**PRAKTIKUM**

**ALGORITMA DAN STRUKTUR DATA**

**MODUL 10**

****

**Nama : NICKY JULYATRIKA SARI**

**NIM : L200200101**

**E**

**PROGRAM STUDI**

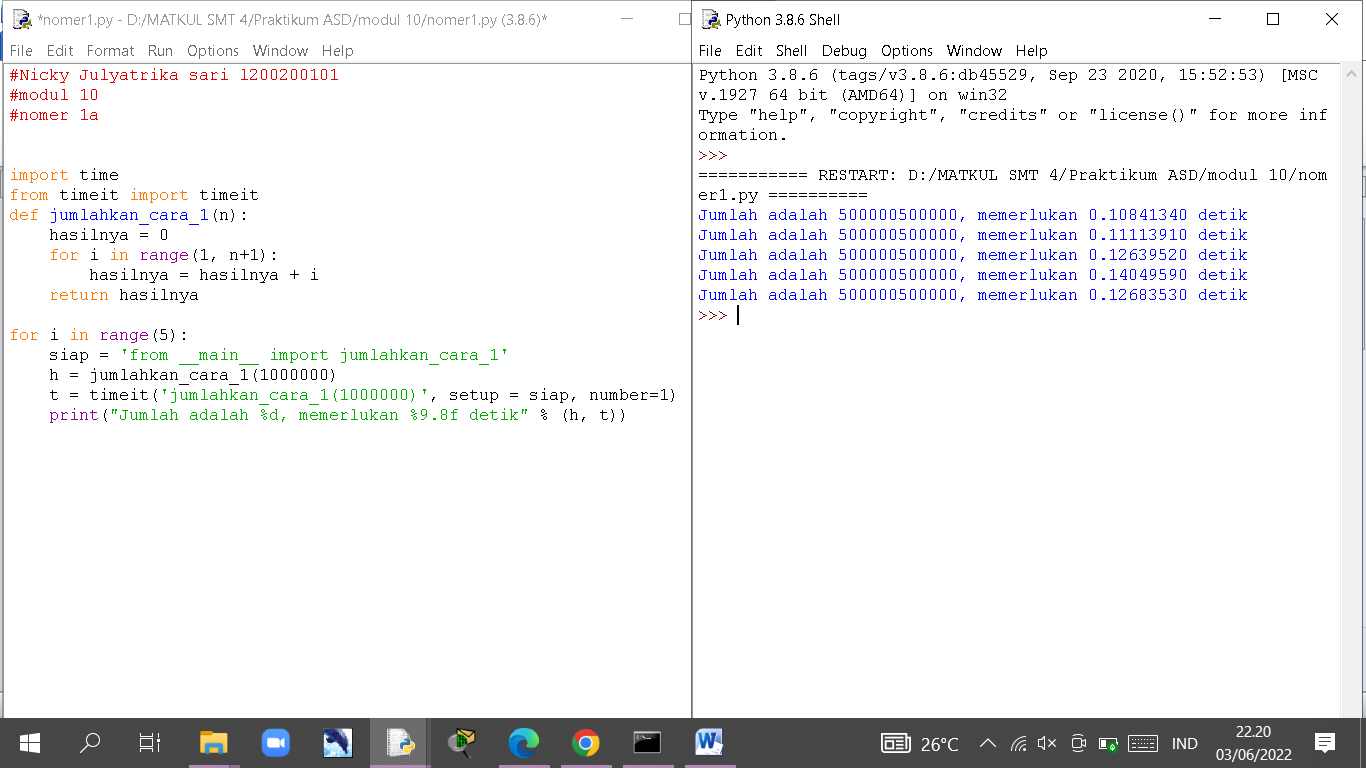
**INFORMATIKA**

**FAKULTAS KOMUNIKASI DAN INFORMATIKA**

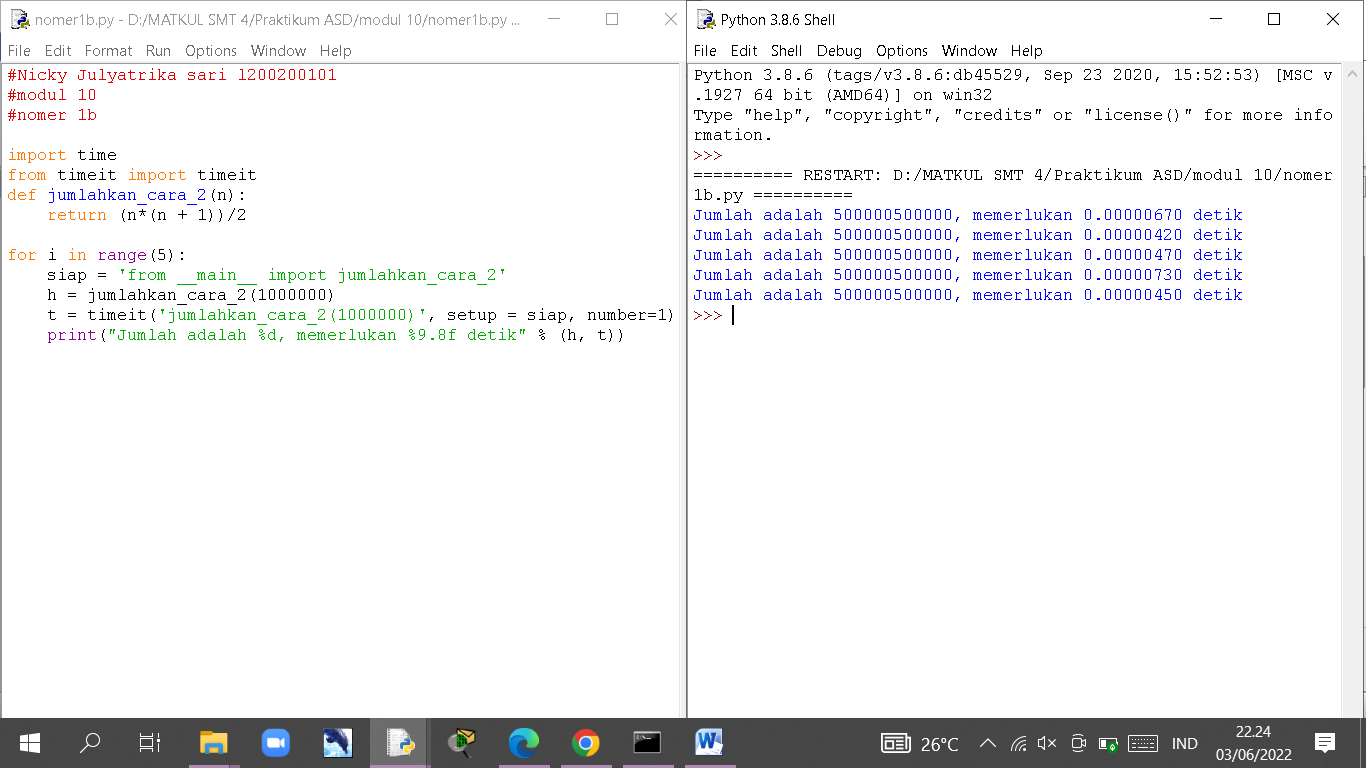
**UNIVERSITAS MUHAMMADIYAH SURAKARTA**

**TAHUN 2021/2022**

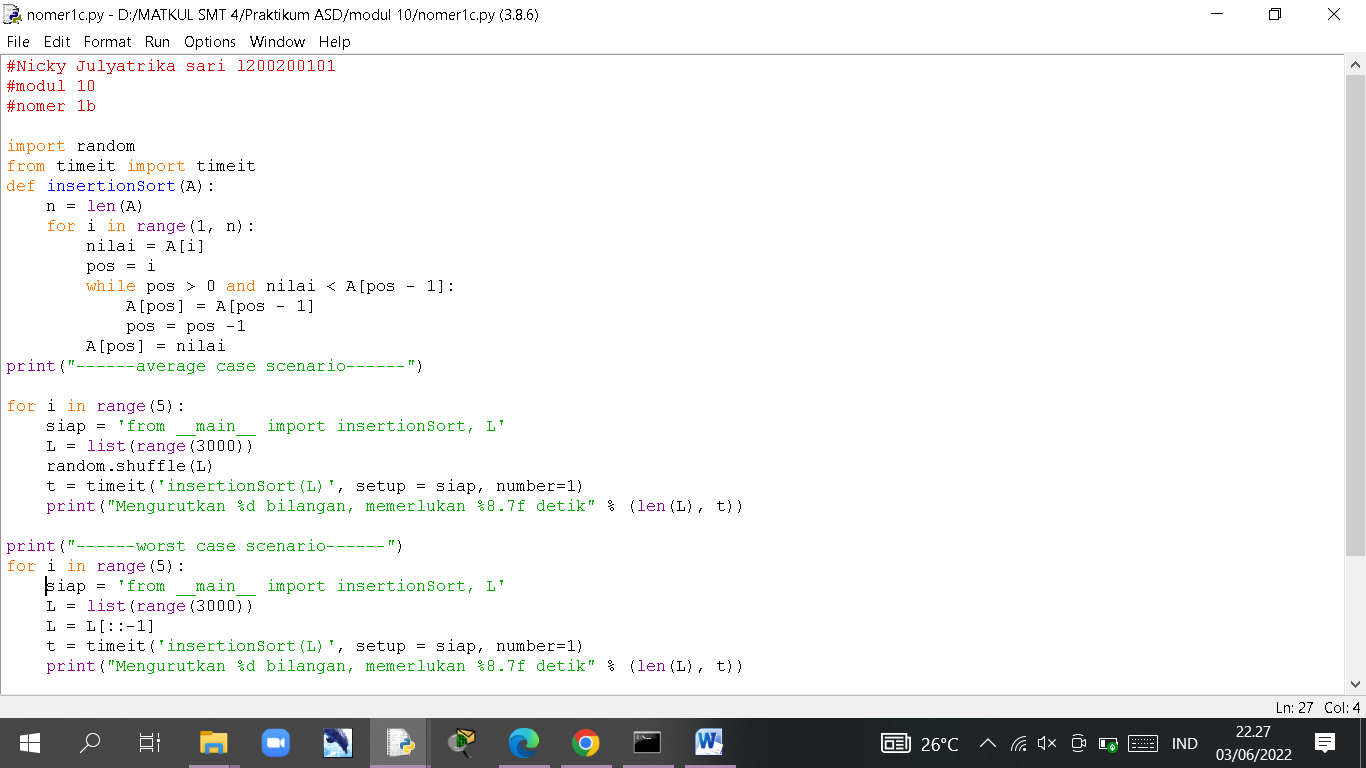
1. Kerjakan ulang contoh dan latihan di modul ini menggunakan modul timeit, yakni
   1. jumlahkan\_cara\_1

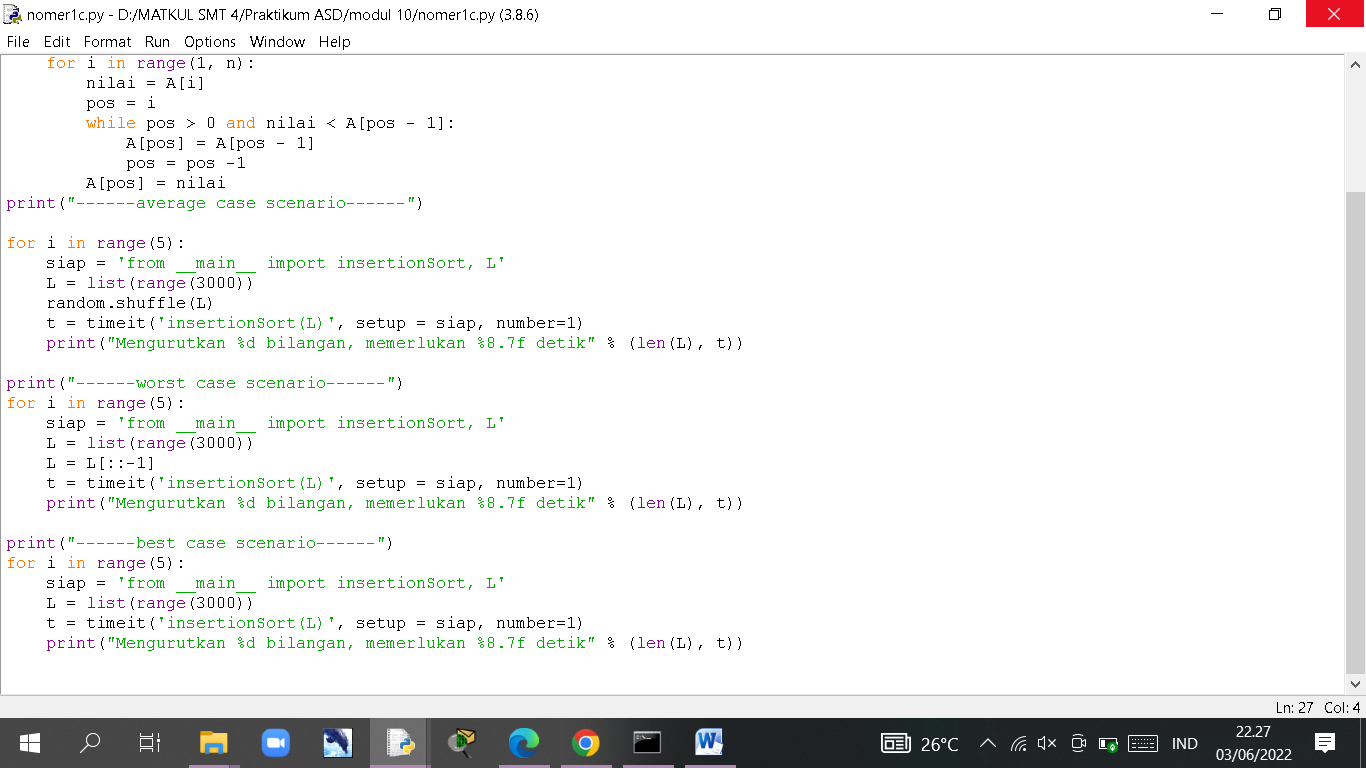


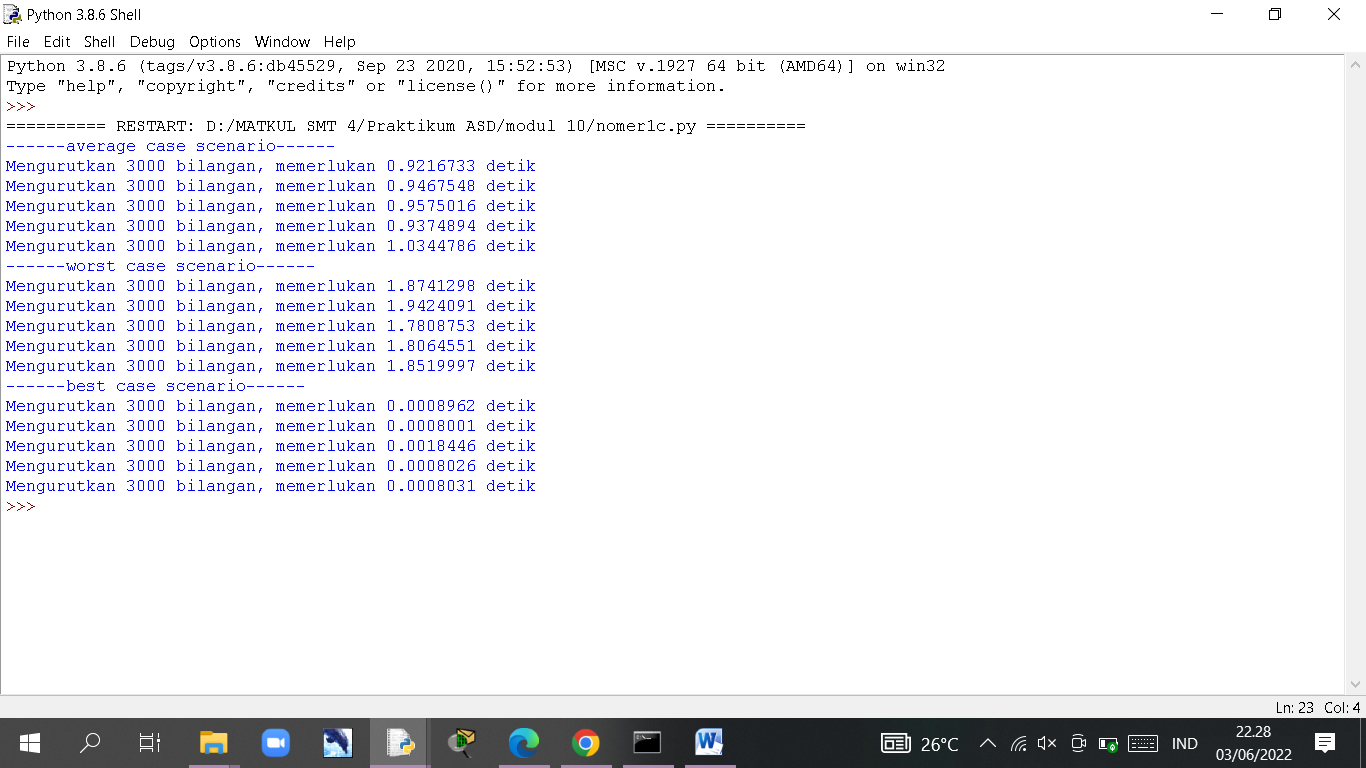
* 1. jumlahkan\_cara\_2



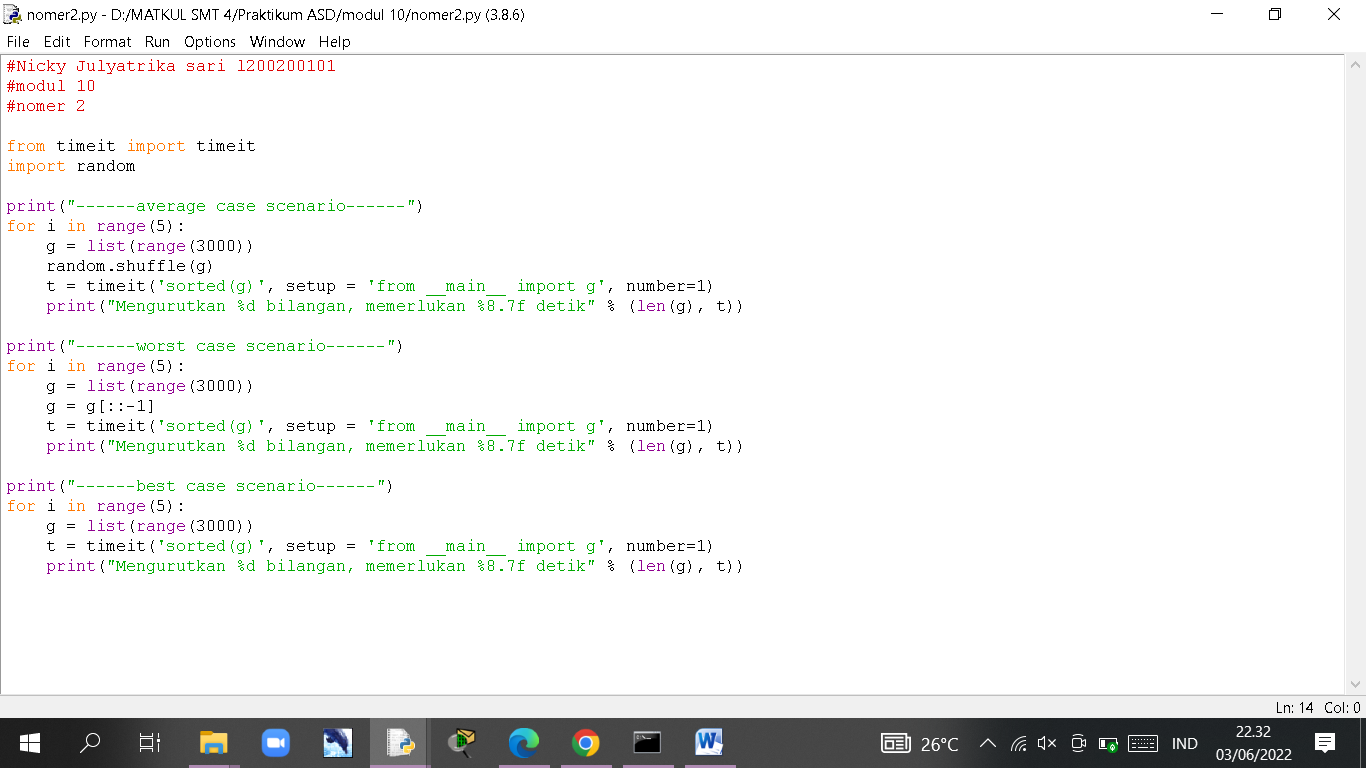
* 1. insertionSort

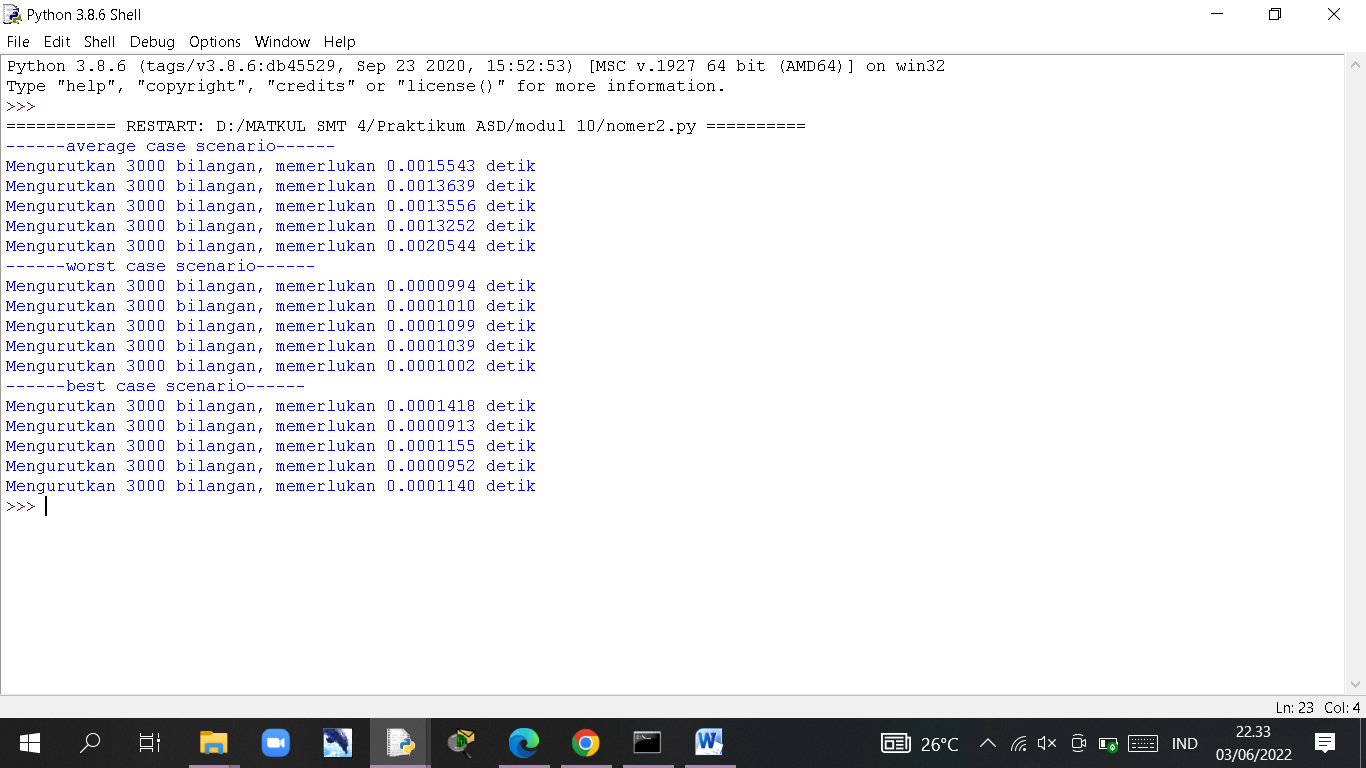




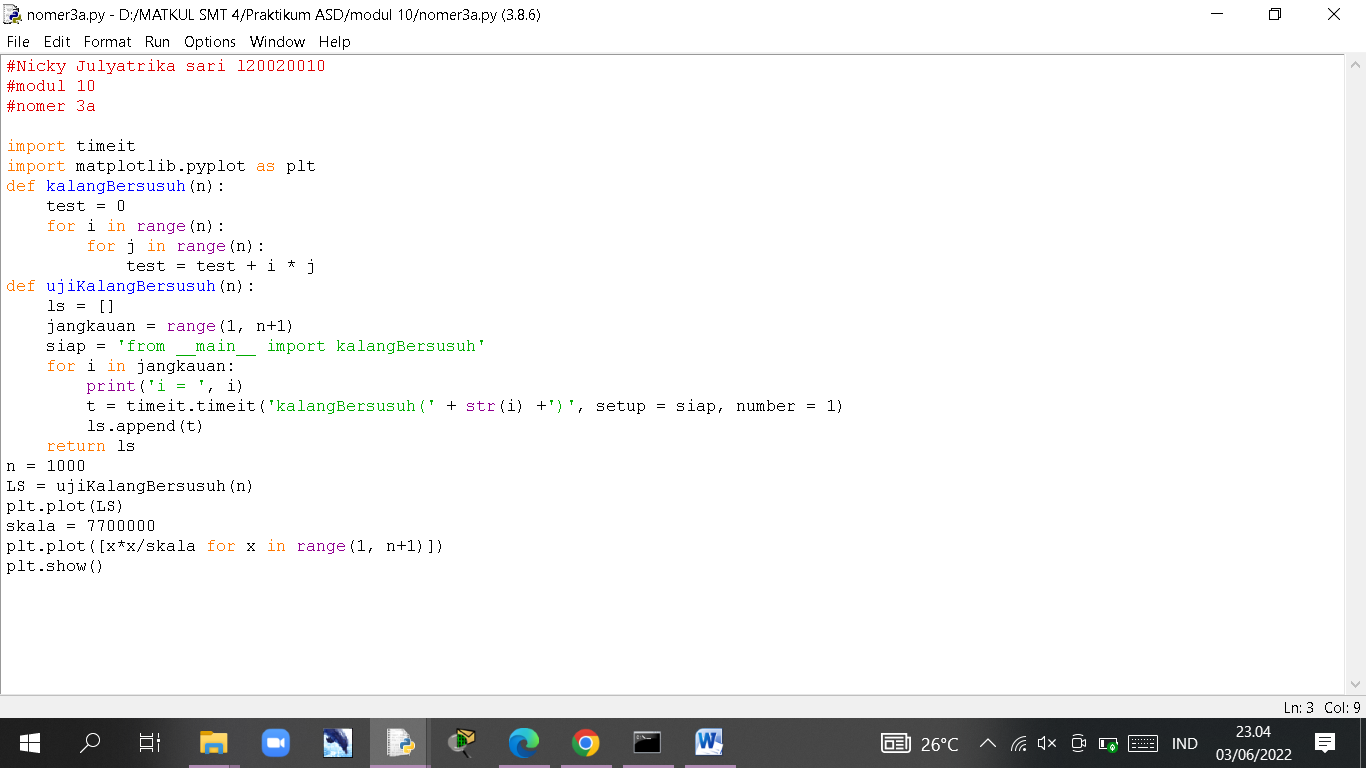


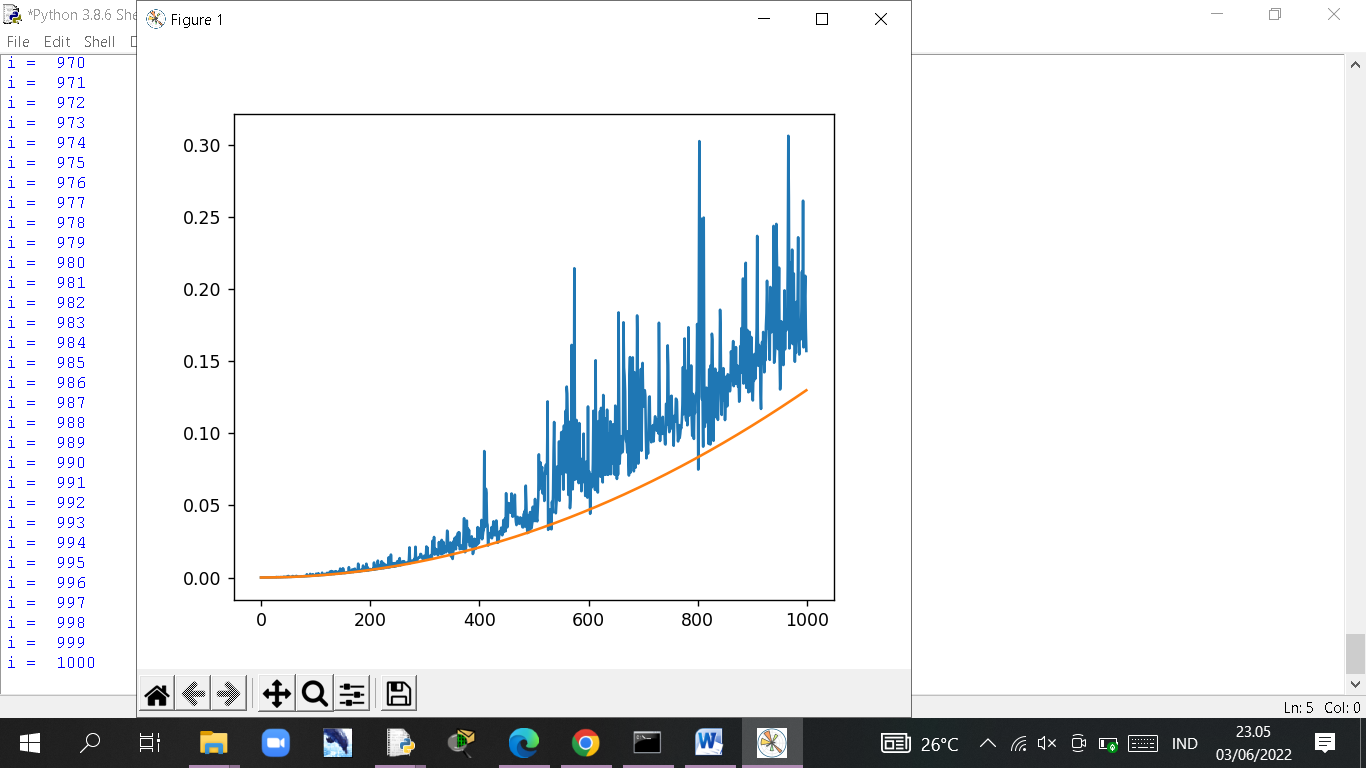
1. Python mempunyai perintah untuk mengurutkan suatu list yang memanfaatkan algoritma Timsort. Jika g adalah suatu list berisi bilangan, maka g.sort() kan mengurutkannya. Perintah yang lain, sorted() mengurutkan list dan mengembalikan sebuah list baru yang sudah urut. Selidikilah fungsi sorted ini menggunakan timeit:
   1. Apakah yang merupakan best case dan average case bagi sorted()?
   2. Confirm bahwa data input urutan terbalik bukan kasus terburuk bagi sorted(). Bahkan dia lebih cepat dalam mengurutkannya daripada data input random



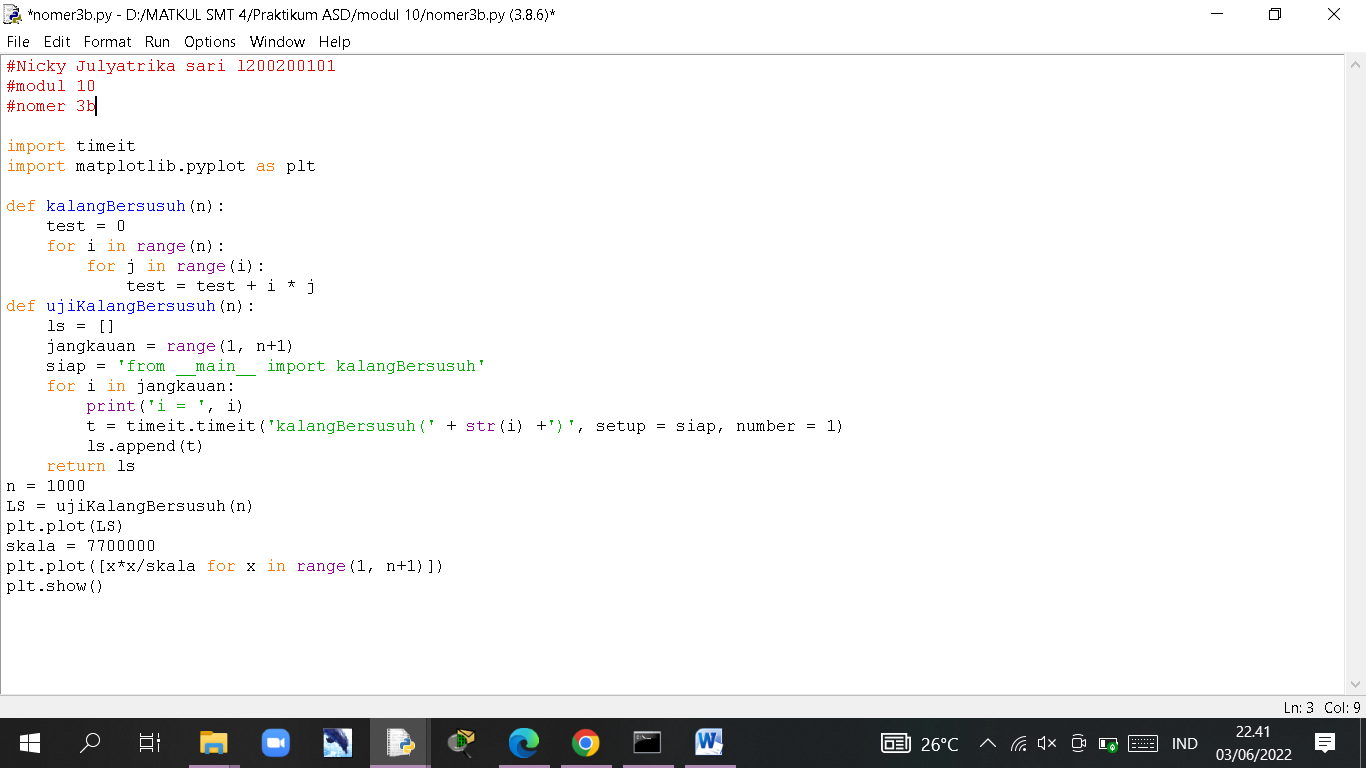


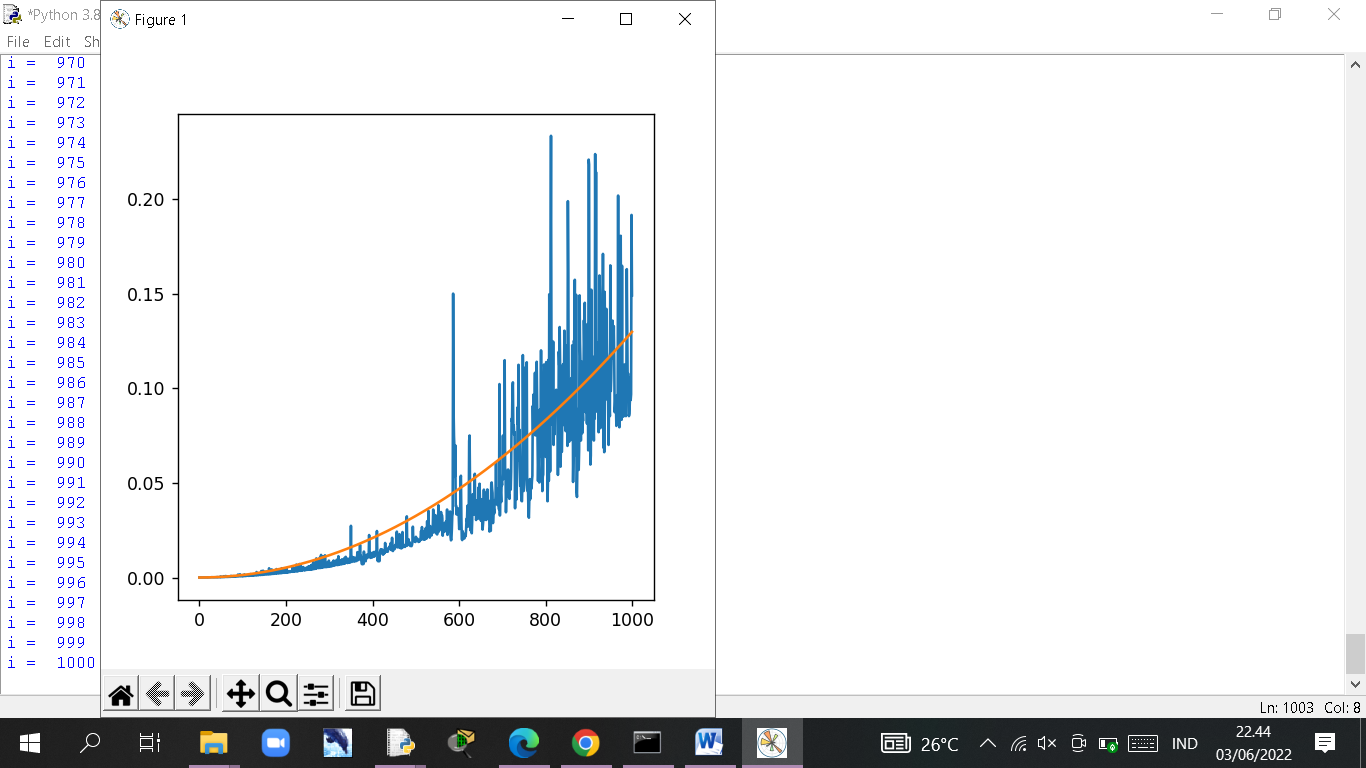
1. Untuk tiap kode berikut, tentukan running time-nya, O(1), O(log n), O(n), O(n log n), O(n2), atau O(n3), atau yang lain. Untuk memulai analisis, ambil suatu nilai n tertentu, lalu ikuti apa yang terjadi di kode itu.
   1. loop di dalam loop, keduanya sebanyak n:



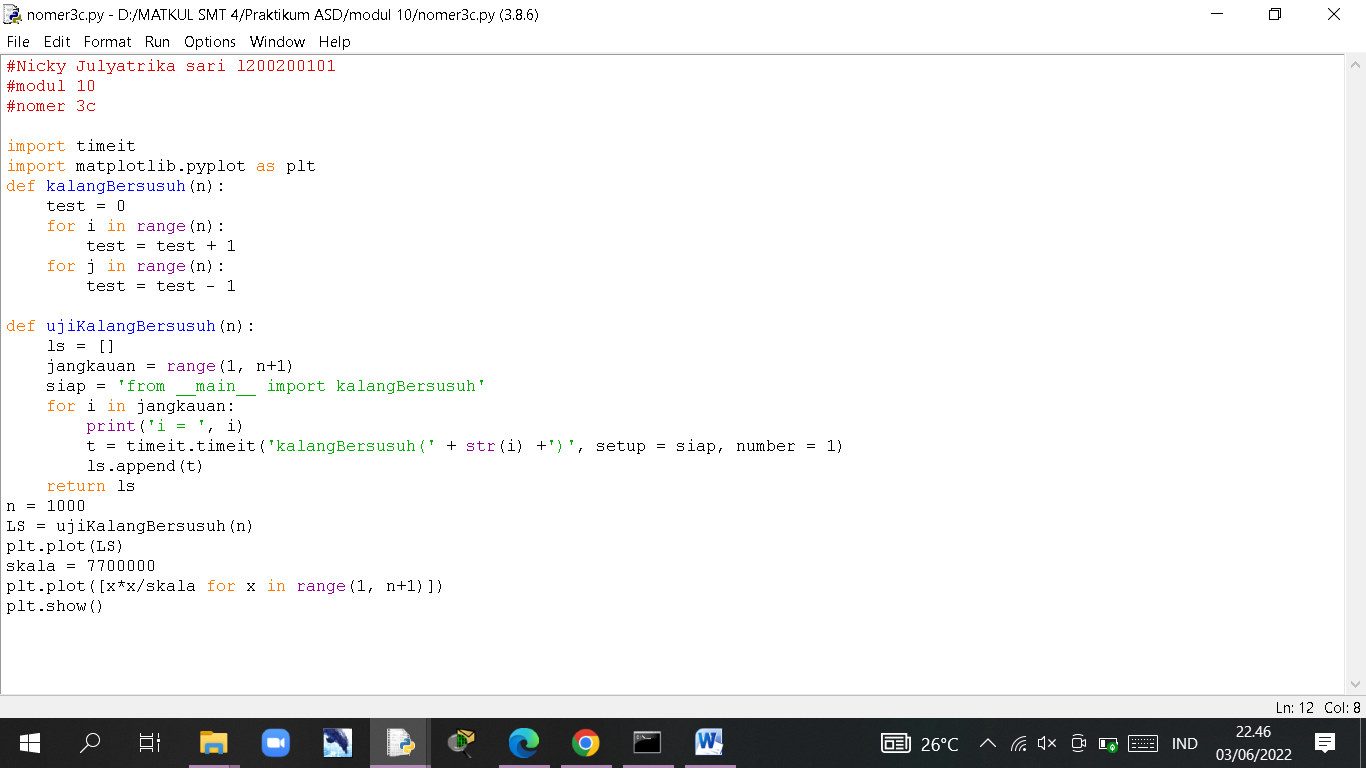


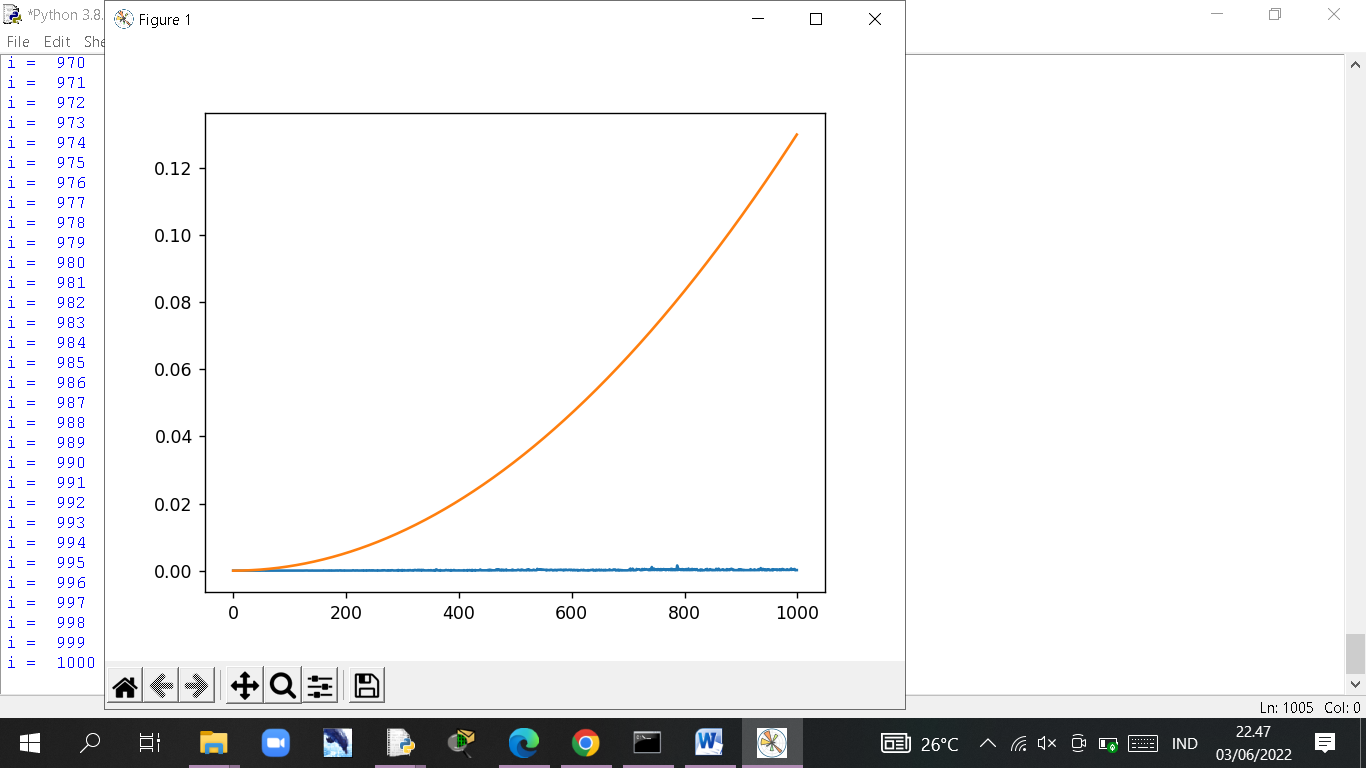
* 1. loop di dalam loop, yang dalam bergantung nilai i loop luar:



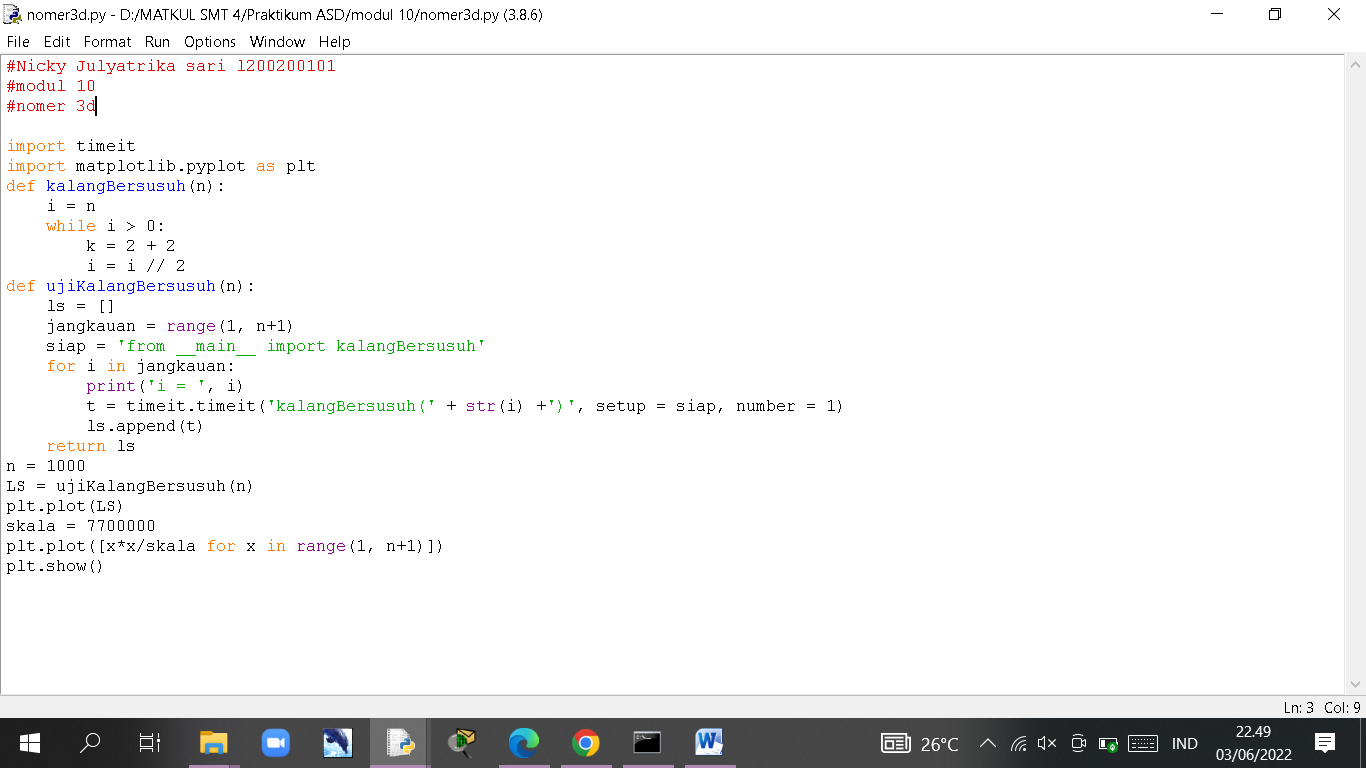


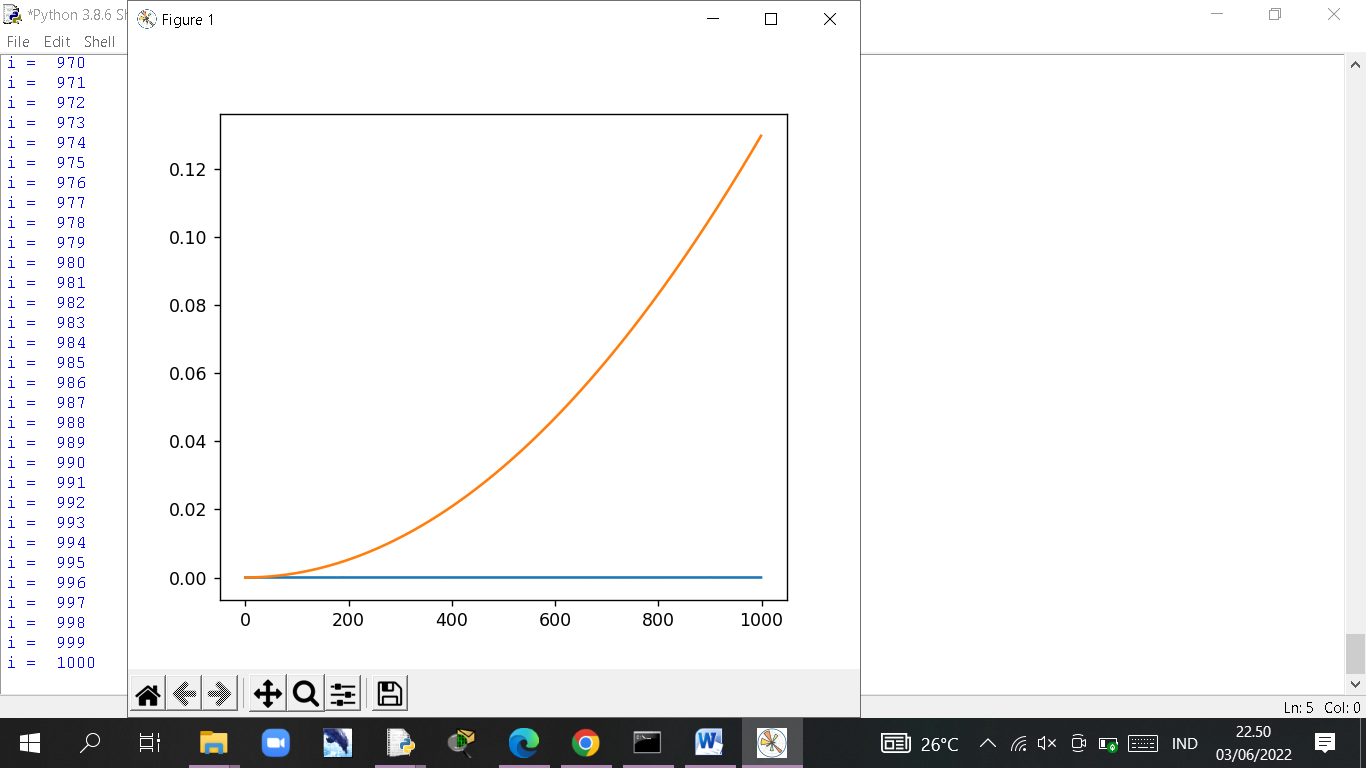
* 1. dua loop terpisah:



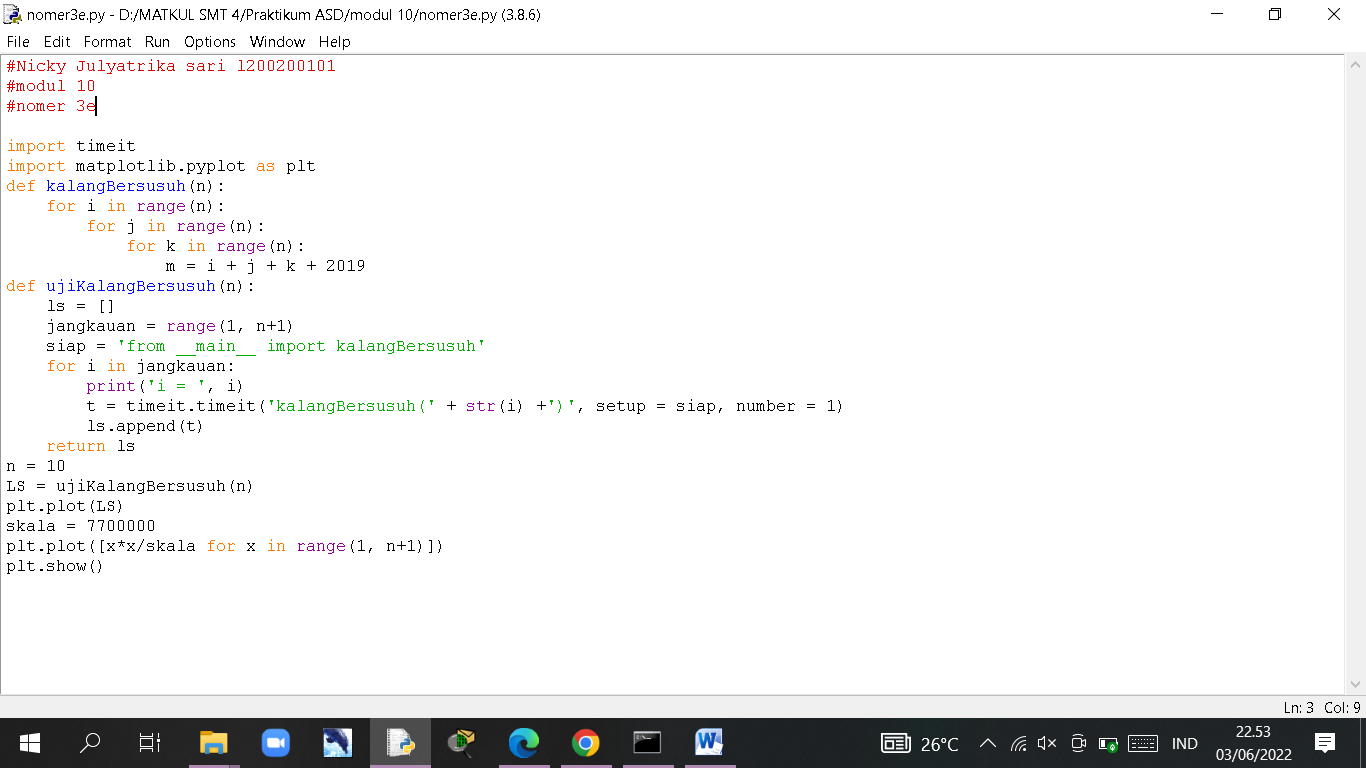


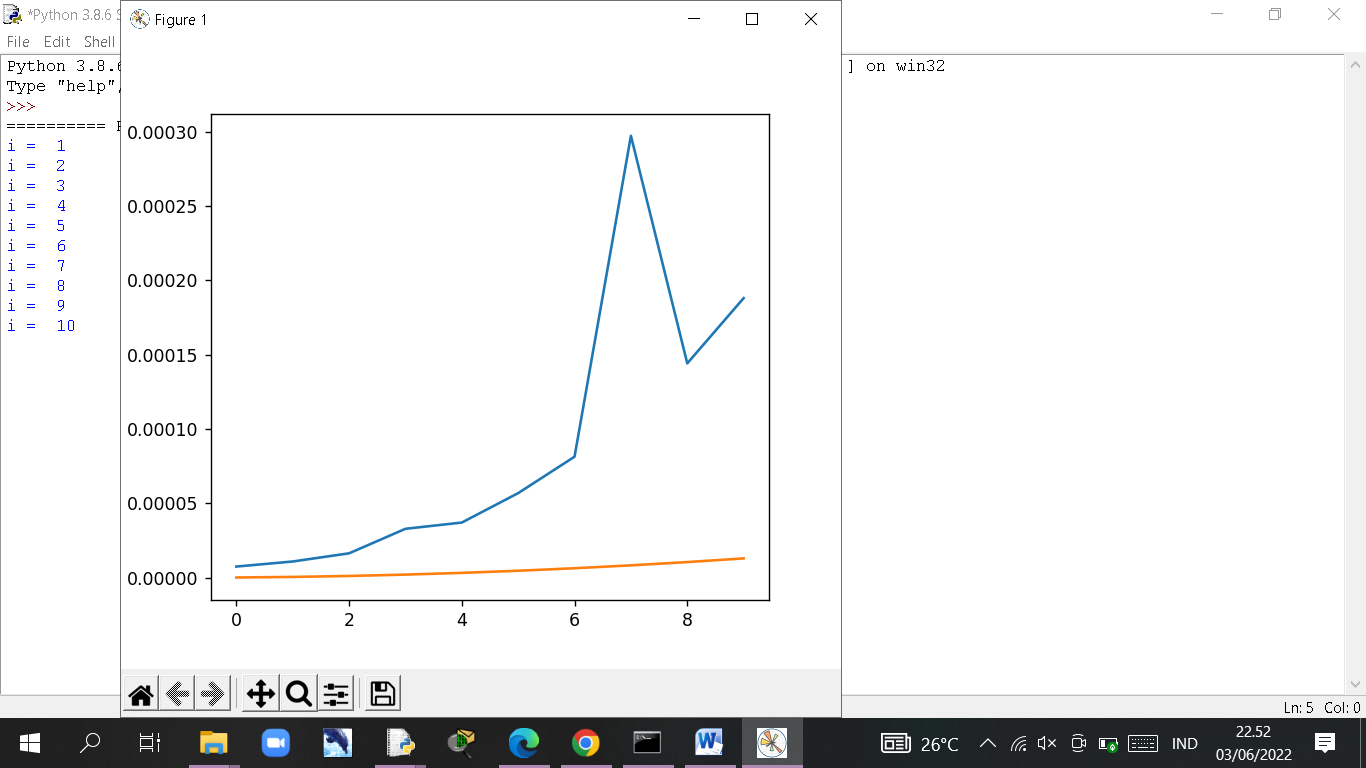
* 1. while loop yang dipangkas separuh tiap putaran



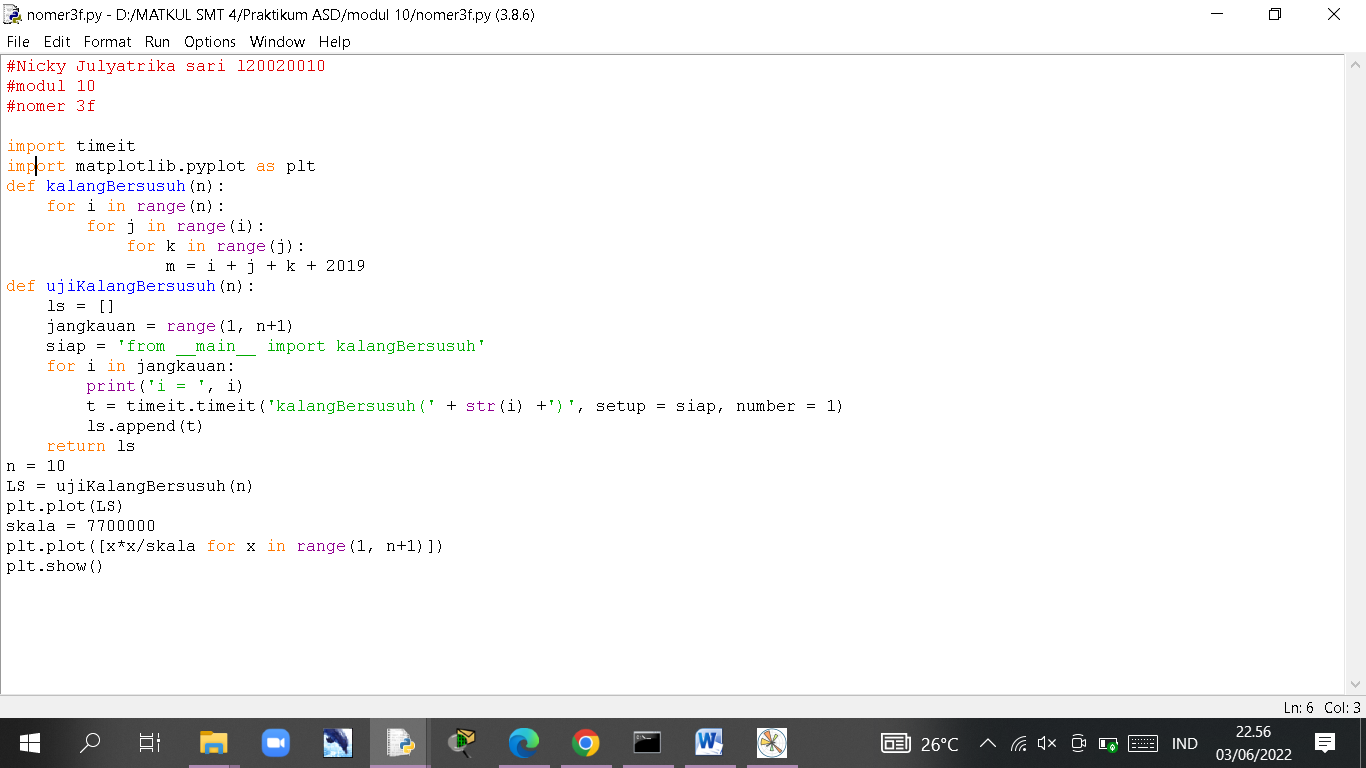


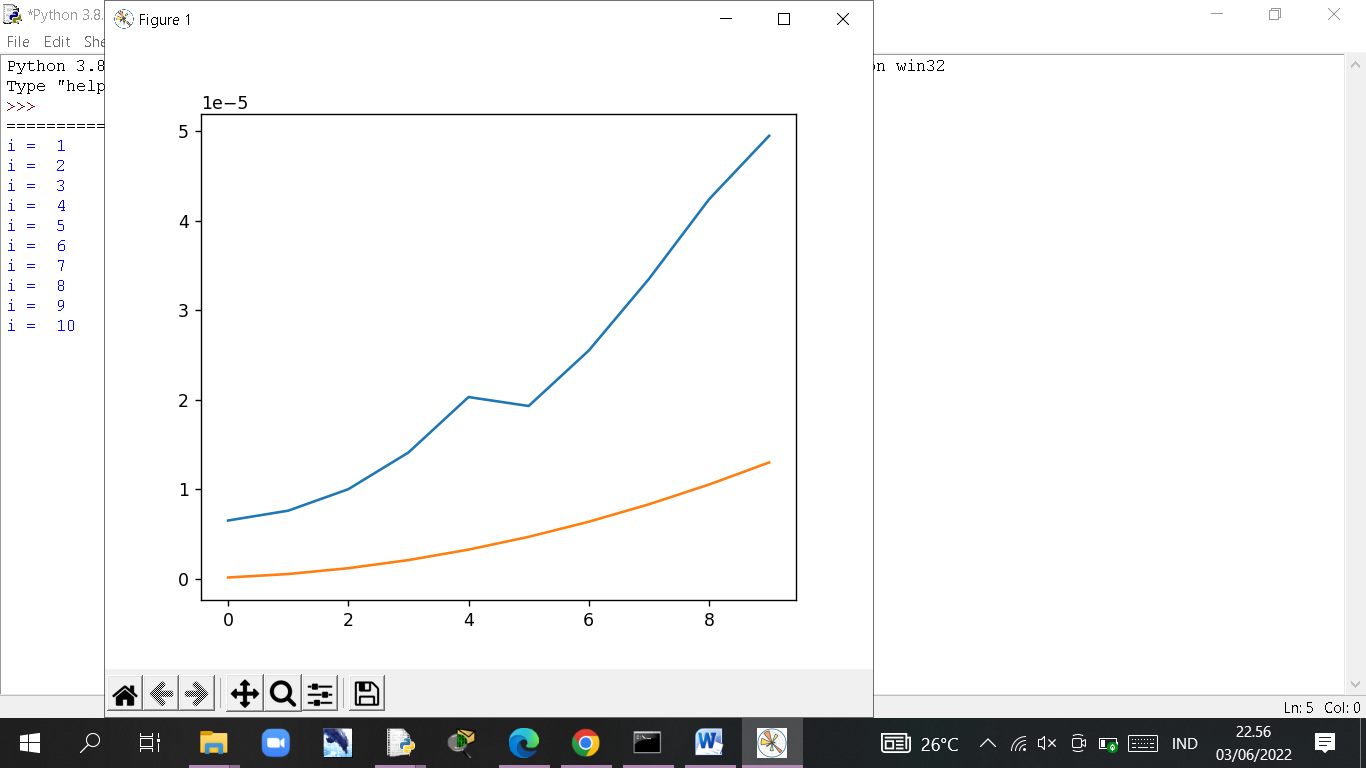
* 1. loop in a loop in a loop, ketiganya sebanyak n



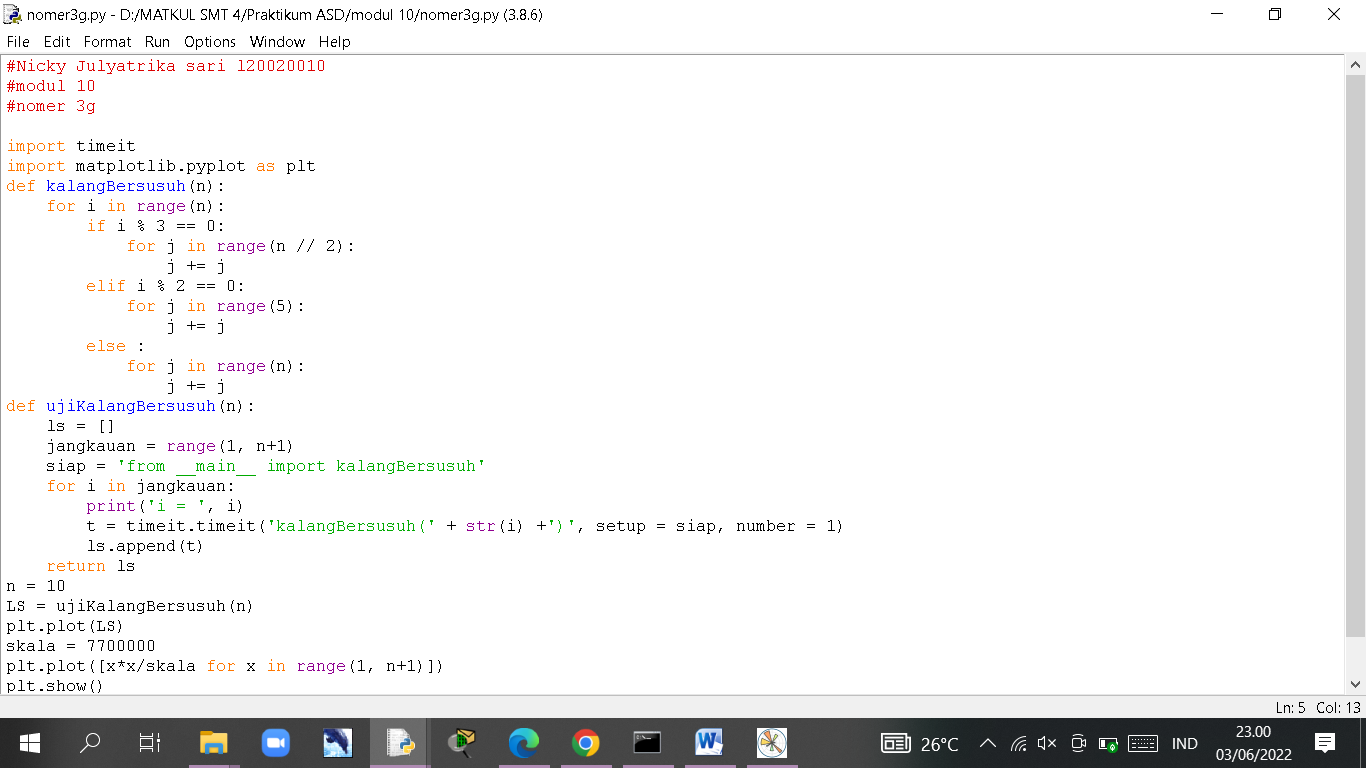


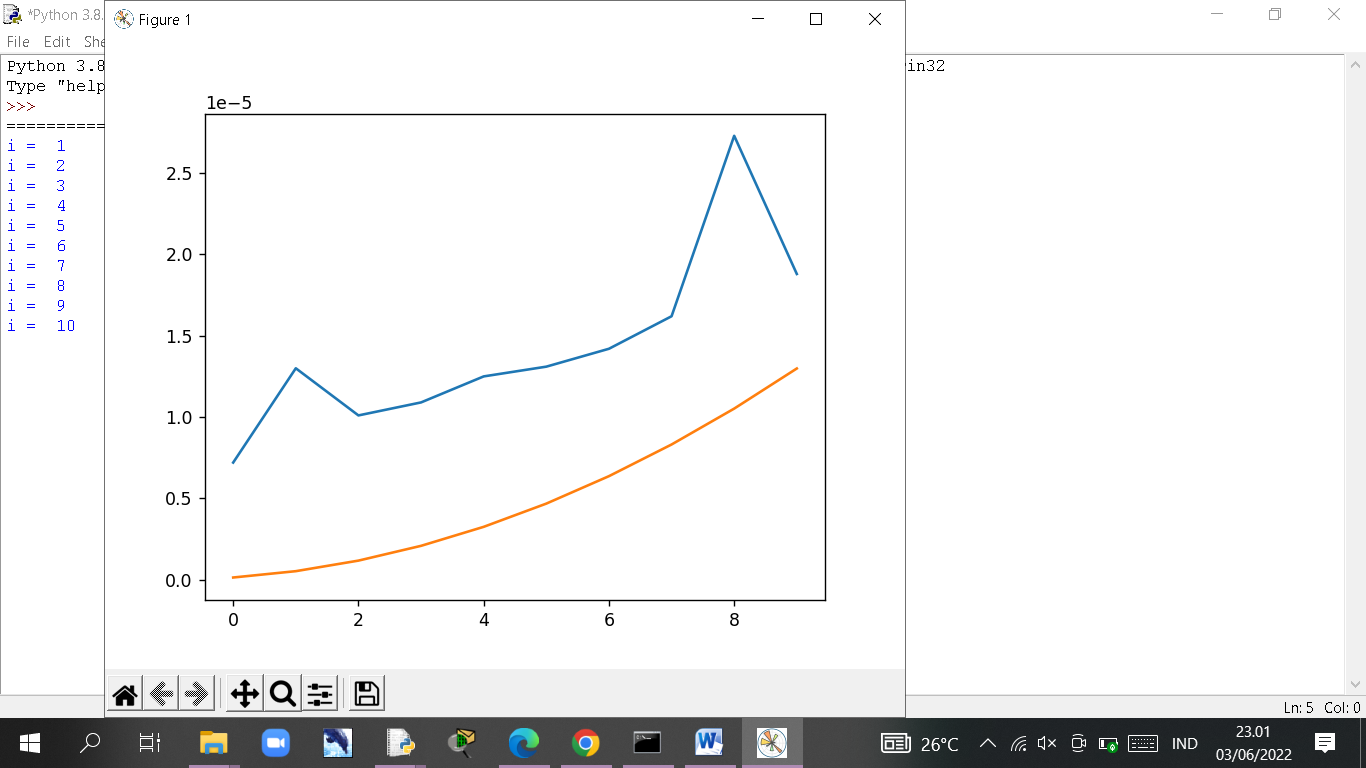
* 1. loop in a loop in a loop, dengan loop dalam sebanyak nilai loop luar terdekat





* 1. fungsi ini:





1. Urutkan dari yang pertumbuhan kompleksitasnya lambat ke yang cepat:

< < < < << <

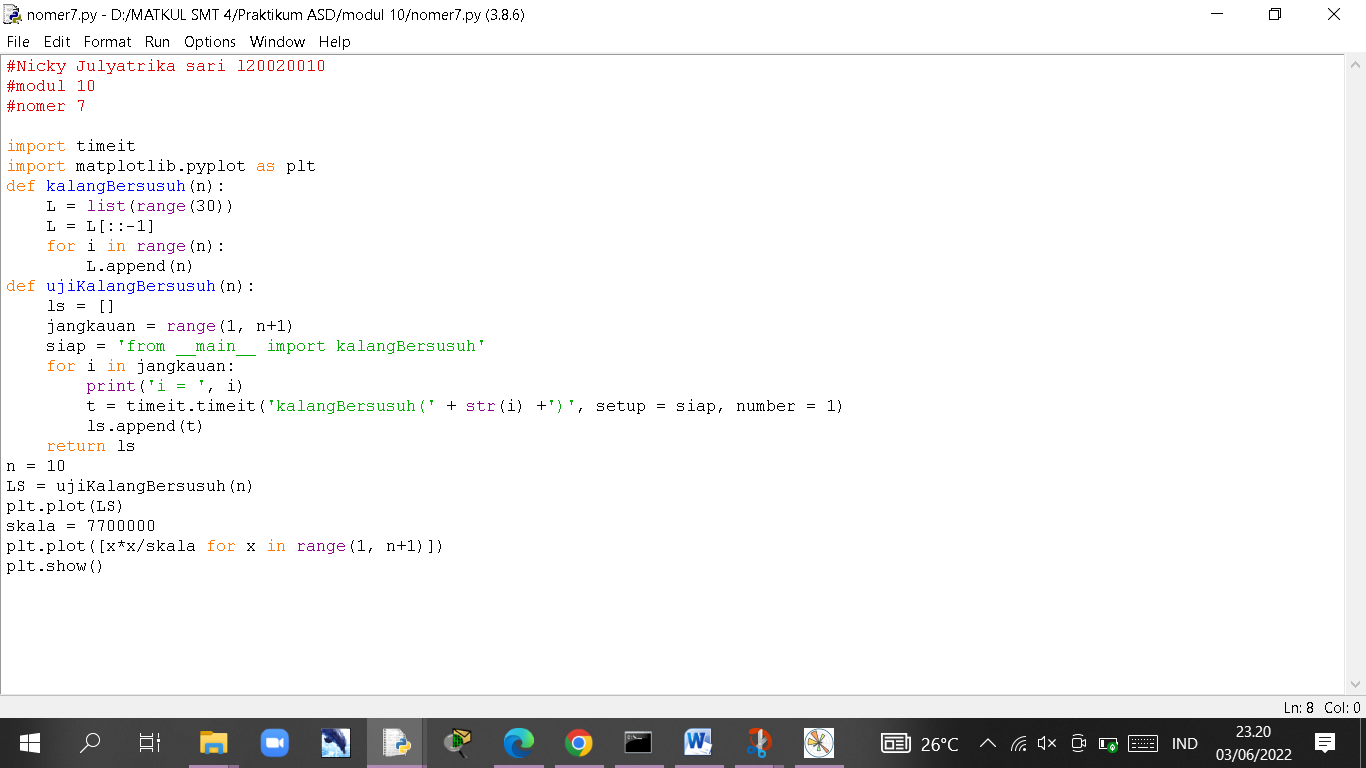
1. Tentukan O(·) dari fungsi-fungsi berikut, yang mewakili banyaknya langkah yang diperlukan untuk beberapa algoritm
   1. = + 32 + 8 = O()
   2. = 87 + 3 = O(
   3. = 4 + 5log + 102 = O( log)
   4. = log + 3 + 88 = O()
   5. = 3(2) + + 647 = O()
   6. = + log = O(
   7. = 8 + log + 800 = O()
   8. = 100 + = O()
2. (Literature Review) Carilah di Internet, kompleksitas metode-metode pada object list di Python. Hint:
   1. Google python list methods complexity . Lihat juga bagian ”Images”-nya
   2. Kunjungi <https://wiki.python.org/moin/TimeComplexit>

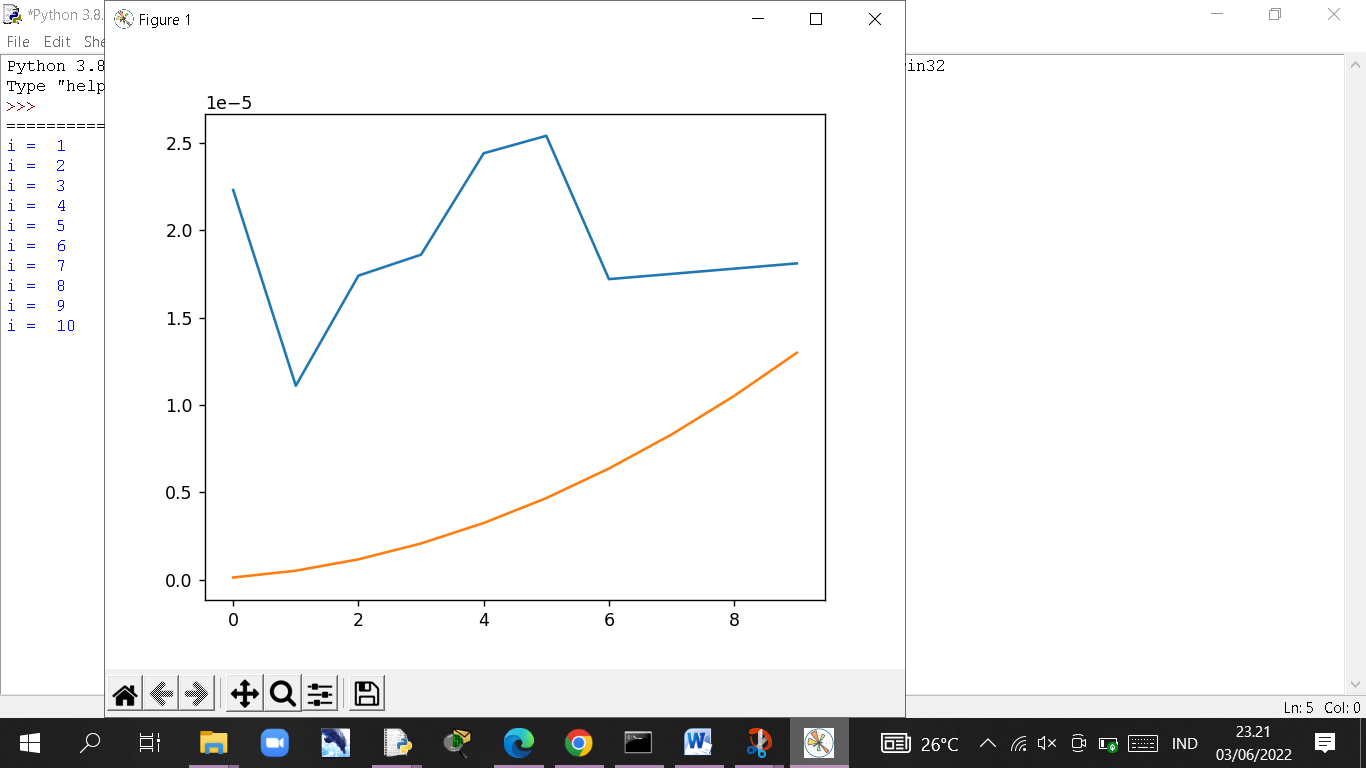
Kasus Rata-rata mengasumsikan parameter yang dihasilkan seragam secara acak.

Secara internal, list direpresentasikan sebagai array; biaya terbesar berasal dari pertumbuhan di luar ukuran alokasi saat ini (karena semuanya harus bergerak), atau dari memasukkan atau menghapus suatu tempat di dekat awal (karena semuanya setelah itu harus bergerak). Jika perlu kita menambahkan/menghapus di kedua ujungnya, pertimbangkan untuk menggunakan collections.deque sebagai gantinya.

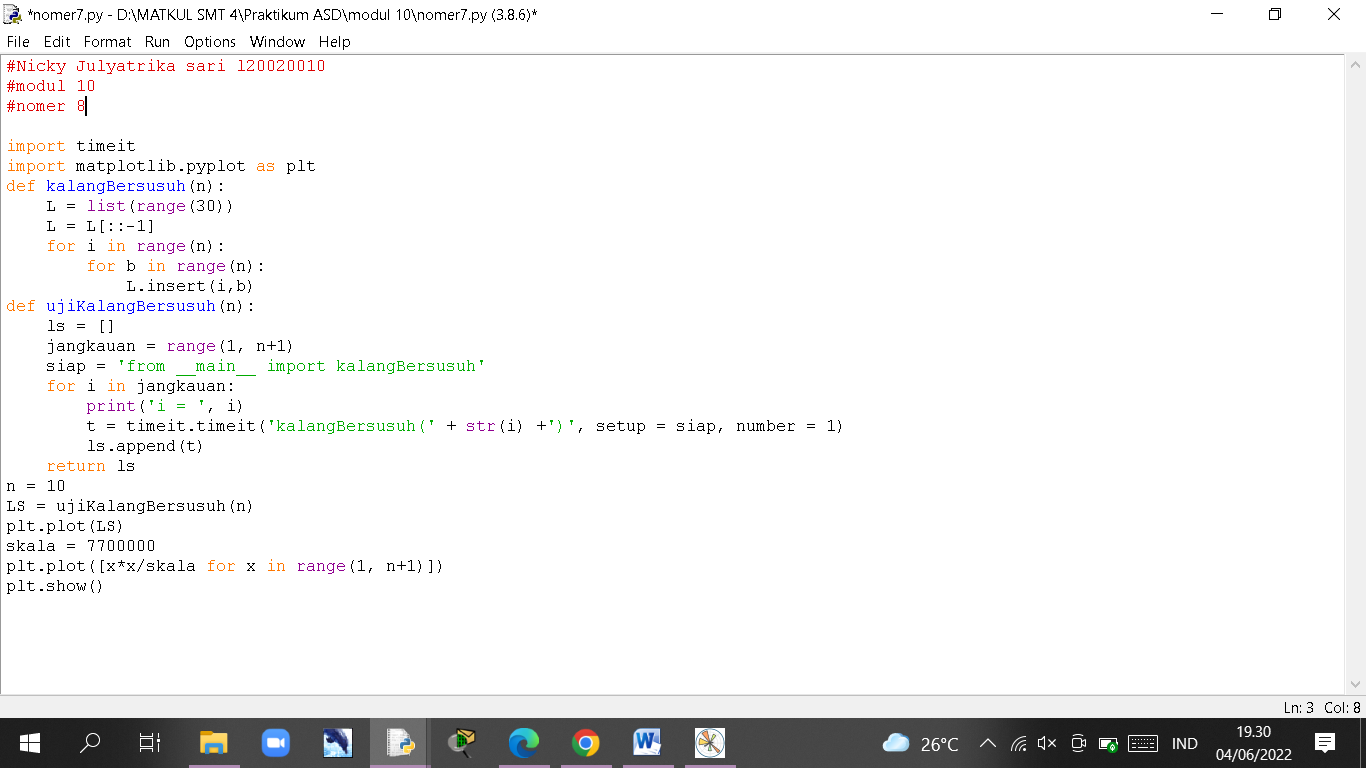
|  |  |  |
| --- | --- | --- |
| operation | Average Case | Amortized Worst Case |
| Copy | O(n) | O(n) |
| Append[1] | O(1) | O(1) |
| Pop last | O(1) | O(1) |
| Pop intermediate | O(n) | O(n) |
| Insert | O(n) | O(n) |
| Get item | O(1) | O(1) |
| Set item | O(1) | O(1) |
| Delete item | O(n) | O(n) |
| Iteration | O(n) | O(n) |
| Get Slice | O(k) | O(k) |
| Del Slice | O(n) | O(n) |
| Set Slice | O(k+n) | O(k+n) |
| Extend[1] | O(k) | O(k) |
| Sort | O(n log n) | O(n log n) |
| multiply | O(nk) | O(nk) |
| x in s | O(n) |  |
| Min(s), max(s) | O(n) |  |
| Get Length | O(1) | O(1) |

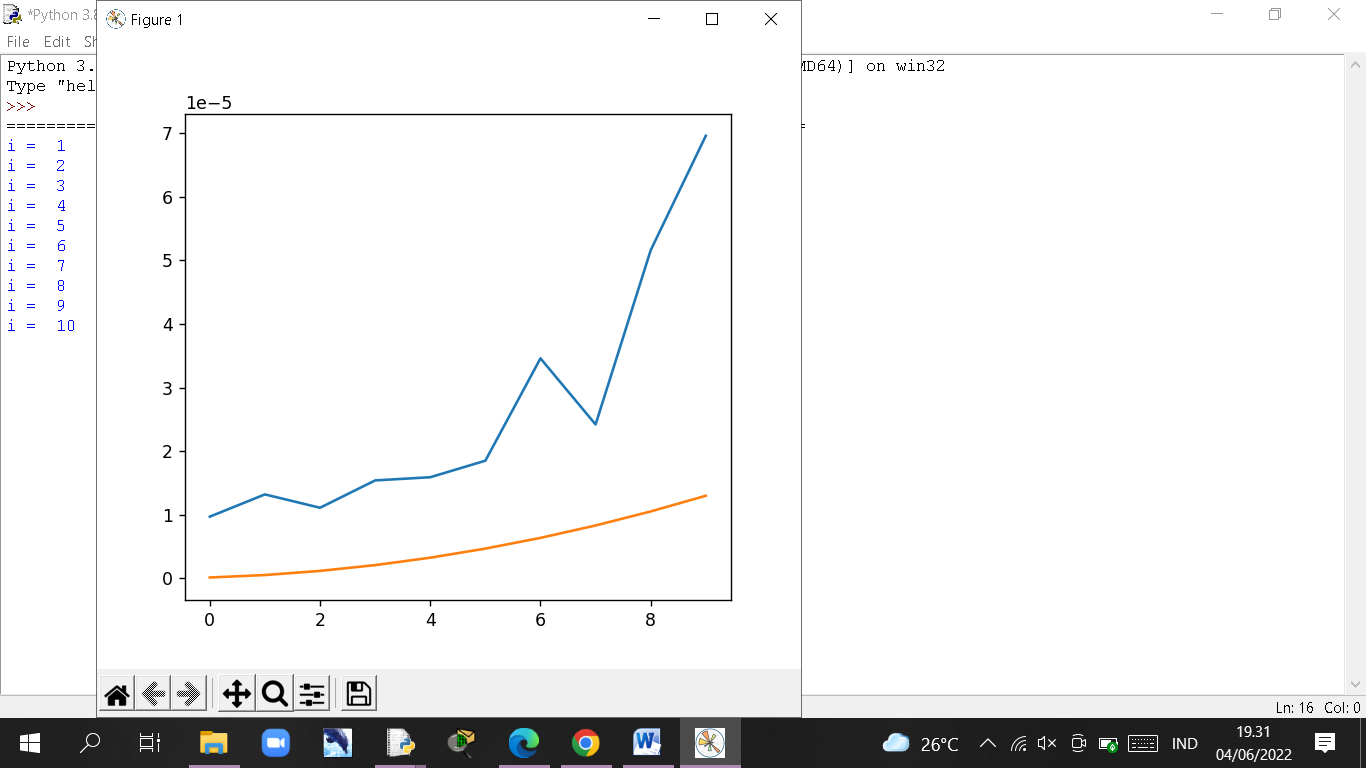
1. Buatlah suatu ujicoba untuk mengkonfirmasi bahwa metode append() adalah O(1). Gunakan timeit dan matplotlib, seperti sebelumnya.



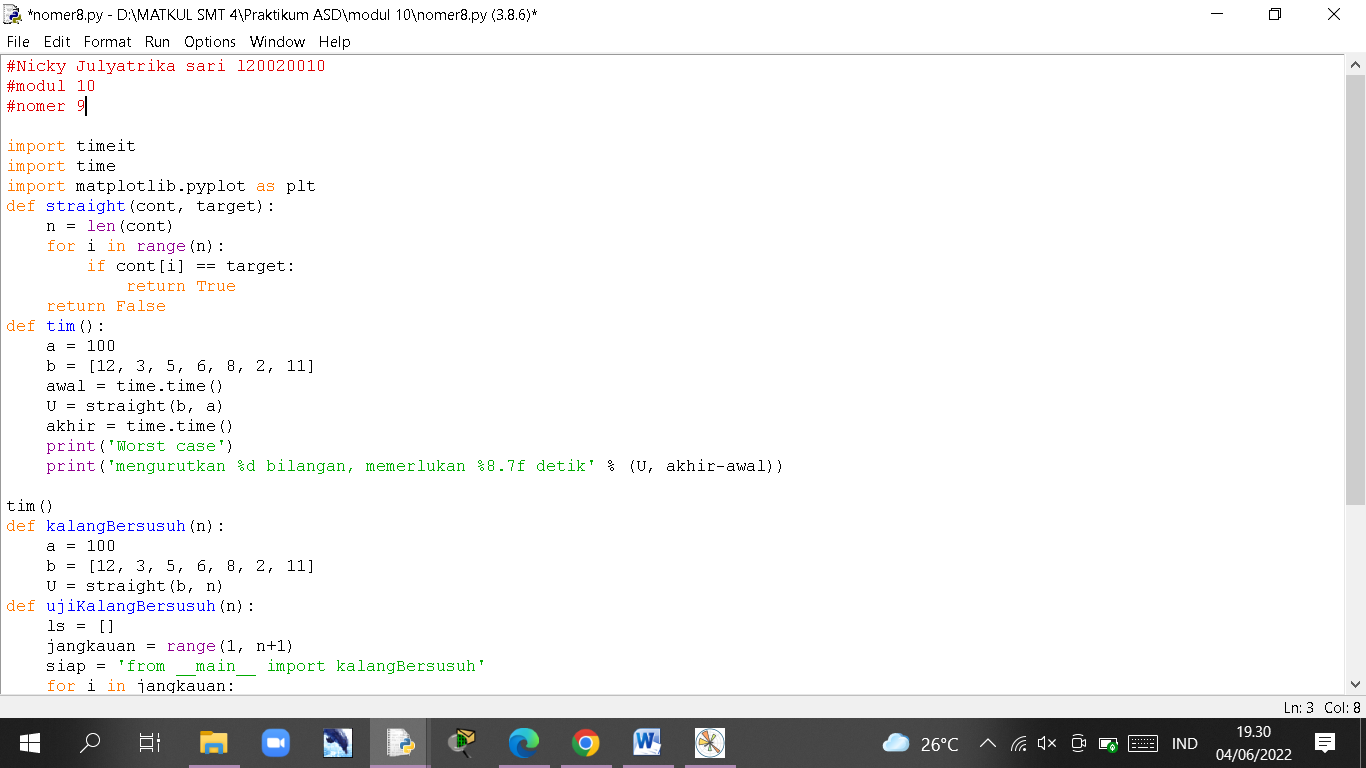


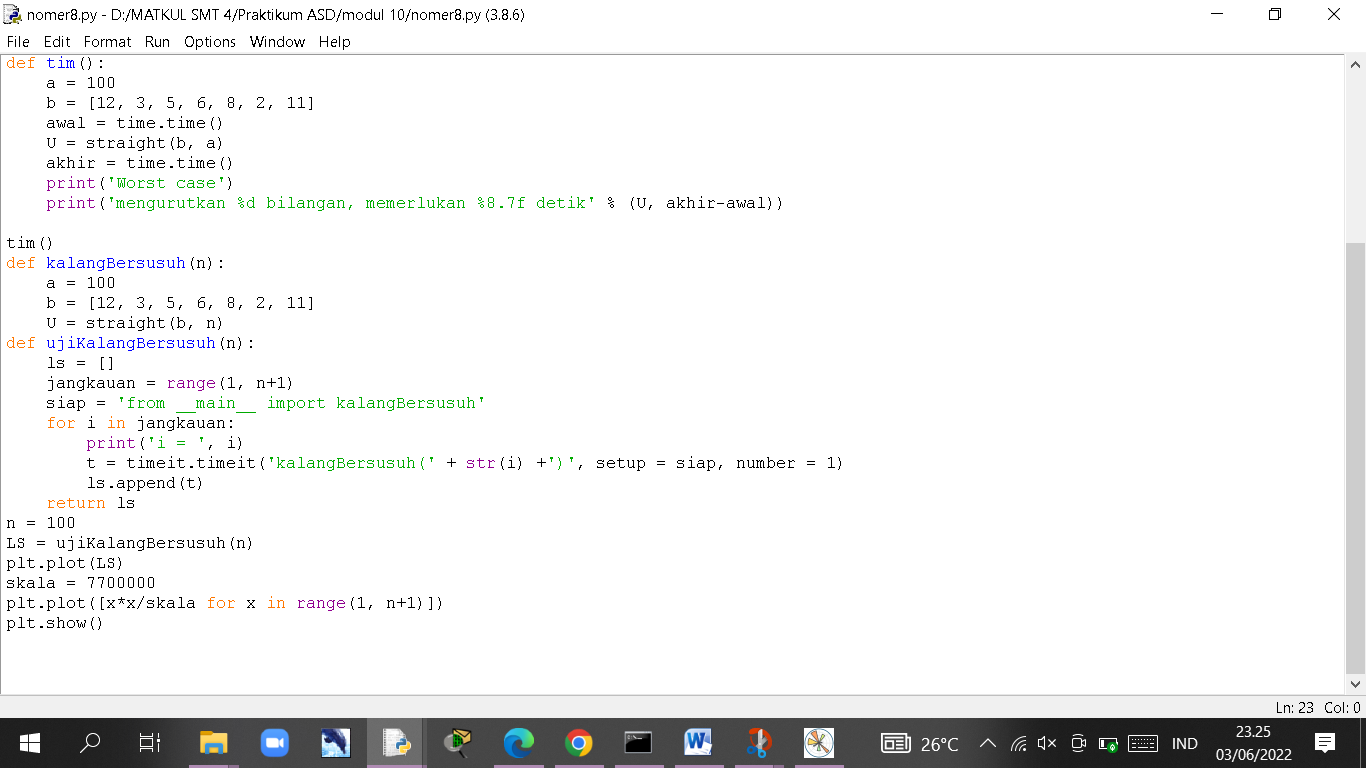
1. Buatlah suatu ujicoba untuk mengkonfirmasi bahwa metode insert() adalah O(n). Gunakan timeit dan matplotlib, seperti sebelumnya.

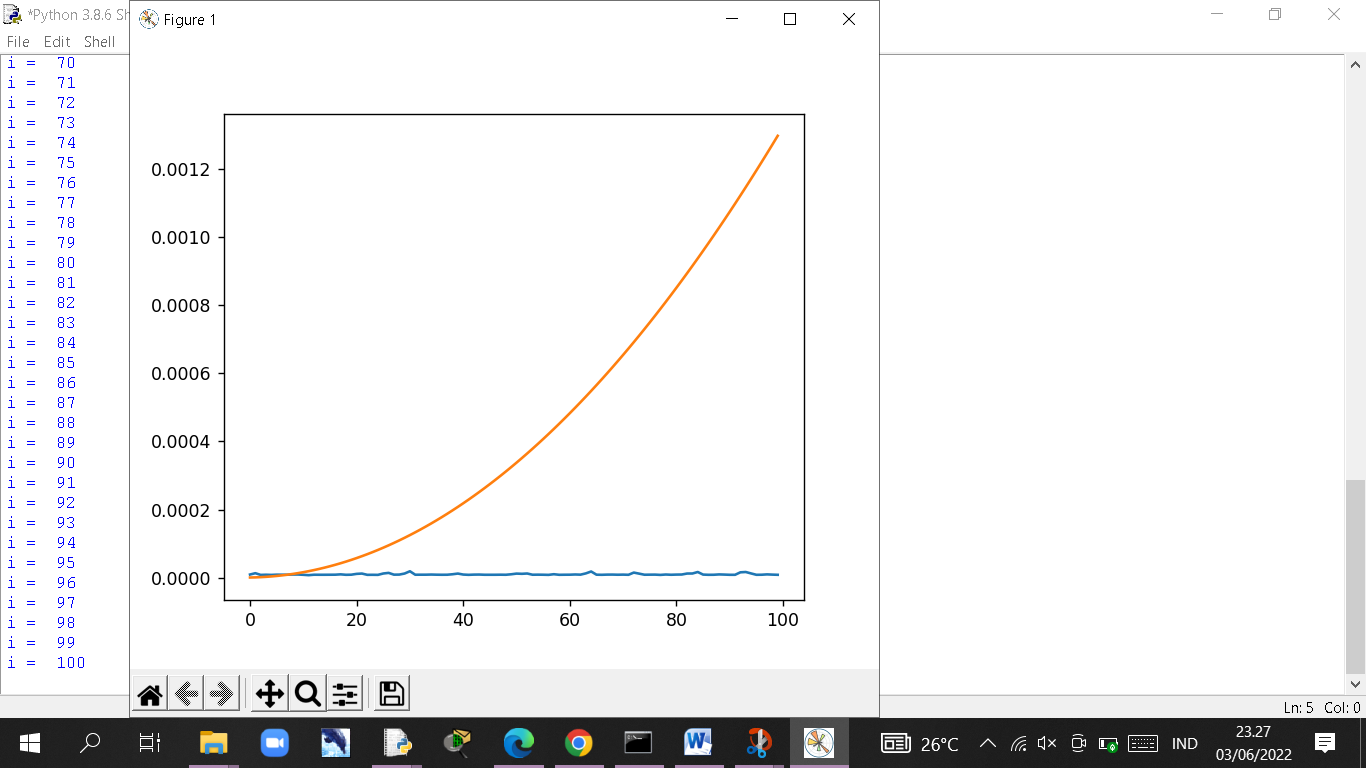




1. Buatlah suatu ujicoba untuk mengkonfirmasi bahwa untuk memeriksa apakah-suatu-nilaiberada-di-suatu-list mempunyai kompleksitas O(n). Gunakan timeit dan matplotlib, seperti sebelumnya.







1. (Literature Review) Carilah di Internet, kompleksitas metode-metode pada object dict di Python

Waktu Kasus Rata-rata yang terdaftar untuk objek dict mengasumsikan bahwa fungsi hash untuk objek cukup kuat untuk membuat tabrakan menjadi tidak biasa. Kasus Rata-rata mengasumsikan kunci yang digunakan dalam parameter dipilih secara seragam secara acak dari kumpulan semua kunci.

Perhatikan bahwa ada jalur cepat untuk dict yang (dalam praktiknya) hanya menangani kunci str; ini tidak mempengaruhi kompleksitas algoritmik, tetapi secara signifikan dapat mempengaruhi faktor konstan: seberapa cepat program biasa selesai.

|  |  |  |
| --- | --- | --- |
| Operation | Average Case | Amortized Worst Case |
| k in d | O(1) | O(n) |
| Copy[3] | O(n) | O(n) |
| Get Item | O(1) | O(n) |
| Set Item[1] | O(1) | O(n) |
| Delete item | O(1) | O(n) |
| Iteration[3] | O(n) | O(n) |

1. (Literature Review) Selain notasi big-O O(·), ada pula notasi big-Theta Θ(·) dan notasi big-Omega Ω(·). Apakah beda di antara ketiganya?
   1. Big O dilambangkan dengan notasi O(...) merupakan keadaan terburuk (worst case). Kinerja seubuah algoritma biasanya diukur menggunakan patokan keadaan Big-O ini. Merupakan notasi asymptotic untuk batas fungsi dari atas dan bawah dengan Berperilaku mirip dengan ≤ operator untuk tingkat pertumbuhan.
   2. Big Theta dilmbangkan dengan notasi Θ(...) merupakan notasi asymptotic untuk batas atas dan bawah dengan keadaan terbaik (best case). Menyatakan persamaan pada pertumbuhan f (n) hingga faktor konstan (lebih lanjut tentang ini nanti). Berperilaku mirip dengan = operator untuk tingkat pertumbuhan.
   3. Big Omega dilambangkan dengan notasi Ω(...) merupakan notasi asymptotic untuk batas bawah dengan keadaan rata-rata(average case) yang berperilaku mirip dengan ≥operator untuk tingkat pertumbuhan.
2. (Literature Review) Apa yang dimaksud dengan amortized analysis dalam analisis algoritma?

Amortized analysis adalah metode untuk menganalisis kompleksitas algoritma yang diberikan, atau berapa banyak resource nya terutama waktu atau memori yang diperlukan untuk mengeksekusi. Dapat ditunjukkan dengan waktu rata-rata yang diperlukan untuk melakukan satu urutan operasi pada struktur data terhadap keseluruhan operasi yang dilakuka

Dalam Amortized Analysis,waktu yang diperlukan untuk memproses struktur-data terurut dirata-ratakan dari semua operasi.Amortized Analysis dapat digunakan untuk menunjukkan bahwa nilai operasi rata-rata cukup kecil, walaupun ada beberapa operasi yang memiliki nilai besar. Metode ini cukup berguna untuk menghitung nilai sumber daya yang terbuang secara keseluruhan.