

Question: How can I align 8 png figures in a 4*4 grid in a readme github file?

To align **8 PNG images** in a clean **4×4-style grid (4 columns × 2 rows)** in a GitHub README, the most reliable method is **HTML tables**.

Here's a ready-to-paste template you can use:



4×4 (4-column) Image Grid for README.md

```
<table align="center">
<tr>
  <td align="center"><br><b>Fig 1</b></td>
  <td align="center"><br><b>Fig 2</b></td>
  <td align="center"><br><b>Fig 3</b></td>
  <td align="center"><br><b>Fig 4</b></td>
</tr>
<tr>
  <td align="center"><br><b>Fig 5</b></td>
  <td align="center"><br><b>Fig 6</b></td>
  <td align="center"><br><b>Fig 7</b></td>
  <td align="center"><br><b>Fig 8</b></td>
</tr>
</table>
```



Instructions

- Replace fig1.png ... fig8.png with your actual filenames.
- Adjust width="200" as needed (typical 180–250 works cleanly).
- GitHub fully supports HTML tables inside markdown.

🎯 Optional:

Even Cleaner, No Captions

If you want only images:

```
<table>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
</table>
```

If you want, tell me your actual filenames and I'll generate the exact grid for you.

Question 2:

Imagine I have a grid of x by x python list of lists. How can I better print the outputs for better visuals and debugging?

Here are the **cleanest and most useful ways** to print and visually debug any **Python grid (list of lists)** — independent of your agent model.

I'll show you a simple example grid:

```
grid = [
  [0, 1, 0, 2],
  [2, 0, 1, 0],
  [0, 3, 0, 1],
  [1, 0, 0, 0]
]
```

Below are **7 excellent techniques** you can use for readable debugging.



1. Clean row-by-row pretty print

The most readable default.

```
def print_grid(grid):
    for row in grid:
        print(" ".join(str(x) for x in row))
```

Output:

```
0 1 0 2
2 0 1 0
0 3 0 1
1 0 0 0
```



2. Use fixed-width formatting (best for alignment)

Great for grids with multi-digit values.

```
def print_aligned(grid):
    for row in grid:
        print(" ".join(f"{x:3}" for x in row))
```

Output:

```
0    1    0    2
2    0    1    0
0    3    0    1
1    0    0    0
```



3. Replace values with symbols

Useful for debugging occupancy maps, walls, empty cells, etc.

```
SYMBOLS = {0: ".", 1: "A", 2: "B", 3: "C"}  
  
def print_symbolic(grid):  
    for row in grid:  
        print("".join(SYMBOLS[x] for x in row))
```

Output:

```
.A.B  
B.A.  
.C.A  
A...
```

Very compact and easy to see patterns.



4. Print only non-empty cells (for sparse grids)

If most cells are empty:

```
def print_nonempty(grid):  
    for y, row in enumerate(grid):  
        for x, val in enumerate(row):  
            if val != 0:  
                print(f"({x}, {y}) → {val}")
```

Output:

```
(1, 0) → 1  
(3, 0) → 2
```

```
(0,1) → 2
```

```
...
```

(Perfect for sparse data.)



5. Print with borders (looks like a table)

Nice for reports or debugging:

```
def print_with_borders(grid):
    print("+" * len(grid[0]) + "+")
    for row in grid:
        print(" " + " ".join(str(x) for x in row) + " |")
    print("+" * len(row) + "+")
```

Output:

```
+---+---+---+---+
| 0 | 1 | 0 | 2 |
+---+---+---+---+
| 2 | 0 | 1 | 0 |
+---+---+---+---+
...
```



6. Use pprint (Python's pretty printer)

Better for nested structures, though less grid-like:

```
from pprint import pprint
pprint(grid)
```



7. Visual debug: Convert to NumPy array

If the grid is numeric:

```
import numpy as np  
print(np.array(grid))
```

Output:

```
[ [0 1 0 2]  
[2 0 1 0]  
[0 3 0 1]  
[1 0 0 0]]
```

This is a **very clean** visual representation.



Recommended for complex grid debugging

If your grid is large (like 75×75), the best two methods are:



np.array(grid)

Super clean overview.

✓ **symbolic map** (

.

for empty,

#

for non-empty)

Perfect for visually tracking where events happen.

If you want, I can generate a **debugging helper class** with all these tools built in so you can drop it directly into your project.