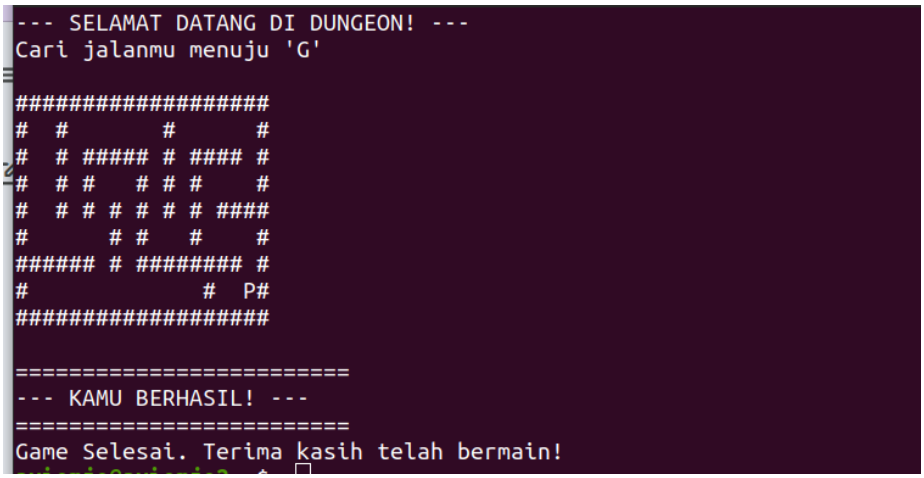
Game pake array (list) di terminal

Figure 1: alt text



Clues TLDR (Kodingan dari Nol)

Ini adalah langkah-langkah ringkas yang harus Anda lakukan di *script* Python Anda, tanpa fungsi, dalam urutan eksekusi:

1. **`import os` dan `import time`** (meski `time` tidak digunakan di kode Anda, `os` penting untuk `clear`).
2. **Deklarasikan `MAP_DATA`** sebagai *list of strings*.
3. **Ubah ke *Grid*:** `map_grid = [list(row) for row in MAP_DATA]`.
4. **Inisialisasi Posisi:** Gunakan *loop* `for` untuk mencari dan menyimpan nilai `player_y` dan `player_x`.
5. **Setel `VIEW_RADIUS`** dan `is_running = True`.
6. **Mulai *Game Loop*:** `while is_running:`
   - Di dalam *loop*, letakkan *clear screen* (`os.system(...)`).
   - Letakkan *nested loop* `for y in range(map_height): for x in range(map_width):` untuk melakukan **FOV Render**. Hitung jarak (`max(abs(...))`) dan cetak '?' atau konten aslinya.
   - Letakkan baris **input** (`key = input(...)`).
   - Tulis *block* `if/elif` untuk menghitung `new_y` dan `new_x`.
   - Tulis *block* **validasi** (cek batas, cek '#', cek 'G').
   - Tulis *block* **update state** (ganti posisi 'P' di `map_grid`, dan perbarui `player_y`/`player_x`).
7. Setelah `while` selesai, cetak pesan "Game Selesai".

---

1. Representasi Data (State Dunia)

Setiap *game* membutuhkan 'dunia' untuk hidup, dan dalam kasus ini, dunia itu adalah **peta labirin**.

Konsep: List of Strings atau List of Lists

- **Peta sebagai Data:** Cara termudah untuk merepresentasikan peta 2D di Python adalah menggunakan *list* dari *strings* atau *list* dari *lists* (sering disebut *grid* atau matriks).
  - **String:** `MAP_DATA = ["###", "#P#", "###"]`. Ini bagus untuk deklarasi awal karena mudah dibaca.
  - **List of Lists:** Jika Anda ingin **mengubah** isi peta (misalnya, menghapus jejak pemain), Anda harus mengubah *list of strings* menjadi *list of lists* (misal: `map_grid = [list(row) for row in MAP_DATA]`). Karena karakter dalam *string* tidak dapat diubah (immutable), Anda perlu menggunakan *list* di mana setiap elemen (karakter) dapat diubah (mutable).

Penerapan: Menyimpan Posisi Pemain

Selain peta, Anda perlu menyimpan **posisi pemain** saat ini. Posisi 2D selalu direpresentasikan sebagai sepasang koordinat, biasanya `(y, x)` atau `(baris, kolom)`.

- **Awal:** Cari karakter 'P' di peta awal untuk mendapatkan `player_y` dan `player_x`.
- **Pergerakan:** Setelah pemain bergerak, Anda akan memperbarui nilai `player_y` dan `player_x`.

---

2. Game Loop (Jantung Permainan)

Semua yang terjadi dalam *game* dilakukan secara berulang. Ini disebut **Game Loop**.

Konsep: `while` loop

- *Game loop* hanyalah sebuah perulangan `while` tak terbatas yang terus berjalan selama *game* aktif.

```
# Inisialisasi:
is_running = True
# Mulai Loop:
while is_running:
    # 1. Input (Tunggu perintah pemain)
    # 2. Update (Hitung posisi baru)
    # 3. Render (Tampilkan peta baru)

    # Kondisi berhenti:
    if player_menang_atau_keluar:
        is_running = False
```

---

3. Siklus I-U-R (Input, Update, Render)

Di dalam *Game Loop* tersebut, terjadi tiga langkah kritis di setiap 'detik' permainan:

A. Input (Mendengar Perintah)

Gunakan `key = input("Gerak (w/a/s/d): ")` untuk menerima input dari pemain. Perintah ini akan menentukan arah gerakan.

B. Update (Perubahan State)

Bagian ini adalah *otak* permainan, di mana Anda menghitung *state* baru berdasarkan input.

1. **Hitung Posisi Baru:** Berdasarkan input (w/a/s/d), hitung calon koordinat baru `new_y` dan `new_x`.
   - Contoh: Jika input adalah 'w' (atas), maka `new_y = player_y - 1`.
2. **Validasi:** Periksa apakah posisi baru itu valid:
   - Apakah `(new_y, new_x)` di luar batas peta? **Jika ya, batalkan gerakan.**
   - Apakah sel target (`map_grid[new_y][new_x]`) berisi '#' (tembok)? **Jika ya, batalkan gerakan.**
3. **Aplikasi Perubahan:** Jika valid, terapkan gerakan:
   - Setel sel lama ke ruang kosong: `map_grid[player_y][player_x] = ' '`.
   - Setel sel baru ke pemain: `map_grid[new_y][new_x] = 'P'`.
   - Perbarui koordinat pemain: `player_y, player_x = new_y, new_x`.

C. Render (Menampilkan Dunia)

Inilah saatnya mencetak peta ke layar. Karena Anda ingin efek **Field of View (FOV)**, Anda perlu melakukan *rendering* yang cerdas.

1. **Membersihkan Layar:** Panggil `os.system('clear')` di awal setiap *render* untuk memberikan efek animasi bergerak.
2. **Iterasi Peta:** *Loop* melalui setiap sel `(y, x)` dalam `map_grid`.
3. **Hitung Jarak (FOV):** Untuk setiap sel `(y, x)`, hitung jaraknya dari `(player_y, player_x)`.
   - Anda bisa menggunakan **Jarak Chebysheff** (jarak maksimum di antara sumbu-X dan sumbu-Y): `distance = max(abs(y - player_y), abs(x - player_x))`.
4. **Tampilkan:**
   - Jika `distance <= VIEW_RADIUS`, cetak isi sel sebenarnya (`map_grid[y][x]`).
   - Jika jaraknya lebih jauh, cetak simbol tersembunyi (misalnya, '?').

5. Cetak baris demi baris, lalu tambahkan *prompt* input di bawahnya.
