|  |  |
| --- | --- |
|  |  |
| Nama Mahasiswa/NIM | Andhika Maulana Shiddiq (301220024) |
| Judul Tugas | Pemandangan Sederhana menggunakan Scripting Python di Blender dengan bantuan AI |
| Tahun | 2024 |

|  |  |
| --- | --- |
| **JUDUL TUGAS** | |
|  | **Teori Pendukung** |
| 1. Python   Python merupakan bahasa pemrograman komputer yang biasa dipakai untuk membangun situs, software/aplikasi, mengotomatiskan tugas dan melakukan analisis data. Bahasa pemrograman ini termasuk bahasa tujuan umum. Artinya, ia bisa digunakan untuk membuat berbagai program berbeda, bukan khusus untuk masalah tertentu saja.  Karena sifatnya yang serba guna dan mudah digunakan, ia menjadi bahasa pemrograman yang paling banyak digunakan. Terutama untuk mereka yang masih pemula.  Berdasarkan survei pengembang Stack Overflow tahun 2022, Python menjadi bahasa pemrograman terpopuler keempat. Sebanyak hampir 50% dari responden mengatakan bahwa mereka menggunakan hampir setengah dari waktu kerja mereka dengan menggunakan bahasa pemrograman ini.  Nama Python sendiri berasal dari Monty Python. Ketika Guido van Rossum membuatnya, dia juga sedang membaca skrip Sirkus Terbang Monty Python BBC. Menurutnya nama itu singkat dan sedikit misterius. Karena itulah, sang kreator memilih menggunakan nama tersebut untuk bahasa pemrograman yang dibuatnya itu.   1. Rantai Markov   Rantai Markov adalah model stokastik yang menggambarkan urutan kejadian yang mungkin terjadi, di mana probabilitas setiap kejadian hanya bergantung pada keadaan yang dicapai pada kejadian sebelumnya. Karakteristik ini sering disebut sebagai “tanpa ingatan” atau properti Markov, yang berarti keadaan proses di masa depan hanya bergantung pada keadaan saat ini dan bukan pada bagaimana proses tersebut sampai pada keadaan saat ini.  Rantai Markov dapat dikategorikan ke dalam dua jenis berdasarkan sifat waktu: rantai Markov waktu diskrit (DTMC) dan rantai Markov waktu kontinu (CTMC). Dalam DTMC, perubahan keadaan dipertimbangkan pada langkah waktu diskrit, sedangkan dalam CTMC, perubahan keadaan dapat terjadi kapan saja.  Ruang keadaan dari rantai Markov, yang merupakan himpunan dari semua keadaan yang mungkin terjadi, dapat berupa apa saja: huruf, angka, kondisi cuaca, skor bisbol, atau kinerja saham. Transisi dari satu keadaan ke keadaan lainnya diatur oleh matriks transisi, yang mewakili distribusi probabilitas transisi keadaan. Jumlah probabilitas di setiap baris matriks adalah satu, menyiratkan bahwa ini adalah matriks stokastik.  Rantai Markov memiliki berbagai macam aplikasi di berbagai bidang. Rantai Markov digunakan dalam model statistik proses dunia nyata, seperti mempelajari sistem cruise control pada kendaraan bermotor, antrean atau antrean pelanggan yang tiba di bandara, nilai tukar mata uang, dan dinamika populasi hewan. Mereka juga merupakan dasar untuk metode simulasi stokastik umum yang dikenal sebagai rantai Markov Monte Carlo, yang digunakan untuk mensimulasikan pengambilan sampel dari distribusi probabilitas yang kompleks. Dalam ilmu data, aplikasi umum dari rantai Markov adalah prediksi teks, yang biasa digunakan dalam industri teknologi oleh perusahaan seperti Google, LinkedIn, dan Instagram.  Konsep rantai Markov dikaitkan dengan ahli matematika Rusia, Andrey Markov, yang memelopori bidang teori probabilitas ini. Rantai Markov pertama kali diperkenalkan dalam makalahnya pada tahun 1906, yang mempelajari proses stokastik yang melibatkan urutan peristiwa acak atau keadaan yang tidak independen satu sama lain. Karyanya meletakkan dasar bagi cabang baru teori probabilitas yang berfokus pada sifat-sifat sistem yang mengalami transisi dari satu keadaan ke keadaan lain dengan cara yang mirip rantai. | |
|  | **Alat Dan Bahan** |
| * PC atau Laptop * Browser * Software Screen Recording (OBS (Open Broadcaster Software)) * Website Google Colab | |
|  | **Tutorial** |
| 1. Buka Blender 2. Ke Google Colab 3. Tambahkan Kode 4. Jalankan Kode     Penjelasan Kode:  “  import numpy as np  from IPython.display import Math  import sympy as sym  “  Kita mengimpor library yang dibutuhkan:   * numpy: Untuk perhitungan matriks dan operasi linier. * Math dari IPython.display: Untuk menampilkan persamaan atau hasil dalam format matematis jika dibutuhkan. * sympy: Untuk simbolisasi persamaan matematis.   “  P = np.array([[0.5, 0.5],                [0.5, 0.5]])  print("Matriks Transisi P:")  print(P)  print(P)  “  Matriks transisi P adalah representasi probabilitas perpindahan antara kategori waktu cepat dan lambat   * + Baris pertama [0.67, 0.33] berarti:     - Jika **Cepat** sekarang, maka:       * 67% tetap **Cepat**.       * 33% berubah menjadi **Lambat**.   + Baris kedua [0.5, 0.5] berarti:     - Jika **Lambat** sekarang, maka:       * 50% berubah menjadi **Cepat**.       * 50% tetap **Lambat**.   “  # Kondisi awal (asumsi: mulai dengan waktu cepat)  initial\_state = np.array([1, 0])  print("Kondisi Awal:")  print(initial\_state)  “  initial\_state adalah vektor probabilitas kondisi awal.   * [1, 0] artinya sesi pertama dimulai dengan kondisi **Cepat** dengan probabilitas 100% (**Cepat** pasti terjadi).   Jika berdasarkan data lain, kondisi awal bisa menggunakan proporsi, seperti [0.6, 0.4].  “  # Prediksi sesi 2  session\_2 = np.dot(initial\_state, P)  print("Probabilitas pada Sesi 2:")  print(session\_2)  “  Perkalian matriks digunakan untuk menghitung distribusi probabilitas sesi berikutnya.   * initial\_state × P menghasilkan distribusi probabilitas untuk sesi ke-2. * Hasilnya adalah [0.67, 0.33], artinya:   + Ada peluang 67% untuk **Cepat**.   + Ada peluang 33% untuk **Lambat**.   “  # Prediksi sesi 3  session\_3 = np.dot(session\_2, P)  print("Probabilitas pada Sesi 3:")  print(session\_3)  “  Distribusi probabilitas sesi 2 (session\_2) digunakan sebagai input untuk prediksi sesi 3.  Perkalian matriks session\_2 × P menghasilkan distribusi probabilitas sesi ke-3.  “  # Iterasi hingga mencapai steady state  tolerance = 0.001  current\_state = session\_3  iteration = 3  while True:      next\_state = np.dot(current\_state, P)      print(f"Probabilitas pada Sesi {iteration + 1}: {next\_state}")      # Cek apakah perubahan sudah di bawah toleransi      if np.all(np.abs(next\_state - current\_state) < tolerance):          print(f"Steady state tercapai pada sesi {iteration + 1}")          break      # Update state untuk iterasi berikutnya      current\_state = next\_state      iteration += 1  “  Iterasi dilakukan untuk menghitung distribusi probabilitas dari sesi ke-4, ke-5, dan seterusnya hingga mencapai **steady state**.  **Steady state** tercapai jika perubahan antara distribusi probabilitas sesi saat ini dan sesi berikutnya lebih kecil dari nilai toleransi (tolerance), yaitu 0.001.  Probabilitas steady state adalah kondisi stabil di mana distribusi probabilitas tidak lagi berubah signifikan.  Hasil kode:  Matriks Transisi P:  [[0.5 0.5]  [0.5 0.5]]  Kondisi Awal:  [1 0]  Probabilitas pada Sesi 2:  [0.5 0.5]  Probabilitas pada Sesi 3:  [0.5 0.5]  Probabilitas pada Sesi 4: [0.5 0.5]  Steady state tercapai pada sesi 4 | |
|  | **Link Video Tutorial** |
| <https://youtu.be/-cPzt0NSl4Q> | |
|  | **Referensi:** |
| Dicoding Intern. (2023, Mei 31). *Python: Pengertian, contoh penggunaan, dan manfaat mempelajarinya*. Dicoding. Diakses pada 25 Oktover 2024 <https://www.dicoding.com/blog/python-pengertian-contoh-penggunaan-dan-manfaat-mempelajarinya/>  Walker, S. M. II. (n.d.). *What is a Markov chain?* Klu.ai. Diakses pada November 19, 2024, dari <https://klu.ai/glossary/markov-chain> | |