases: `donjuan.randomizer.Randomizer`

Randomly set the `filled` attribute of cells.

**randomize\_cell** (*self*, *cell*: *Cell*) → None  
Randomly fill the cell with probability 50%

**randomize\_grid** (*self*, *grid*: *Grid*) → None  
Randomly fill all cells of the grid individually

`donjuan.renderer`

## Module Contents

### Classes

<code>BaseRenderer</code>	Base class for rendering dungeons into images.
<code>Renderer</code>	Default renderer for rendering dungeons into PNG files using <i>matplotlib</i> .

**class** `donjuan.renderer.BaseRenderer` (*scale*: *float*)  
Bases: `abc.ABC`

Base class for rendering dungeons into images.

**property scale** (*self*) → *float*  
The scale size of a single *Cell*. The meaning differs depending on the subclass.

**abstract render** (*self*, *dungeon*: *Dungeon*) → None

**class** `donjuan.renderer.Renderer` (*scale*: *float* = 1.0)  
Bases: `donjuan.renderer.BaseRenderer`

Default renderer for rendering dungeons into PNG files using *matplotlib*.

**Parameters** **scale** (*float*, *optional*) – size of a single cell in inches (default is 1 inch).

**render** (*self*, *dungeon*: *Dungeon*, *file\_path*: *str* = 'rendered\_dungeon.png', *dpi*: *int* = 200, *save*: *bool* = *True*) → *Tuple*  
Render the dungeon.

#### Parameters

- **dungeon** (*Dungeon*) – dungeon to render
- **file\_path** (*str*, *optional*) – path to save the dungeon at (default is *rendered\_dungeon.png*)
- **dpi** (*int*, *optional*) – dots per inch used to save (default is 200)
- **save** (*bool*, *optional*) – flag indicating whether to save the dungeon with `matplotlib.pyplot.savefig()`

`donjuan.room`

## Module Contents

### Classes

---

<i>Room</i>	A room in a dungeon. A room has is a named Space.
-------------	---

---

**class** `donjuan.room.Room` (*cells: Optional[Set[Cell]] = None, name: Union[int, str] = ''*)

Bases: *donjuan.space.Space*

A room in a dungeon. A room has is a named Space.

`donjuan.room_randomizer`

## Module Contents

### Classes

---

<i>RoomSizeRandomizer</i>	Randomize the size of a <i>Room</i> .
<i>AlphaNumRoomName</i>	Simple room name randomizer that names rooms as alphabetical letters.
<i>RoomPositionRandomizer</i>	Randomly shift a room, assuming its left edge is at column 0 and it's top
<i>RoomEntrancesRandomizer</i>	Randomizes the number of entrances on a room. The number is picked to be

---

**class** `donjuan.room_randomizer.RoomSizeRandomizer` (*min\_size: int = 2, max\_size: int = 4, cell\_type: Type[Cell] = SquareCell*)

Bases: *donjuan.randomizer.Randomizer*

Randomize the size of a *Room*.

**randomize\_room\_size** (*self, room: Room*) → None

Randomly determine the size of the room, and set the cells of the room to a 2D array of unfilled cells of that size.

**class** `donjuan.room_randomizer.AlphaNumRoomName`

Bases: *donjuan.randomizer.Randomizer*

Simple room name randomizer that names rooms as alphabetical letters. followed by a number. Rooms are sequentially named 'A0', 'B0', ... 'Z0', 'A1', 'B1', ...

**next\_name** (*self*) → str

**randomize\_room\_name** (*self, room: Room, \*args*) → None

Randomize the name of a *Room*

**class** `donjuan.room_randomizer.RoomPositionRandomizer`

Bases: *donjuan.randomizer.Randomizer*

Randomly shift a room, assuming its left edge is at column 0 and it's top edge is at row 0.

**randomize\_room\_position** (*self, room: Room, dungeon: Dungeon*) → None

**Parameters**

- **room** ([Room](#)) – room to move around
- **dungeon** ([Dungeon](#)) – dungeon to move the room around in

**class** donjuan.room\_randomizer.**RoomEntrancesRandomizer** (*max\_attempts: int = 100*)

Bases: [donjuan.randomizer.Randomizer](#)

Randomizes the number of entrances on a room. The number is picked to be the square root of the number of cells in the room divided by 2 plus 1 (N) plus a uniform random integer from 0 to N.

**gen\_num\_entrances** (*self, cells: Set[Cell]*) → int

**randomize\_room\_entrances** (*self, room: Room, \*args*) → None

Randomly open edges of cells in a *Room*. The cells in the room must already be linked to edges in a *Grid*. See `emplace_rooms()`.

---

**Note:** This algorithm does not allow for a cell in a room to have two entrances.

---

**Parameters** **room** ([Room](#)) – room to try to create entrances for

donjuan.space

**Module Contents****Classes**


---

[Space](#)

A space is a section of a dungeon composed of [Cell](#)'s.  
This object

---

**class** donjuan.space.**Space** (*cells: Optional[Set[Cell]] = None, name: Union[int, str] = ''*)

Bases: `abc.ABC`

A space is a section of a dungeon composed of *Cell*'s. *This object contains these cells in a ``set`` under the property `cells`. It also has a `name` and knows about any entrances to the room (a list of *Edge* objects) via the `entrances` property.*

**Parameters**

- **cells** (*Optional[Set[Cell]]*) – cells that make up this space
- **name** (*Union[int, str], optional*) – defaults to '', the room name

**assign\_space\_to\_cells** (*self*) → None

Set the space attribute for each *Cell* to self.

**reset\_cell\_coordinates** (*self*) → None

**property cells** (*self*) → Set[Cell]

**property cell\_coordinates** (*self*) → Set[Tuple[int, int]]

**property name** (*self*) → Union[int, str]

**add\_cells** (*self, cells: Sequence[Cell]*) → None

Add cells to the set of cells in this space. Cells are added to both the `cells` set and the cell coordinates to the `cell_coordinates` set.

**Parameters** `cells` (*Sequence*[*Cell*]) – any iterable collection of cells

**overlaps** (*self*, *other*: *Space*) → bool

Compare the cells of this space to the other space to determine whether they overlap or not. Note, this algorithm is  $O(N)$  where  $N$  is the number of cells in this space, since set lookup is  $O(1)$ .

**Parameters** `other` (*Space*) – other space to check against

**Returns** True if they overlap, False if not

**set\_name** (*self*, *name*: *Union*[*int*, *str*]) → None

**shift\_vertical** (*self*, *n*: *int*) → None

Change the y coordinates of all *cells* by *n*.

**Parameters** `n` (*int*) – number to increment vertical position of cells

**shift\_horizontal** (*self*, *n*: *int*) → None

Change the x coordinates of all *cells* by *n*.

**Parameters** `n` (*int*) – number to increment horizontal position of cells

### 3.1.2 Package Contents

#### Classes

<i>Cell</i>	The cell represents a single ‘unit’ in the map. The attributes on the cell
<i>HexCell</i>	A cell for a hexagonal grid.
<i>SquareCell</i>	A cell for a square grid.
<i>Archway</i>	An archway to walk through.
<i>Door</i>	A generic door.
<i>DoorSpace</i>	Abstract base class for different kinds of doors. Door spaces can have many
<i>Portcullis</i>	You can look but can’t touch!
<i>Dungeon</i>	
<i>DungeonRandomizer</i>	Randomize a dungeon by first creating rooms and then applying
<i>Edge</i>	An edge sits between two <i>Cell</i> ’s.
<i>Grid</i>	Abstract base class for a grid of cells. The underlying grid can either
<i>HexGrid</i>	Rectangular grid of hexagonal cells. In a hex grid, the cell positions
<i>SquareGrid</i>	Rectangular grid of square cells. In a square grid, the cell positions are
<i>Hallway</i>	A hallway in a dungeon. It has a start and end cell.
<i>RandomFilled</i>	Randomly set the <i>filled</i> attribute of cells.
<i>Randomizer</i>	Class for randomizing features of a dungeon.
<i>BaseRenderer</i>	Base class for rendering dungeons into images.
<i>Renderer</i>	Default renderer for rendering dungeons into PNG files using <i>matplotlib</i> .
<i>Room</i>	A room in a dungeon. A room has a named <i>Space</i> .
<i>AlphaNumRoomName</i>	Simple room name randomizer that names rooms as alphabetical letters.

continues on next page

Table 14 – continued from previous page

<i>RoomEntrancesRandomizer</i>	Randomizes the number of entrances on a room. The number is picked to be
<i>RoomPositionRandomizer</i>	Randomly shift a room, assuming its left edge is at column 0 and it's top
<i>RoomSizeRandomizer</i>	Randomize the size of a <i>Room</i> .
<i>Space</i>	A space is a section of a dungeon composed of <i>Cell</i> 's. This object

donjuan.\_\_version\_\_ = 0.0.3

donjuan.\_\_docs\_\_ = Package for generating dungeons.

**class** donjuan.Cell (*filled: bool = False, door\_space: Optional[DoorSpace] = None, contents: Optional[List[Any]] = None, coordinates: Optional[Tuple[int, int]] = None, space: Optional['Space'] = None, edges: Optional[List['Edge']] = None*)

Bases: abc.ABC

The cell represents a single 'unit' in the map. The attributes on the cell define what exists in that unit. This includes the terrain, doors, walls, lights, other room details etc.

**set\_coordinates** (*self, y: int, x: int*) → None

**set\_edges** (*self, edges: List['Edge']*) → None

**set\_space** (*self, space: Type['Space']*) → None

**set\_x** (*self, x: int*) → None

**set\_y** (*self, y: int*) → None

**property coordinates** (*self*) → Tuple[int, int]

**property edges** (*self*) → List['Edge']

**property space** (*self*) → Optional[Type['Space']]

Space this cell is a part of.

**property x** (*self*) → int

**property y** (*self*) → int

**property n\_sides** (*self*) → int

**class** donjuan.HexCell (*filled: bool = False, door\_space: Optional[DoorSpace] = None, contents: Optional[List[Any]] = None, coordinates: Optional[Tuple[int, int]] = None*)

Bases: donjuan.cell.Cell

A cell for a hexagonal grid.

#### Parameters

- **filled** (*bool, optional*) – flag indicating whether the cell is filled (default False)
- **door\_space** (*Optional[DoorSpace]*) – kind of doorway in this cell
- **contents** (*Optional[List[Any]]*) – things in this cell

**\_n\_sides** = 6

**class** donjuan.SquareCell (*filled: bool = False, door\_space: Optional[DoorSpace] = None, contents: Optional[List[Any]] = None, coordinates: Optional[Tuple[int, int]] = None*)

Bases: donjuan.cell.Cell

A cell for a square grid.

**Parameters**

- **filled** (*bool*, *optional*) – flag indicating whether the cell is filled (default `False`)
- **door\_space** (*Optional* [`DoorSpace`]) – kind of doorway in this cell
- **contents** (*Optional* [`List` [`Any`]]) – things in this cell

**\_n\_sides** = 4

**class** donjuan.Archway (*material: str = 'stone', blocked: bool = False, broken: bool = False, secret: bool = False*)

Bases: `donjuan.door_space.DoorSpace`

An archway to walk through.

**class** donjuan.Door (*secret: bool = False, locked: bool = False, closed: bool = True, jammed: bool = False, blocked: bool = False, broken: bool = False, material: str = 'wood'*)

Bases: `donjuan.door_space.DoorSpace`

A generic door.

**class** donjuan.DoorSpace (*secret: bool, locked: bool, closed: bool, jammed: bool, blocked: bool, broken: bool, material: str, name: str*)

Bases: `abc.ABC`

Abstract base class for different kinds of doors. Door spaces can have many properties, like if they are locked or blocked etc. To facilitate this logic in the generative process, these are encompassed in the attributes of a `DoorSpace`.

**\_\_slots\_\_** = ['locked', 'closed', 'jammed', 'blocked', 'secret', 'broken', 'material',

**\_\_str\_\_** (*self*)

Return `str(self)`.

**class** donjuan.Portcullis (*locked: bool = False, closed: bool = True, jammed: bool = False, broken: bool = False, material: str = 'metal'*)

Bases: `donjuan.door_space.DoorSpace`

You can look but can't touch!

**class** donjuan.Dungeon (*n\_rows: Optional[int] = 5, n\_cols: Optional[int] = 5, grid: Optional[Grid] = None, rooms: Optional[Dict[str, Room]] = None, hallways: Optional[Dict[str, Hallway]] = None, randomizers: Optional[List['Randomizer']] = None*)

**add\_room** (*self, room: Room*) → `None`

**add\_hallway** (*self, hallway: Hallway*) → `None`

**property grid** (*self*) → `Grid`

**property hallways** (*self*) → `Dict[str, Hallway]`

**property n\_cols** (*self*) → `int`

**property n\_rows** (*self*) → `int`

**property randomizers** (*self*) → `List['Randomizer']`

**property rooms** (*self*) → `Dict[str, Room]`

**randomize** (*self*) → `None`

For each item in `randomizers`, run the `Randomizer.randomize_dungeon()` method on this `dungeon`.



**emplace\_rooms** (*self*) → None

Replace the cells in the *grid* with the cells of the *rooms*.

**emplace\_space** (*self*, *space*: *Space*) → None

Replace the cells in the *grid* with the cells of the *Space*, and automatically link them with the *Edge*'s in the *Grid*.

**Parameters** *space* (*Space*) – room to emplace in the *grid*

```
class donjuan.DungeonRandomizer (room_entrance_randomizer: Optional[Randomizer] = None,
                                room_size_randomizer: Optional[Randomizer] = None,
                                room_name_randomizer: Optional[Randomizer] = None,
                                room_position_randomizer: Optional[Randomizer] = None,
                                max_num_rooms: Optional[int] = None, max_room_attempts:
                                int = 100)
```

Bases: *donjuan.randomizer.Randomizer*

Randomize a dungeon by first creating rooms and then applying room size, name, and position randomizers to sequentially generated rooms.

#### Parameters

- **room\_entrance\_randomizer** (*Optional[Randomizer]*) – randomizer for the entrances of a room. If None then default to a RoomEntrancesRandomizer.
- **room\_size\_randomizer** (*Optional[Randomizer]*) – randomizer for the room size. It must have a 'max\_size' attribute. If None then default to a RoomSizeRandomizer.
- **room\_name\_randomizer** (*Optional[Randomizer]*) – randomizer for the room name. If None default to a AlphaNumRoomName.
- **room\_position\_randomizer** (*Optional[Randomizer]*) – randomizer for the room position. If None default to a RoomPositionRandomizer.
- **max\_num\_rooms** (*Optional[int]*) – maximum number of rooms to draw, if 'None' then default to the max\_room\_attempts. See DungeonRoomRandomizer.get\_number\_of\_rooms() for details.
- **max\_room\_attempts** (*int, optional*) – default is 100. Maximum number of attempts to generate rooms.

**get\_number\_of\_rooms** (*self*, *dungeon\_n\_rows*: *int*, *dungeon\_n\_cols*: *int*) → *int*

Randomly determine the number of rooms based on the size of the incoming grid or the max\_num\_rooms attribute, whichever is less.

#### Parameters

- **dungeon\_n\_rows** (*int*) – number of rows
- **dungeon\_n\_cols** (*int*) – number of columns

**randomize\_dungeon** (*self*, *dungeon*: *Dungeon*) → None

Randomly put rooms in the dungeon.

**Parameters** *dungeon* (*Dungeon*) – dungeon to randomize the rooms of

```
class donjuan.Edge (cell1: Optional[Cell] = None, cell2: Optional[Cell] = None, has_door: bool =
                    False)
```

An edge sits between two *Cell*'s.

#### Parameters

- **cell1** (*Optional[Cell]*) – cell on one side of the edge

- **cell12** (*Optional[Cell]*) – cell on the other side of the edge
- **has\_door** (*bool, optional*) – default `False`, indicates whether this object has a door

**property** **cell11** (*self*) → *Optional[Cell]*

**property** **cell12** (*self*) → *Optional[Cell]*

**set\_cell11** (*self, cell: Cell*) → *None*

**set\_cell12** (*self, cell: Cell*) → *None*

**property** **is\_wall** (*self*) → *bool*

**class** **donjuan.Grid** (*n\_rows: int, n\_cols: int, cells: Optional[List[List[Cell]]] = None, edges: Optional[List[List[List[Edge]]] = None*)

Bases: *abc.ABC*

Abstract base class for a grid of cells. The underlying grid can either be square or hexagonal.

**get\_filled\_grid** (*self*) → *List[List[bool]]*

Obtain a 2D array of boolean values representing the `filled` state of the cells attached to the grid.

**property** **n\_rows** (*self*) → *int*

**property** **n\_cols** (*self*) → *int*

**property** **cells** (*self*) → *List[List[Cell]]*

**property** **edges** (*self*) → *List[List[List[Edge]]]*

**reset\_cell\_coordinates** (*self*) → *None*

Helper function that sets the coordinates of the cells in the grid to their index values.

**check\_edges** (*self, edges: Optional[List[List[List[Edge]]]]*) → *None*

Check the dimensions of the *edges*.

**init\_edges** (*self*) → *List[List[List[Edge]]]*

**link\_edges\_to\_cells** (*self*) → *None*

For an *Edge*, the *Edge.cell11* always points to either the left or upper *Cell*. The *Edge.cell12* always points to the right or the bottom.

**abstract link\_cells\_to\_edges** (*self*) → *None*

**class** **donjuan.HexGrid** (*n\_rows: int, n\_cols: int, cells: Optional[List[List[HexCell]]] = None*)

Bases: *donjuan.grid.Grid*

Rectangular grid of hexagonal cells. In a hex grid, the cell positions are integers, with odd rows being “offset” by half a cell size when rendered.

**cell\_type**

**link\_cells\_to\_edges** (*self*) → *None*

**class** **donjuan.SquareGrid** (*n\_rows: int, n\_cols: int, cells: Optional[List[List[SquareCell]]] = None*)

Bases: *donjuan.grid.Grid*

Rectangular grid of square cells. In a square grid, the cell positions are integers.

**cell\_type**

**link\_cells\_to\_edges** (*self*) → *None*

```
class donjuan.Hallway (ordered_cells: Optional[List[Cell]] = None, name: Union[int, str] = "")
    Bases: donjuan.space.Space
```

A hallway in a dungeon. It has a start and end cell.

#### Parameters

- **ordered\_cells** (*Optional[Sequence[Cell]]*) – ordered list of cells, where the order defines the path of the hallway. If None defaults to an empty list.
- **name** (*Union[int, str]*, *optional*) – defaults to ‘’, the name of the hallway

```
property ordered_cells (self) → List[Cell]
```

Cells that make up the path of the hallway. Does not contain extra cells that may be associated with this object (i.e. those off of the “path”). For the set of all cells, use `cells`.

```
property end_cell (self) → Cell
```

```
property start_cell (self) → Cell
```

```
append_ordered_cell_list (self, cells: List[Cell]) → None
```

Append cells in order to the hallway. To add a cell to the hallway without including it in the hallway's path, use `add_cells()`.

**Parameters** `cells` – (List[Cell]): cells to append to the hallway

```
get_coordinate_path (self) → List[Tuple[int, int]]
```

Get the coordinates of the cells that make up this path this hallway takes. Does not contain coordinates of extra cells on this object.

**Returns** coordinates of the hallway path

```
class donjuan.RandomFilled
    Bases: donjuan.randomizer.Randomizer
```

Randomly set the `filled` attribute of cells.

```
randomize_cell (self, cell: Cell) → None
```

Randomly fill the cell with probability 50%

```
randomize_grid (self, grid: Grid) → None
```

Randomly fill all cells of the grid individually

```
class donjuan.Randomizer
    Class for randomizing features of a dungeon.
```

```
randomize_cell (self, cell: Cell) → None
```

Randomize properties of the *Cell*

```
randomize_dungeon (self, dungeon: Dungeon) → None
```

Randomize properties of the *Dungeon*

```
randomize_grid (self, grid: Grid) → None
```

Randomize properties of the *Grid*

```
randomize_hallway (self, hallway: Hallway) → None
```

Randomize properties of the *Hallway*

```
randomize_room_entrances (self, room: Room, *args) → None
```

Randomize the entrances of the *Room*

```
randomize_room_size (self, room: Room, *args) → None
```

Randomize the size of the *Room*

**randomize\_room\_name** (*self*, *room*: [Room](#), \**args*) → None

Randomize the name of a *Room*

**randomize\_room\_position** (*self*, *room*: [Room](#), \**args*) → None

Randomize the position of a *Room*

**classmethod seed** (*cls*, *seed*: *Optional[int]* = None) → None

**Parameters** **seed** (*Optional[int]*) – seed passed to `random.seed()`

**class** [donjuan.BaseRenderer](#) (*scale*: *float*)

Bases: [abc.ABC](#)

Base class for rendering dungeons into images.

**property scale** (*self*) → *float*

The scale size of a single [Cell](#). The meaning differs depending on the subclass.

**abstract render** (*self*, *dungeon*: [Dungeon](#)) → None

**class** [donjuan.Renderer](#) (*scale*: *float* = 1.0)

Bases: [donjuan.renderer.BaseRenderer](#)

Default renderer for rendering dungeons into PNG files using *matplotlib*.

**Parameters** **scale** (*float*, *optional*) – size of a single cell in inches (default is 1 inch).

**render** (*self*, *dungeon*: [Dungeon](#), *file\_path*: *str* = 'rendered\_dungeon.png', *dpi*: *int* = 200, *save*: *bool* = *True*) → *Tuple*

Render the dungeon.

**Parameters**

- **dungeon** ([Dungeon](#)) – dungeon to render
- **file\_path** (*str*, *optional*) – path to save the dungeon at (default is *rendered\_dungeon.png*)
- **dpi** (*int*, *optional*) – dots per inch used to save (default is 200)
- **save** (*bool*, *optional*) – flag indicating whether to save the dungeon with `matplotlib.pyplot.savefig()`

**class** [donjuan.Room](#) (*cells*: *Optional[Set[Cell]]* = None, *name*: *Union[int, str]* = "")

Bases: [donjuan.space.Space](#)

A room in a dungeon. A room has is a named *Space*.

**class** [donjuan.AlphaNumRoomName](#)

Bases: [donjuan.randomizer.Randomizer](#)

Simple room name randomizer that names rooms as alphabetical letters. followed by a number. Rooms are sequentially named 'A0', 'B0', ... 'Z0', 'A1', 'B1', ...

**next\_name** (*self*) → *str*

**randomize\_room\_name** (*self*, *room*: [Room](#), \**args*) → None

Randomize the name of a *Room*

**class** [donjuan.RoomEntrancesRandomizer](#) (*max\_attempts*: *int* = 100)

Bases: [donjuan.randomizer.Randomizer](#)

Randomizes the number of entrances on a room. The number is picked to be the square root of the number of cells in the room divided by 2 plus 1 (N) plus a uniform random integer from 0 to N.

**gen\_num\_entrances** (*self*, *cells*: *Set[Cell]*) → *int*

**randomize\_room\_entrances** (*self*, *room*: [Room](#), \**args*) → None

Randomly open edges of cells in a *Room*. The cells in the room must already be linked to edges in a *Grid*. See `emplace_rooms()`.

---

**Note:** This algorithm does not allow for a cell in a room to have two entrances.

---

**Parameters** *room* ([Room](#)) – room to try to create entrances for

**class** `donjuan.RoomPositionRandomizer`

Bases: `donjuan.randomizer.Randomizer`

Randomly shift a room, assuming its left edge is at column 0 and its top edge is at row 0.

**randomize\_room\_position** (*self*, *room*: [Room](#), *dungeon*: [Dungeon](#)) → None

**Parameters**

- *room* ([Room](#)) – room to move around
- *dungeon* ([Dungeon](#)) – dungeon to move the room around in

**class** `donjuan.RoomSizeRandomizer` (*min\_size*: *int* = 2, *max\_size*: *int* = 4, *cell\_type*: *Type*[[Cell](#)] = [SquareCell](#))

Bases: `donjuan.randomizer.Randomizer`

Randomize the size of a *Room*.

**randomize\_room\_size** (*self*, *room*: [Room](#)) → None

Randomly determine the size of the room, and set the cells of the room to a 2D array of unfilled cells of that size.

**class** `donjuan.Space` (*cells*: *Optional*[*Set*[[Cell](#)]] = None, *name*: *Union*[*int*, *str*] = "")

Bases: `abc.ABC`

A space is a section of a dungeon composed of *Cell*'s. This object contains these cells in a ``set`` under the property *cells*. It also has a *name* and knows about any entrances to the room (a list of *Edge* objects) via the *entrances* property.

**Parameters**

- *cells* (*Optional*[*Set*[[Cell](#)]]) – cells that make up this space
- *name* (*Union*[*int*, *str*], *optional*) – defaults to "", the room name

**assign\_space\_to\_cells** (*self*) → None

Set the *space* attribute for each *Cell* to *self*.

**reset\_cell\_coordinates** (*self*) → None

**property** *cells* (*self*) → *Set*[[Cell](#)]

**property** *cell\_coordinates* (*self*) → *Set*[*Tuple*[*int*, *int*]]

**property** *name* (*self*) → *Union*[*int*, *str*]

**add\_cells** (*self*, *cells*: *Sequence*[[Cell](#)]) → None

Add cells to the set of cells in this space. Cells are added to both the *cells* set and the *cell\_coordinates* set.

**Parameters** *cells* (*Sequence*[[Cell](#)]) – any iterable collection of cells

**overlaps** (*self*, *other*: [Space](#)) → bool

Compare the cells of this space to the other space to determine whether they overlap or not. Note, this algorithm is  $O(N)$  where  $N$  is the number of cells in this space, since set lookup is  $O(1)$ .

**Parameters** *other* ([Space](#)) – other space to check against

**Returns** True if they overlap, False if not

**set\_name** (*self*, *name*: [Union\[int, str\]](#)) → None

**shift\_vertical** (*self*, *n*: int) → None

Change the y coordinates of all [cells](#) by *n*.

**Parameters** *n* (int) – number to increment vertical position of cells

**shift\_horizontal** (*self*, *n*: int) → None

Change the x coordinates of all [cells](#) by *n*.

**Parameters** *n* (int) – number to increment horizontal position of cells

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## PYTHON MODULE INDEX

### d

- `donjuan`, [7](#)
- `donjuan.cell`, [7](#)
- `donjuan.door_space`, [8](#)
- `donjuan.dungeon`, [9](#)
- `donjuan.dungeon_randomizer`, [10](#)
- `donjuan.edge`, [11](#)
- `donjuan.face`, [11](#)
- `donjuan.grid`, [12](#)
- `donjuan.hallway`, [13](#)
- `donjuan.randomizer`, [14](#)
- `donjuan.renderer`, [15](#)
- `donjuan.room`, [16](#)
- `donjuan.room_randomizer`, [16](#)
- `donjuan.space`, [17](#)



## Symbols

[\\_\\_docs\\_\\_](#) (in module *donjuan*), 19  
[\\_\\_getitem\\_\\_](#)() (*donjuan.face.Faces* method), 12  
[\\_\\_len\\_\\_](#)() (*donjuan.face.Faces* method), 12  
[\\_\\_slots\\_\\_](#) (*donjuan.DoorSpace* attribute), 20  
[\\_\\_slots\\_\\_](#) (*donjuan.door\_space.DoorSpace* attribute), 9  
[\\_\\_str\\_\\_](#)() (*donjuan.DoorSpace* method), 20  
[\\_\\_str\\_\\_](#)() (*donjuan.door\_space.DoorSpace* method), 9  
[\\_\\_version\\_\\_](#) (in module *donjuan*), 19  
[\\_init\\_faces](#)() (*donjuan.face.Faces* method), 12  
[\\_n\\_sides](#) (*donjuan.HexCell* attribute), 19  
[\\_n\\_sides](#) (*donjuan.SquareCell* attribute), 20  
[\\_n\\_sides](#) (*donjuan.cell.HexCell* attribute), 8  
[\\_n\\_sides](#) (*donjuan.cell.SquareCell* attribute), 8

## A

[add\\_cells](#)() (*donjuan.Space* method), 25  
[add\\_cells](#)() (*donjuan.space.Space* method), 17  
[add\\_hallway](#)() (*donjuan.Dungeon* method), 20  
[add\\_hallway](#)() (*donjuan.dungeon.Dungeon* method), 9  
[add\\_room](#)() (*donjuan.Dungeon* method), 20  
[add\\_room](#)() (*donjuan.dungeon.Dungeon* method), 9  
[AlphaNumRoomName](#) (class in *donjuan*), 24  
[AlphaNumRoomName](#) (class in *donjuan.room\_randomizer*), 16  
[append\\_ordered\\_cell\\_list](#)() (*donjuan.Hallway* method), 23  
[append\\_ordered\\_cell\\_list](#)() (*donjuan.hallway.Hallway* method), 14  
[Archway](#) (class in *donjuan*), 20  
[Archway](#) (class in *donjuan.door\_space*), 9  
[assign\\_space\\_to\\_cells](#)() (*donjuan.Space* method), 25  
[assign\\_space\\_to\\_cells](#)() (*donjuan.space.Space* method), 17

## B

[BareFace](#) (class in *donjuan.face*), 12  
[BaseRenderer](#) (class in *donjuan*), 24

[BaseRenderer](#) (class in *donjuan.renderer*), 15

## C

[Cell](#) (class in *donjuan*), 19  
[Cell](#) (class in *donjuan.cell*), 7  
[cell11](#)() (*donjuan.Edge* property), 22  
[cell11](#)() (*donjuan.edge.Edge* property), 11  
[cell12](#)() (*donjuan.Edge* property), 22  
[cell12](#)() (*donjuan.edge.Edge* property), 11  
[cell\\_coordinates](#)() (*donjuan.Space* property), 25  
[cell\\_coordinates](#)() (*donjuan.space.Space* property), 17  
[cell\\_type](#) (*donjuan.grid.HexGrid* attribute), 13  
[cell\\_type](#) (*donjuan.grid.SquareGrid* attribute), 13  
[cell\\_type](#) (*donjuan.HexGrid* attribute), 22  
[cell\\_type](#) (*donjuan.SquareGrid* attribute), 22  
[cells](#)() (*donjuan.Grid* property), 22  
[cells](#)() (*donjuan.grid.Grid* property), 13  
[cells](#)() (*donjuan.Space* property), 25  
[cells](#)() (*donjuan.space.Space* property), 17  
[check\\_edges](#)() (*donjuan.Grid* method), 22  
[check\\_edges](#)() (*donjuan.grid.Grid* method), 13  
[coordinates](#)() (*donjuan.Cell* property), 19  
[coordinates](#)() (*donjuan.cell.Cell* property), 7

## D

[donjuan](#)  
     module, 7  
[donjuan.cell](#)  
     module, 7  
[donjuan.door\\_space](#)  
     module, 8  
[donjuan.dungeon](#)  
     module, 9  
[donjuan.dungeon\\_randomizer](#)  
     module, 10  
[donjuan.edge](#)  
     module, 11  
[donjuan.face](#)  
     module, 11  
[donjuan.grid](#)  
     module, 12

donjuan.hallway  
     module, 13  
 donjuan.randomizer  
     module, 14  
 donjuan.renderer  
     module, 15  
 donjuan.room  
     module, 16  
 donjuan.room\_randomizer  
     module, 16  
 donjuan.space  
     module, 17  
 Door (class in donjuan), 20  
 Door (class in donjuan.door\_space), 9  
 DoorFace (class in donjuan.face), 12  
 DoorSpace (class in donjuan), 20  
 DoorSpace (class in donjuan.door\_space), 8  
 Dungeon (class in donjuan), 20  
 Dungeon (class in donjuan.dungeon), 9  
 DungeonRandomizer (class in donjuan), 21  
 DungeonRandomizer (class in don-  
     juan.dungeon\_randomizer), 10

## E

Edge (class in donjuan), 21  
 Edge (class in donjuan.edge), 11  
 edges () (donjuan.Cell property), 19  
 edges () (donjuan.cell.Cell property), 7  
 edges () (donjuan.Grid property), 22  
 edges () (donjuan.grid.Grid property), 13  
 emplace\_rooms () (donjuan.Dungeon method), 20  
 emplace\_rooms () (donjuan.dungeon.Dungeon  
     method), 9  
 emplace\_space () (donjuan.Dungeon method), 21  
 emplace\_space () (donjuan.dungeon.Dungeon  
     method), 10  
 end\_cell () (donjuan.Hallway property), 23  
 end\_cell () (donjuan.hallway.Hallway property), 14

## F

Face (class in donjuan.face), 12  
 Faces (class in donjuan.face), 12  
 faces () (donjuan.face.Faces property), 12

## G

gen\_num\_entrances () (don-  
     juan.room\_randomizer.RoomEntrancesRandomizer  
     method), 17  
 gen\_num\_entrances () (don-  
     juan.RoomEntrancesRandomizer method),  
     24  
 get\_coordinate\_path () (donjuan.Hallway  
     method), 23

get\_coordinate\_path () (don-  
     juan.hallway.Hallway method), 14  
 get\_filled\_grid () (donjuan.Grid method), 22  
 get\_filled\_grid () (donjuan.grid.Grid method),  
     13  
 get\_number\_of\_rooms () (don-  
     juan.dungeon\_randomizer.DungeonRandomizer  
     method), 10  
 get\_number\_of\_rooms () (don-  
     juan.DungeonRandomizer method), 21  
 Grid (class in donjuan), 22  
 Grid (class in donjuan.grid), 12  
 grid () (donjuan.Dungeon property), 20  
 grid () (donjuan.dungeon.Dungeon property), 9

## H

Hallway (class in donjuan), 22  
 Hallway (class in donjuan.hallway), 13  
 hallways () (donjuan.Dungeon property), 20  
 hallways () (donjuan.dungeon.Dungeon property), 9  
 HexCell (class in donjuan), 19  
 HexCell (class in donjuan.cell), 8  
 HexFaces (class in donjuan.face), 12  
 HexGrid (class in donjuan), 22  
 HexGrid (class in donjuan.grid), 13

## I

init\_edges () (donjuan.Grid method), 22  
 init\_edges () (donjuan.grid.Grid method), 13  
 is\_wall () (donjuan.Edge property), 22  
 is\_wall () (donjuan.edge.Edge property), 11

## L

link\_cells\_to\_edges () (donjuan.Grid method),  
     22  
 link\_cells\_to\_edges () (donjuan.grid.Grid  
     method), 13  
 link\_cells\_to\_edges () (donjuan.grid.HexGrid  
     method), 13  
 link\_cells\_to\_edges () (don-  
     juan.grid.SquareGrid method), 13  
 link\_cells\_to\_edges () (donjuan.HexGrid  
     method), 22  
 link\_cells\_to\_edges () (donjuan.SquareGrid  
     method), 22  
 link\_edges\_to\_cells () (donjuan.Grid method),  
     22  
 link\_edges\_to\_cells () (donjuan.grid.Grid  
     method), 13

## M

module  
     donjuan, 7

donjuan.cell, 7  
 donjuan.door\_space, 8  
 donjuan.dungeon, 9  
 donjuan.dungeon\_randomizer, 10  
 donjuan.edge, 11  
 donjuan.face, 11  
 donjuan.grid, 12  
 donjuan.hallway, 13  
 donjuan.randomizer, 14  
 donjuan.renderer, 15  
 donjuan.room, 16  
 donjuan.room\_randomizer, 16  
 donjuan.space, 17

## N

n\_cols() (*donjuan.Dungeon property*), 20  
 n\_cols() (*donjuan.dungeon.Dungeon property*), 9  
 n\_cols() (*donjuan.Grid property*), 22  
 n\_cols() (*donjuan.grid.Grid property*), 13  
 n\_rows() (*donjuan.Dungeon property*), 20  
 n\_rows() (*donjuan.dungeon.Dungeon property*), 9  
 n\_rows() (*donjuan.Grid property*), 22  
 n\_rows() (*donjuan.grid.Grid property*), 13  
 n\_sides() (*donjuan.Cell property*), 19  
 n\_sides() (*donjuan.cell.Cell property*), 8  
 name() (*donjuan.Space property*), 25  
 name() (*donjuan.space.Space property*), 17  
 next\_name() (*donjuan.AlphaNumRoomName method*), 24  
 next\_name() (*donjuan.room\_randomizer.AlphaNumRoomName method*), 16

## O

ordered\_cells() (*donjuan.Hallway property*), 23  
 ordered\_cells() (*donjuan.hallway.Hallway property*), 14  
 overlaps() (*donjuan.Space method*), 25  
 overlaps() (*donjuan.space.Space method*), 18

## P

Portcullis (*class in donjuan*), 20  
 Portcullis (*class in donjuan.door\_space*), 9

## R

RandomFilled (*class in donjuan*), 23  
 RandomFilled (*class in donjuan.randomizer*), 15  
 randomize() (*donjuan.Dungeon method*), 20  
 randomize() (*donjuan.dungeon.Dungeon method*), 9  
 randomize\_cell() (*donjuan.RandomFilled method*), 23  
 randomize\_cell() (*donjuan.Randomizer method*), 23

randomize\_cell() (*donjuan.randomizer.RandomFilled method*), 15  
 randomize\_cell() (*donjuan.randomizer.Randomizer method*), 14  
 randomize\_dungeon() (*donjuan.dungeon\_randomizer.DungeonRandomizer method*), 11  
 randomize\_dungeon() (*donjuan.DungeonRandomizer method*), 21  
 randomize\_dungeon() (*donjuan.Randomizer method*), 23  
 randomize\_dungeon() (*donjuan.randomizer.Randomizer method*), 14  
 randomize\_grid() (*donjuan.RandomFilled method*), 23  
 randomize\_grid() (*donjuan.Randomizer method*), 23  
 randomize\_grid() (*donjuan.randomizer.RandomFilled method*), 15  
 randomize\_grid() (*donjuan.randomizer.Randomizer method*), 14  
 randomize\_hallway() (*donjuan.Randomizer method*), 23  
 randomize\_hallway() (*donjuan.randomizer.Randomizer method*), 14  
 randomize\_room\_entrances() (*donjuan.Randomizer method*), 23  
 randomize\_room\_entrances() (*donjuan.randomizer.Randomizer method*), 14  
 randomize\_room\_entrances() (*donjuan.room\_randomizer.RoomEntrancesRandomizer method*), 17  
 randomize\_room\_entrances() (*donjuan.RoomEntrancesRandomizer method*), 24  
 randomize\_room\_name() (*donjuan.AlphaNumRoomName method*), 24  
 randomize\_room\_name() (*donjuan.Randomizer method*), 23  
 randomize\_room\_name() (*donjuan.randomizer.Randomizer method*), 14  
 randomize\_room\_name() (*donjuan.room\_randomizer.AlphaNumRoomName method*), 16  
 randomize\_room\_position() (*donjuan.Randomizer method*), 24  
 randomize\_room\_position() (*donjuan.randomizer.Randomizer method*), 14  
 randomize\_room\_position() (*donjuan.room\_randomizer.RoomPositionRandomizer method*), 16  
 randomize\_room\_position() (*don-*)

[juan.RoomPositionRandomizer](#) (*method*), [25](#)  
[randomize\\_room\\_size\(\)](#) (*donjuan.Randomizer method*), [23](#)  
[randomize\\_room\\_size\(\)](#) (*donjuan.randomizer.Randomizer method*), [14](#)  
[randomize\\_room\\_size\(\)](#) (*donjuan.room\_randomizer.RoomSizeRandomizer method*), [16](#)  
[randomize\\_room\\_size\(\)](#) (*donjuan.RoomSizeRandomizer method*), [25](#)  
[Randomizer](#) (*class in donjuan*), [23](#)  
[Randomizer](#) (*class in donjuan.randomizer*), [14](#)  
[randomizers\(\)](#) (*donjuan.Dungeon property*), [20](#)  
[randomizers\(\)](#) (*donjuan.dungeon.Dungeon property*), [9](#)  
[render\(\)](#) (*donjuan.BaseRenderer method*), [24](#)  
[render\(\)](#) (*donjuan.Renderer method*), [24](#)  
[render\(\)](#) (*donjuan.renderer.BaseRenderer method*), [15](#)  
[render\(\)](#) (*donjuan.renderer.Renderer method*), [15](#)  
[Renderer](#) (*class in donjuan*), [24](#)  
[Renderer](#) (*class in donjuan.renderer*), [15](#)  
[reset\\_cell\\_coordinates\(\)](#) (*donjuan.Grid method*), [22](#)  
[reset\\_cell\\_coordinates\(\)](#) (*donjuan.grid.Grid method*), [13](#)  
[reset\\_cell\\_coordinates\(\)](#) (*donjuan.Space method*), [25](#)  
[reset\\_cell\\_coordinates\(\)](#) (*donjuan.space.Space method*), [17](#)  
[Room](#) (*class in donjuan*), [24](#)  
[Room](#) (*class in donjuan.room*), [16](#)  
[RoomEntrancesRandomizer](#) (*class in donjuan*), [24](#)  
[RoomEntrancesRandomizer](#) (*class in donjuan.room\_randomizer*), [17](#)  
[RoomPositionRandomizer](#) (*class in donjuan*), [25](#)  
[RoomPositionRandomizer](#) (*class in donjuan.room\_randomizer*), [16](#)  
[rooms\(\)](#) (*donjuan.Dungeon property*), [20](#)  
[rooms\(\)](#) (*donjuan.dungeon.Dungeon property*), [9](#)  
[RoomSizeRandomizer](#) (*class in donjuan*), [25](#)  
[RoomSizeRandomizer](#) (*class in donjuan.room\_randomizer*), [16](#)  
[set\\_cell11\(\)](#) (*donjuan.Edge method*), [22](#)  
[set\\_cell11\(\)](#) (*donjuan.edge.Edge method*), [11](#)  
[set\\_cell12\(\)](#) (*donjuan.Edge method*), [11](#)  
[set\\_coordinates\(\)](#) (*donjuan.Cell method*), [19](#)  
[set\\_coordinates\(\)](#) (*donjuan.cell.Cell method*), [7](#)  
[set\\_edges\(\)](#) (*donjuan.Cell method*), [19](#)  
[set\\_edges\(\)](#) (*donjuan.cell.Cell method*), [7](#)  
[set\\_name\(\)](#) (*donjuan.Space method*), [26](#)  
[set\\_name\(\)](#) (*donjuan.space.Space method*), [18](#)  
[set\\_space\(\)](#) (*donjuan.Cell method*), [19](#)  
[set\\_space\(\)](#) (*donjuan.cell.Cell method*), [7](#)  
[set\\_x\(\)](#) (*donjuan.Cell method*), [19](#)  
[set\\_x\(\)](#) (*donjuan.cell.Cell method*), [7](#)  
[set\\_y\(\)](#) (*donjuan.Cell method*), [19](#)  
[set\\_y\(\)](#) (*donjuan.cell.Cell method*), [7](#)  
[shift\\_horizontal\(\)](#) (*donjuan.Space method*), [26](#)  
[shift\\_horizontal\(\)](#) (*donjuan.space.Space method*), [18](#)  
[shift\\_vertical\(\)](#) (*donjuan.Space method*), [26](#)  
[shift\\_vertical\(\)](#) (*donjuan.space.Space method*), [18](#)  
[Space](#) (*class in donjuan*), [25](#)  
[Space](#) (*class in donjuan.space*), [17](#)  
[space\(\)](#) (*donjuan.Cell property*), [19](#)  
[space\(\)](#) (*donjuan.cell.Cell property*), [7](#)  
[SquareCell](#) (*class in donjuan*), [19](#)  
[SquareCell](#) (*class in donjuan.cell*), [8](#)  
[SquareFaces](#) (*class in donjuan.face*), [12](#)  
[SquareGrid](#) (*class in donjuan*), [22](#)  
[SquareGrid](#) (*class in donjuan.grid*), [13](#)  
[start\\_cell\(\)](#) (*donjuan.Hallway property*), [23](#)  
[start\\_cell\(\)](#) (*donjuan.hallway.Hallway property*), [14](#)

## X

[x\(\)](#) (*donjuan.Cell property*), [19](#)  
[x\(\)](#) (*donjuan.cell.Cell property*), [8](#)

## Y

[y\(\)](#) (*donjuan.Cell property*), [19](#)  
[y\(\)](#) (*donjuan.cell.Cell property*), [8](#)

## S

[scale\(\)](#) (*donjuan.BaseRenderer property*), [24](#)  
[scale\(\)](#) (*donjuan.renderer.BaseRenderer property*), [15](#)  
[seed\(\)](#) (*donjuan.Randomizer class method*), [24](#)  
[seed\(\)](#) (*donjuan.randomizer.Randomizer class method*), [15](#)  
[set\\_cell11\(\)](#) (*donjuan.Edge method*), [22](#)  
[set\\_cell11\(\)](#) (*donjuan.edge.Edge method*), [11](#)  
[set\\_cell12\(\)](#) (*donjuan.Edge method*), [22](#)