

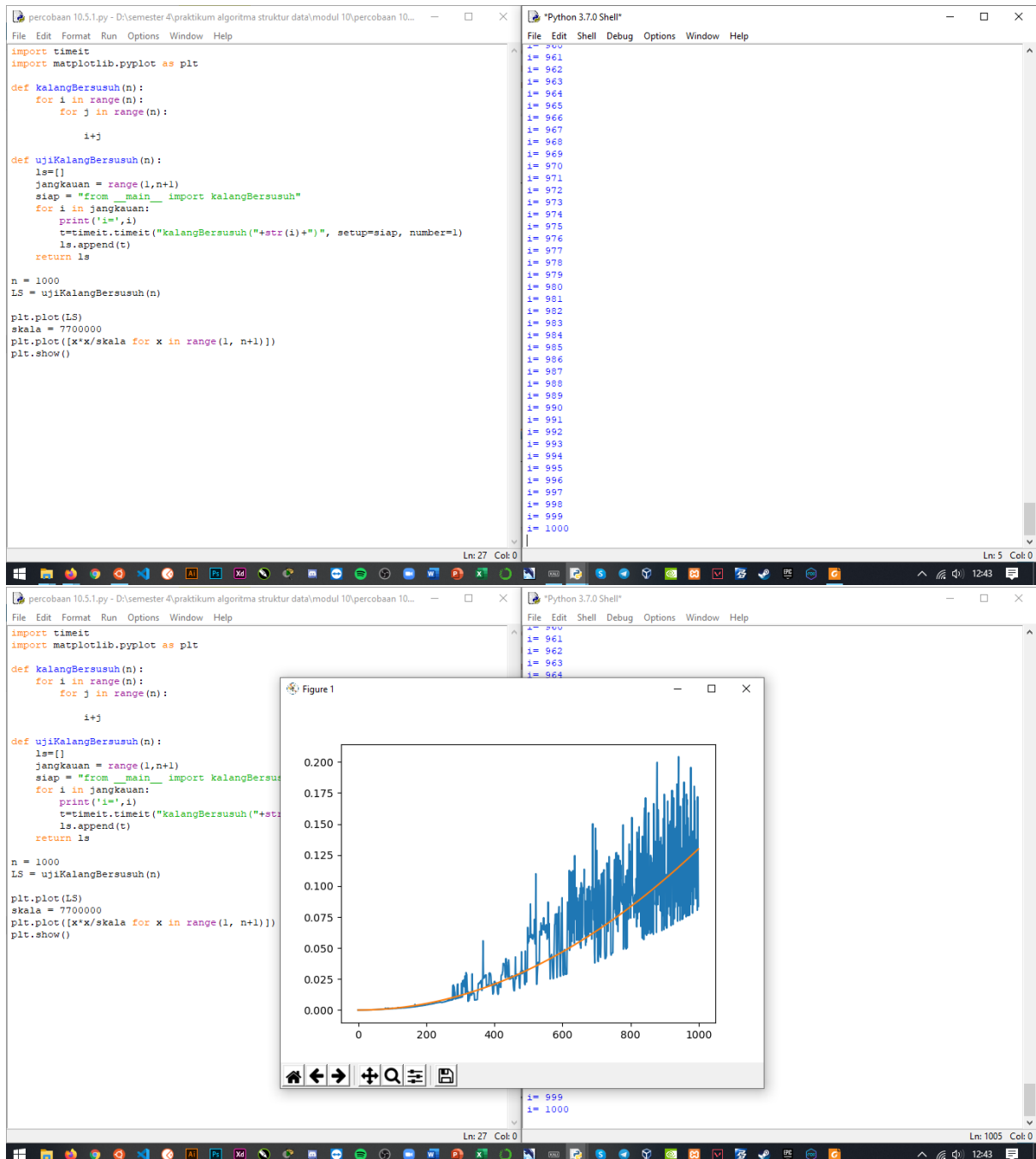
Nama : Maulana Alhif Ikhsan

NIM : L200180120

Kelas : E

Modul 10

Kegiatan



```
def swap(A, p, q):
    tmp = A[p]
    A[p] = A[q]
    A[q] = tmp

def cariPosisiTerkecil(A, dariSini, sampaiSini):
    posisiTerkecil = dariSini
    for i in range(dariSini+1, sampaiSini):
        if A[i] < A[posisiTerkecil]:
            posisiTerkecil = i
    return posisiTerkecil

A = [18, 13, 45, 25, 66, 107, 78, 89]
j = cariPosisiTerkecil(A, 2, len(A))

L = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
def bubbleSort(A):
    n = len(A)
    for i in range(n-1):
        for j in range(n-i-1):
            if A[j] > A[j+1]:
                swap(A, j, j+1)
    bubbleSort(L)

L = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
def selectionSort(A):
    n = len(A)
    for i in range(n-1):
        indexKecil = cariPosisiTerkecil(A, i, n)
        if indexKecil != i:
            swap(A, i, indexKecil)
    selectionSort(L)

L = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
def insertionSort(A):
    n = len(A)
    for i in range(1, n):
        nilai = A[i]
        pos = i
        while pos > 0 and nilai < A[pos-1]:
            swap(A, pos, pos-1)
            pos = pos-1
```

Ln: 1 Col: 0

```
##def jumlahkan_cara_1(n):
##    hasilnya = 0
##    for i in range(1, n+1):
##        hasilnya = hasilnya + i
##    return hasilnya
##
##print(jumlahkan_cara_1(10))
##print(jumlahkan_cara_1(100))

##import time
##
##def jumlahkan_cara_1(n):
##    hasilnya = 0
##    for i in range(1, n+1):
##        hasilnya = hasilnya + i
##    return hasilnya
##
##for i in range(5):
##    awal = time.time()
##    h = jumlahkan_cara_1(10000)
##    akhir = time.time()
##    print('Jumlah adalah %d, memerlukan %9.8f detik' % (h, akhir-awal))

##import time
##
##def jumlahkan_cara_1(n):
##    hasilnya = 0
##    for i in range(1, n+1):
##        hasilnya = hasilnya + i
##    return hasilnya
##
##for i in range(5):
##    awal = time.time()
##    h = jumlahkan_cara_1(1000000)
##    akhir = time.time()
##    print('Jumlah adalah %d, memerlukan %9.8f detik' % (h, akhir-awal))
##
```

Ln: 1 Col: 0

percobaan 10.3-10.4.py - D:\semester 4\praktikum algoritma struktur data\modul 10\percobaan 10.3-10.4.py (3.7.0)
File Edit Format Run Options Window Help

```
##10.3
import time
from praktikum_5 import *
from Tugas3 import *

for i in range(5):
    L = list(range(3000))
    kocok(L)
    awal = time.time()
    U = insertionSort(L)
    akhir = time.time()
    print('Jumlah adalah %d bilangan, memerlukan %0.7f detik' %(len(L), akhir-awal))

for i in range(5):
    L = list(range(3000))
    L = L[::-1]
    awal = time.time()
    U = insertionSort(L)
    akhir = time.time()
    print('Jumlah adalah %d bilangan, memerlukan %0.7f detik' %(len(L), akhir-awal))

for i in range(5):
    L = list(range(3000))

    awal = time.time()
    U = insertionSort(L)
    akhir = time.time()
    print('Jumlah adalah %d bilangan, memerlukan %0.7f detik' %(len(L), akhir-awal))

##10.4
count = 0
i = 32
while i >= 1:
    count = count + 1
    i = i//2
print(count)
```

Ln: 1 Col: 0

percobaan 10.3-10.4.py - D:\semester 4\praktikum algoritma struktur data\modul 10\percobaan 10.3-10.4.py (3.7.0)
File Edit Format Run Options Window Help

```
##10.3
import time
from praktikum_5 import *
from Tugas3 import *

for i in range(5):
    L = list(range(3000))
    kocok(L)
    awal = time.time()
    U = insertionSort(L)
    akhir = time.time()
    print('Jumlah adalah %d bilangan, m

for i in range(5):
    L = list(range(3000))
    L = L[::-1]
    awal = time.time()
    U = insertionSort(L)
    akhir = time.time()
    print('Jumlah adalah %d bilangan, m

for i in range(5):
    L = list(range(3000))

    awal = time.time()
    U = insertionSort(L)
    akhir = time.time()
    print('Jumlah adalah %d bilangan, m

##10.4
count = 0
i = 32
while i >= 1:
    count = count + 1
    i = i//2
print(count)
```

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: D:\semester 4\praktikum algoritma struktur data\modul 10\percobaan 10.3-10.4.py
bubble: 8.48428 detik
selection: 3.30414 detik
```

Ln: 27 Col: 0

```

from time import time as detik
from random import shuffle as kocok

def swap(A, p, q):
    tmp = A[p]
    A[p] = A[q]
    A[q] = tmp

def cariPosisiYangTerkecil(A, dariSini, sampaiSini):
    posisiYangTerkecil = dariSini
    for i in range(dariSini+1, sampaiSini):
        if A[i] < A[posisiYangTerkecil]:
            posisiYangTerkecil = i
    return posisiYangTerkecil

def bubbleSort(A):
    n = len(A)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if A[j] > A[j+1]:
                swap(A, j, j+1)

def selectionSort(A):
    n = len(A)
    for i in range(n - 1):
        indexKecil = cariPosisiYangTerkecil(A, i, n)
        if indexKecil != i:
            swap(A, i, indexKecil)

def insertionSort(A):
    n = len(A)
    for i in range(1, n):
        nilai = A[i]
        pos = i
        while pos > 0 and nilai < A[pos - 1]:
            A[pos] = A[pos - 1]
            pos = pos - 1
        A[pos] = nilai

k = [i for i in range(1, 6001)]
kocok(k)
u_bub = k[:]

```

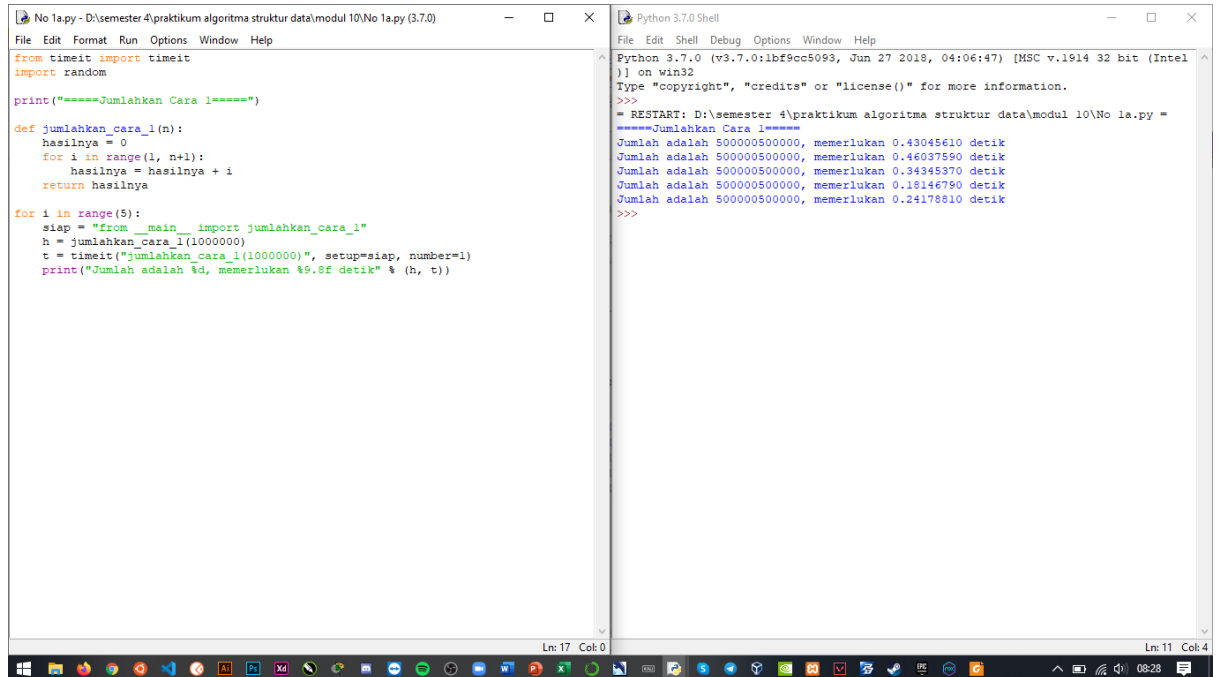
```

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: D:\semester 4\praktikum algoritma struktur data\modul 10\Tugas3.py
bubble: 7.48135 detik
selection: 3.78811 detik
insertion: 4.60483 detik
>>>

```

Latihan

1.



The screenshot shows a Python IDE with two windows. The left window, titled 'No 1a.py', contains a script that defines a function `jumlahkan_cara_1(n)` to calculate the sum of the first `n` natural numbers. The function uses a loop to accumulate the sum. The script then calls the function with `n=1000000` and prints the result and the time taken. The right window, titled 'Python 3.7.0 Shell', shows the execution output. It displays the function definition, the input value `n=1000000`, and the calculated sum `500000500000`. It also shows the time taken for the calculation, which is approximately 0.43 seconds.

```
File Edit Format Run Options Window Help
No 1a.py - D:\semester 4\praktikum algoritma struktur data\modul 10\No 1a.py (3.7.0)

from timeit import timeit
import random

print("====Jumlahkan Cara 1====")

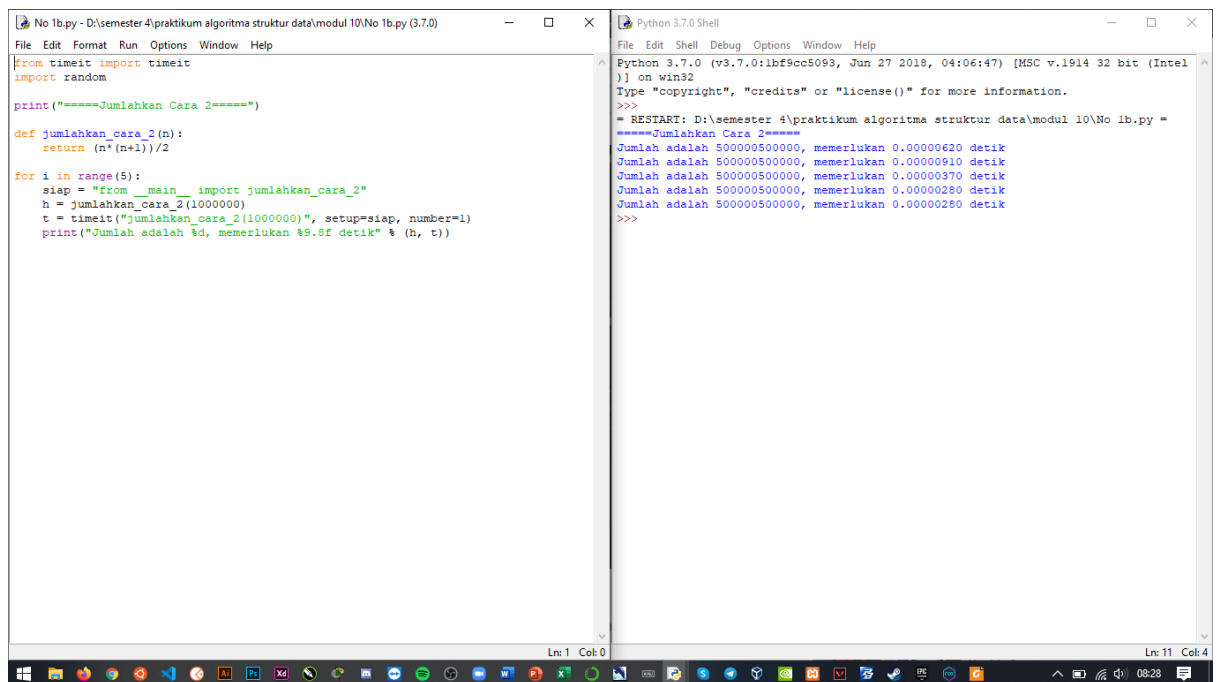
def jumlahkan_cara_1(n):
    hasilnya = 0
    for i in range(1, n+1):
        hasilnya = hasilnya + i
    return hasilnya

for i in range(5):
    siap = "from __main__ import jumlahkan_cara_1"
    h = jumlahkan_cara_1(1000000)
    t = timeit("jumlahkan_cara_1(1000000)", setup=siap, number=1)
    print("Jumlah adalah %d, memerlukan %.8f detik" % (h, t))

Ln: 17 Col: 0
```

```
File Edit Shell Debug Options Window Help
Python 3.7.0 Shell

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel
)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\semester 4\praktikum algoritma struktur data\modul 10\No 1a.py =
====Jumlahkan Cara 1====
Jumlah adalah 500000500000, memerlukan 0.43045610 detik
Jumlah adalah 500000500000, memerlukan 0.46037590 detik
Jumlah adalah 500000500000, memerlukan 0.34345370 detik
Jumlah adalah 500000500000, memerlukan 0.18146790 detik
Jumlah adalah 500000500000, memerlukan 0.24178810 detik
>>>
```



The screenshot shows a Python IDE with two windows. The left window, titled 'No 1b.py', contains a script that defines a function `jumlahkan_cara_2(n)` to calculate the sum of the first `n` natural numbers using the formula $n(n+1)/2$. The script then calls the function with `n=1000000` and prints the result and the time taken. The right window, titled 'Python 3.7.0 Shell', shows the execution output. It displays the function definition, the input value `n=1000000`, and the calculated sum `500000500000`. It also shows the time taken for the calculation, which is approximately 0.000062 seconds.

```
File Edit Format Run Options Window Help
No 1b.py - D:\semester 4\praktikum algoritma struktur data\modul 10\No 1b.py (3.7.0)

from timeit import timeit
import random

print("====Jumlahkan Cara 2====")

def jumlahkan_cara_2(n):
    return (n*(n+1))/2

for i in range(5):
    siap = "from __main__ import jumlahkan_cara_2"
    h = jumlahkan_cara_2(1000000)
    t = timeit("jumlahkan_cara_2(1000000)", setup=siap, number=1)
    print("Jumlah adalah %d, memerlukan %.8f detik" % (h, t))

Ln: 1 Col: 0
```

```
File Edit Shell Debug Options Window Help
Python 3.7.0 Shell

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel
)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\semester 4\praktikum algoritma struktur data\modul 10\No 1b.py =
====Jumlahkan Cara 2====
Jumlah adalah 500000500000, memerlukan 0.00000620 detik
Jumlah adalah 500000500000, memerlukan 0.00000910 detik
Jumlah adalah 500000500000, memerlukan 0.00000370 detik
Jumlah adalah 500000500000, memerlukan 0.00000280 detik
Jumlah adalah 500000500000, memerlukan 0.00000280 detik
>>>
```

```
No 1c.py - D:\semester 4\praktikum algoritma struktur data\modul 10\No 1c.py (3.7.0)
File Edit Format Run Options Window Help
from timeit import timeit
import random

print("====Insertion Sort====")

def insertionSort(A):
    n = len(A)
    for i in range(1, n):
        nilai = A[i]
        pos = i
        while pos > 0 and nilai < A[pos - 1]:
            A[pos] = A[pos - 1]
            pos = pos - 1
        A[pos] = nilai

for i in range(5):
    siap = "from __main__ import insertionSort,L"
    L = list(range(3000))
    random.shuffle(L)
    t = timeit("insertionSort(L)", setup=siap, number=1)
    print("Mengurutkan %d bilangan, memerlukan %8.7f detik" % (len(L), t))

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel
)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\semester 4\praktikum algoritma struktur data\modul 10\No 1c.py =
====Insertion Sort====
Mengurutkan 3000 bilangan, memerlukan 1.8825942 detik
Mengurutkan 3000 bilangan, memerlukan 2.5478720 detik
Mengurutkan 3000 bilangan, memerlukan 2.0817900 detik
Mengurutkan 3000 bilangan, memerlukan 2.7555630 detik
Mengurutkan 3000 bilangan, memerlukan 2.3056474 detik
>>>
```

2.

```
No 2.py - D:\semester 4\praktikum algoritma struktur data\modul 10\No 2.py (3.7.0)
File Edit Format Run Options Window Help
from timeit import timeit
import random

print("====Sorted average case====")

for i in range(5):
    g = list(range(3000))
    random.shuffle(g)
    t = timeit("sorted(g)", setup="from __main__ import g", number=1)
    print("Mengurutkan %d bilangan, memerlukan %8.7f detik" % (len(g), t))

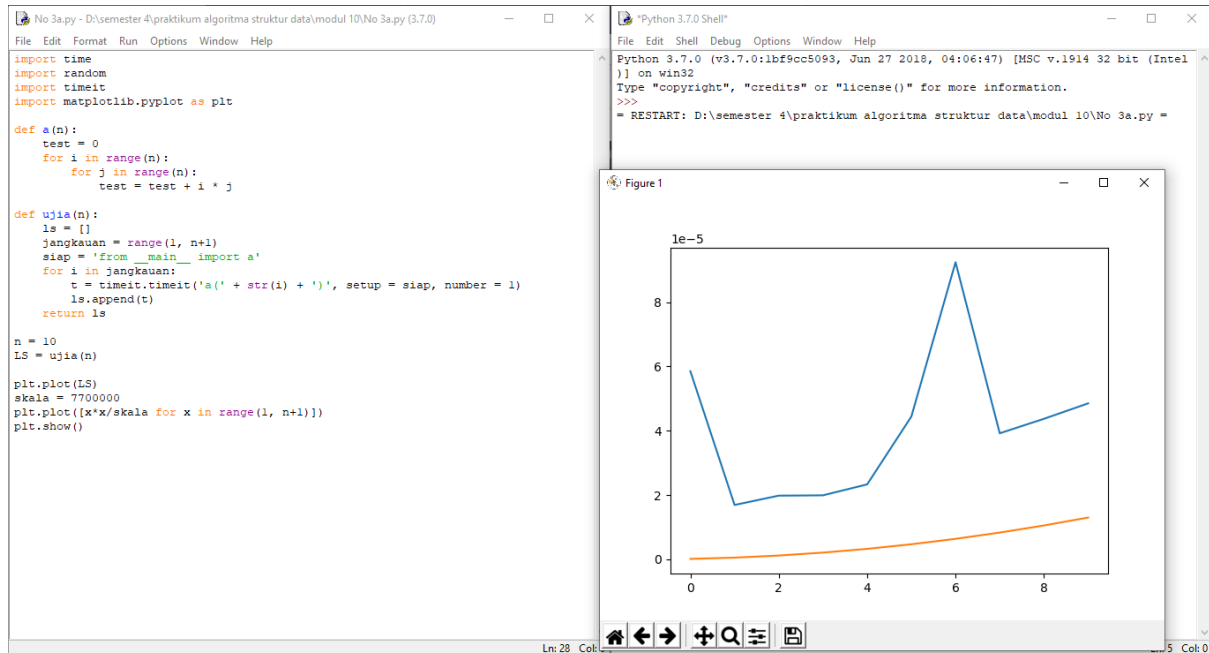
print("====Sorted best case====")

for i in range(5):
    g = list(range(3000))
    t = timeit("sorted(g)", setup="from __main__ import g", number=1)
    print("Mengurutkan %d bilangan, memerlukan %8.7f detik" % (len(g), t))

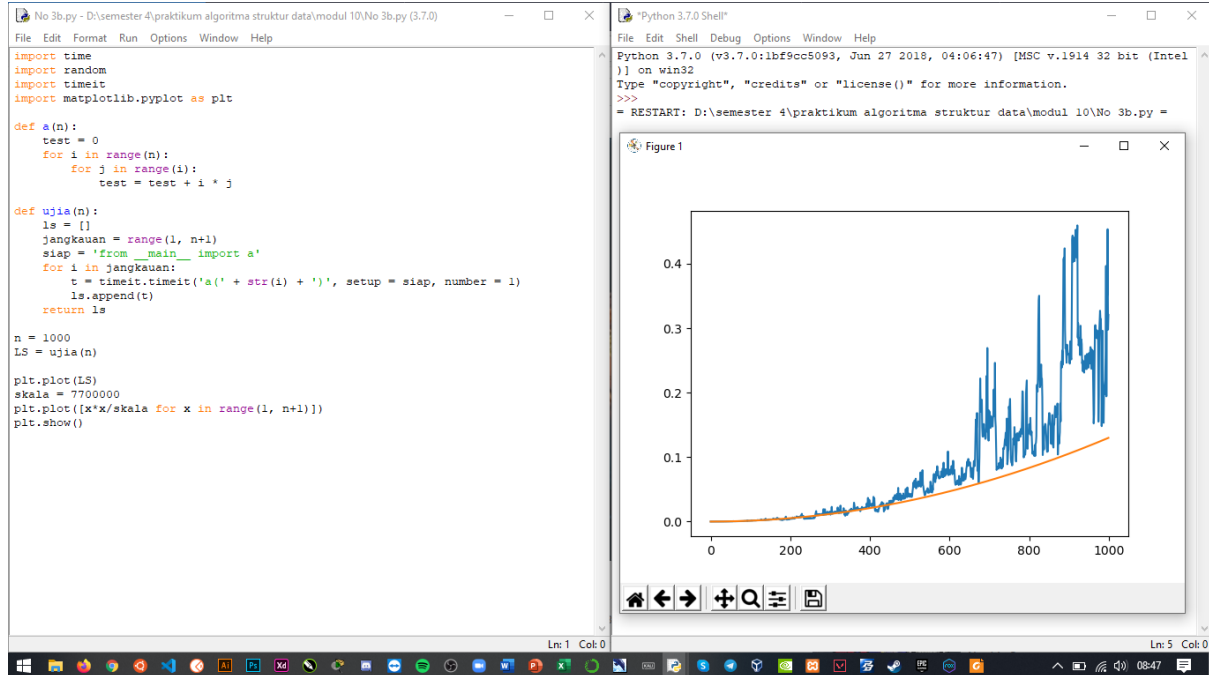
print("====Sorted worst case====")

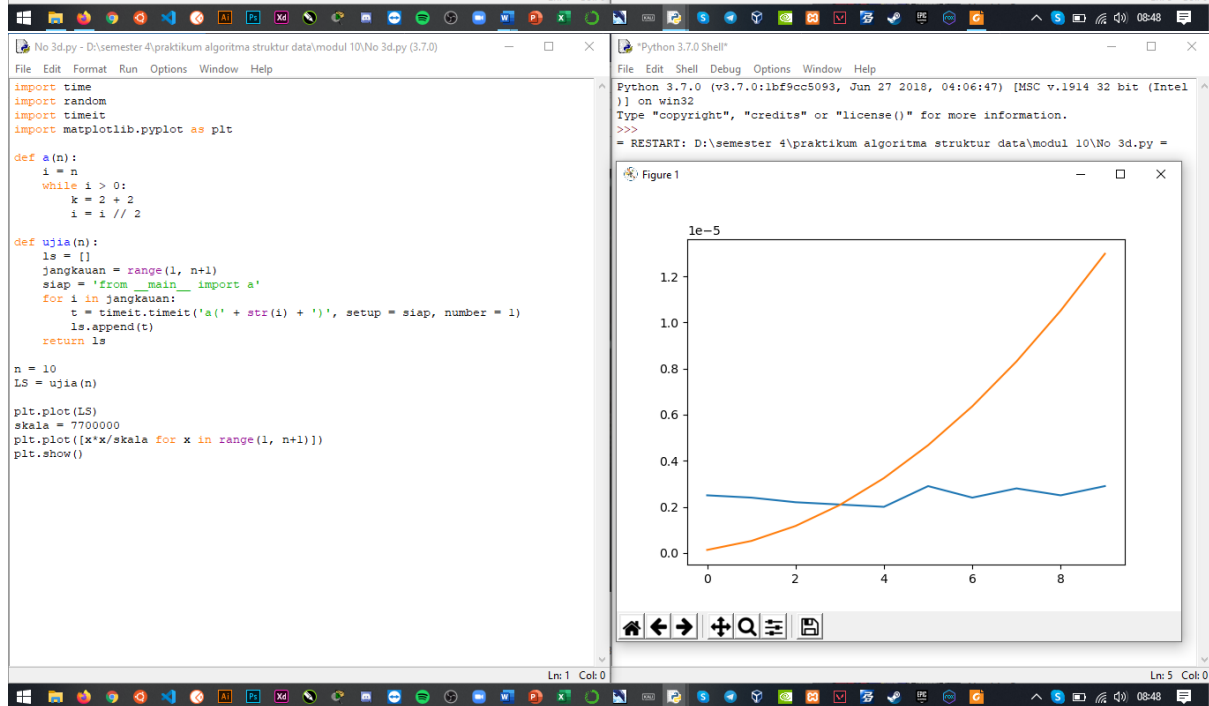
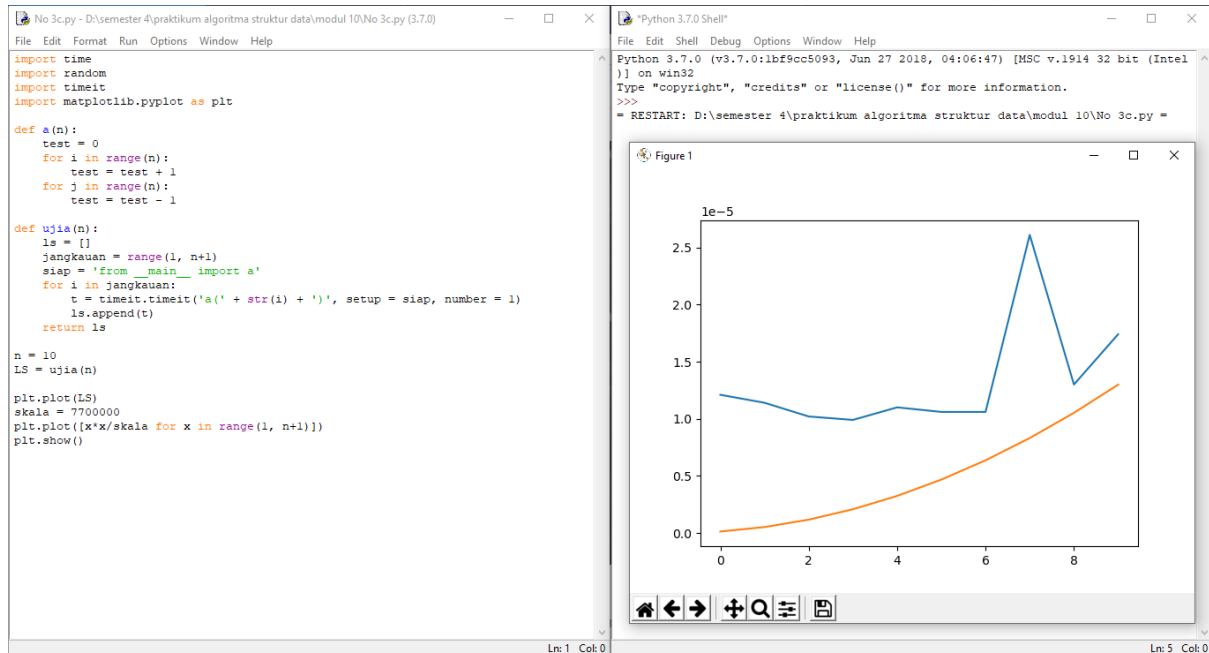
for i in range(5):
    g = list(range(3000))
    g = g[::-1]
    t = timeit("sorted(g)", setup="from __main__ import g", number=1)
    print("Mengurutkan %d bilangan, memerlukan %8.7f detik" % (len(g), t))

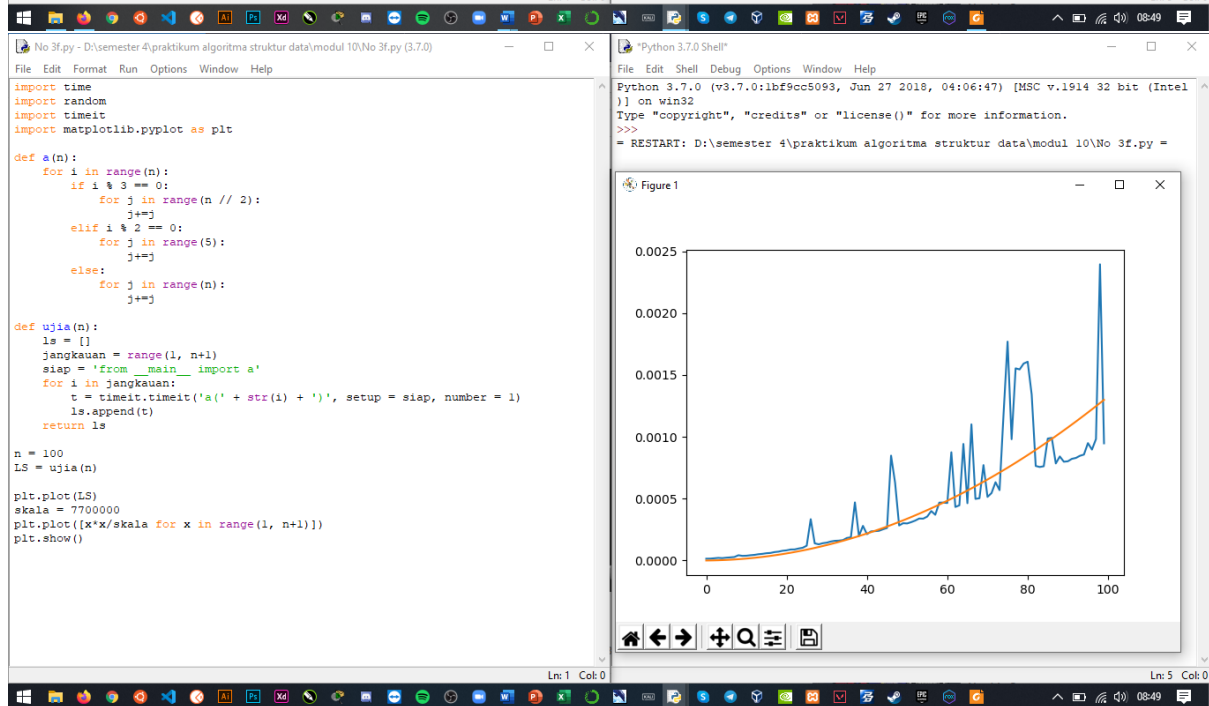
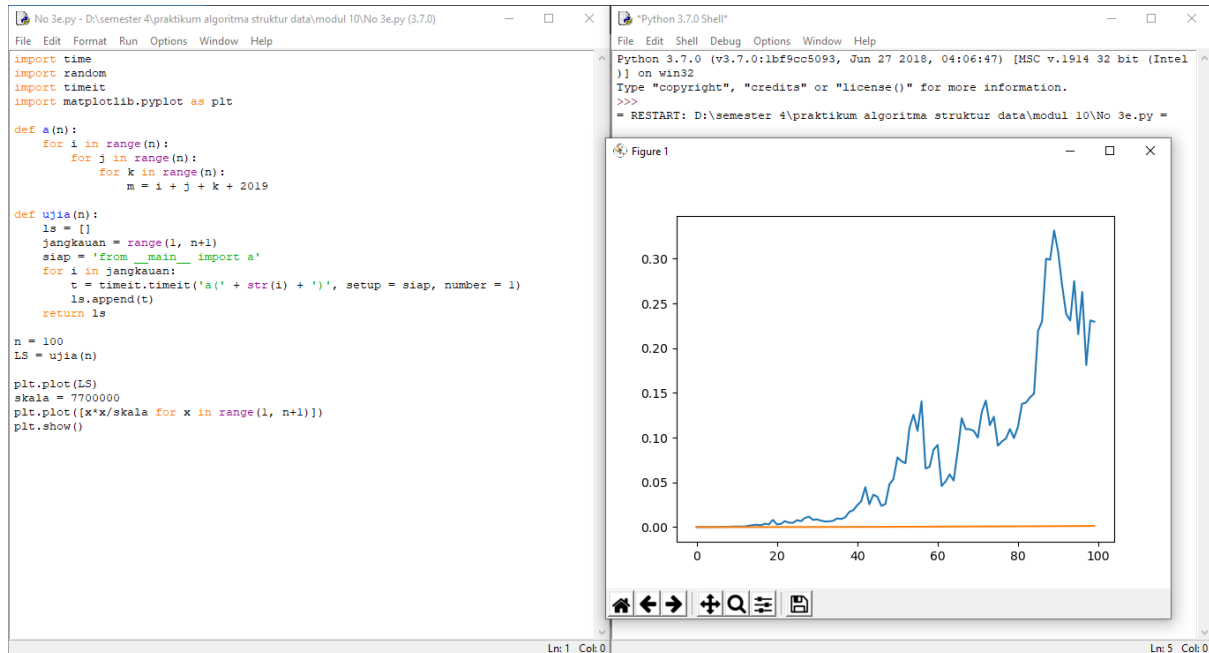
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel
)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\semester 4\praktikum algoritma struktur data\modul 10\No 2.py =
====Sorted average case====
Mengurutkan 3000 bilangan, memerlukan 0.0016172 detik
Mengurutkan 3000 bilangan, memerlukan 0.0018327 detik
Mengurutkan 3000 bilangan, memerlukan 0.0017011 detik
Mengurutkan 3000 bilangan, memerlukan 0.0016097 detik
Mengurutkan 3000 bilangan, memerlukan 0.0015800 detik
====Sorted best case====
Mengurutkan 3000 bilangan, memerlukan 0.0000995 detik
Mengurutkan 3000 bilangan, memerlukan 0.0001034 detik
Mengurutkan 3000 bilangan, memerlukan 0.0001048 detik
Mengurutkan 3000 bilangan, memerlukan 0.0001054 detik
Mengurutkan 3000 bilangan, memerlukan 0.0001045 detik
====Sorted worst case====
Mengurutkan 3000 bilangan, memerlukan 0.0000993 detik
Mengurutkan 3000 bilangan, memerlukan 0.0000970 detik
Mengurutkan 3000 bilangan, memerlukan 0.0001963 detik
Mengurutkan 3000 bilangan, memerlukan 0.0001077 detik
Mengurutkan 3000 bilangan, memerlukan 0.0000979 detik
>>>
```

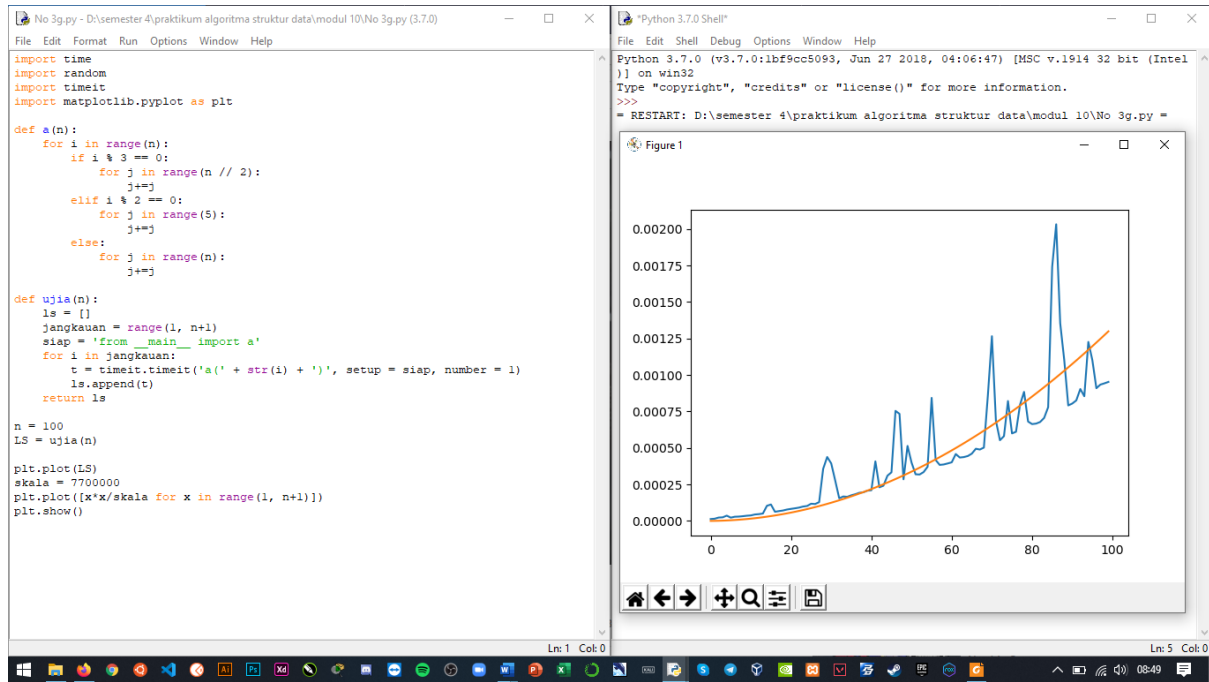


3.







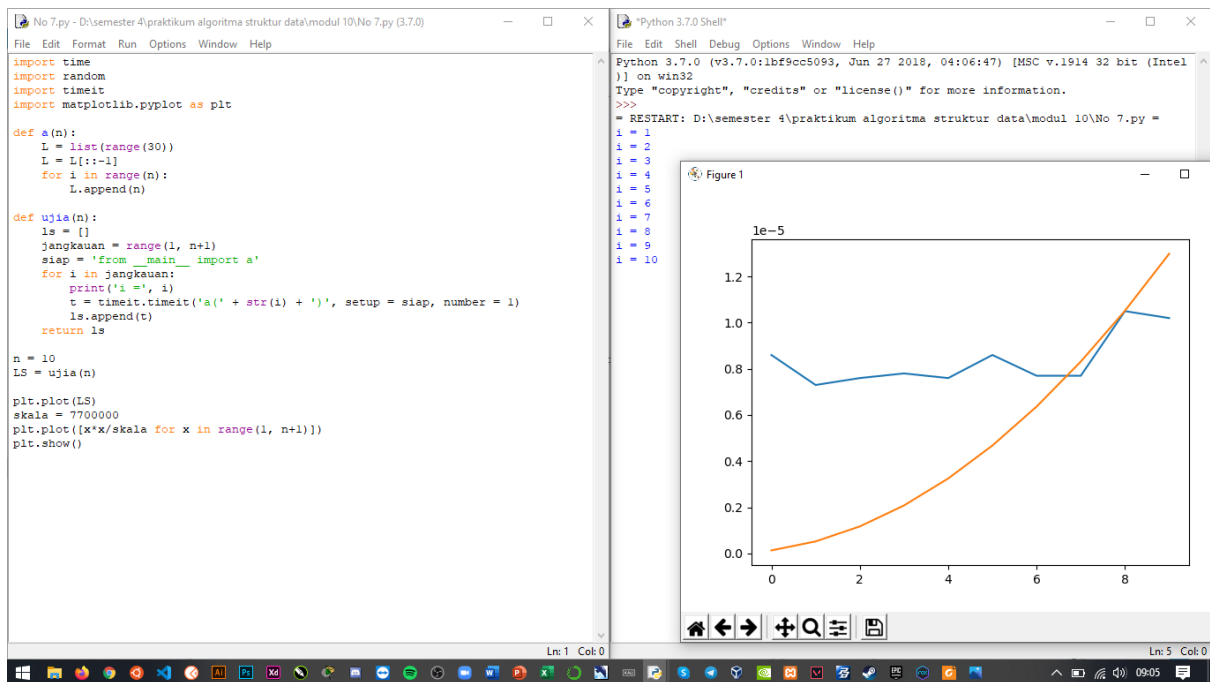




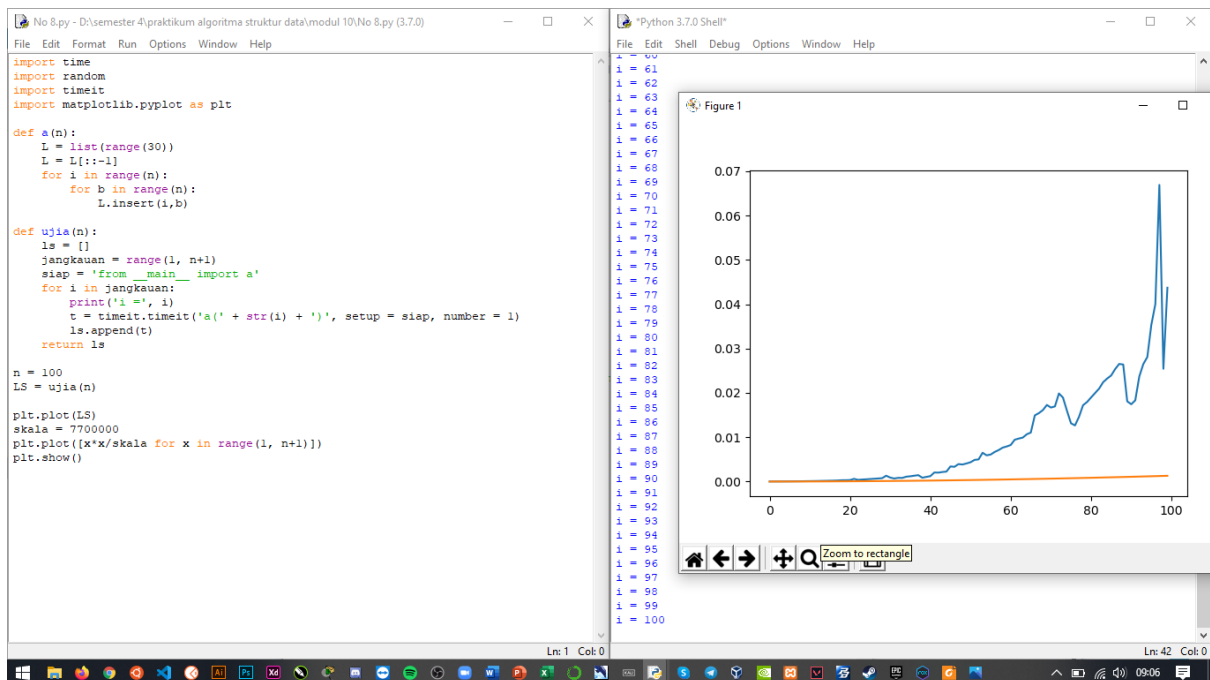
4. Urutkan dari yang pertumbuhan kompleksitasnya lambat ke yang cepat $\log_4 n < 10 \log_2 n < n \log_2 n < 2 \log_2 n < 5n^2 < n^3 < 12n^6 < 4n$
5. Tentukan $O(\cdot)$ dari fungsi-fungsi berikut, yang mewakili banyaknya langkah yang diperlukan untuk beberapa algoritma
- $T(n) = n^2 + 32n + 8 = O(n^2)$
 - $T(n) = 87n + 8n = O(n)$
 - $T(n) = 4n + 5n \log n + 102 = O(n \log n)$
 - $T(n) = \log n + 3n^2 + 88 = O(n^2)$
 - $T(n) = 3(2^n) + n^2 + 647 = O(2^n)$
 - $T(n, k) = kn + \log k = O(kn)$
 - $T(n, k) = 8n + k \log n + 800 = O(n)$
 - $T(n, k) = 100kn + n = O(kn)$
6. (Literatur Review) carilah di internet, kompleksitas metode-metode pada object list di Python. Hint
- Google [python list methods complexity](https://wiki.python.org/moin/TimeComplexity). Lihat juga bagian 'Images'-nya
 - Kunjungi <https://wiki.python.org/moin/TimeComplexity>

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	$O(k)$	$O(k)$
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	$O(1)$	$O(1)$

7.



8.



No 9a.py - D:\semester 4\praktikum algoritma struktur data\modul 10\No 9a.py (3.7.0)

File Edit Format Run Options Window Help

```

import timeit
import time
import matplotlib.pyplot as plt

def carilurus(wadah, target):
    n = len(wadah)
    for i in range(n):
        if wadah[i] == target:
            return True
    return False

def tim():
    z=100
    a = [8, 7, 2, 1, 3, 2, 10]
    awal = time.time()
    U = carilurus(a, z)
    akhir=time.time()
    print("Worst case")
    print("mengurutkan %d bilangan, memerlukan %0.7f detik" %(U,akhir-awal))

tim()

```

Ln: 1 Col: 0

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

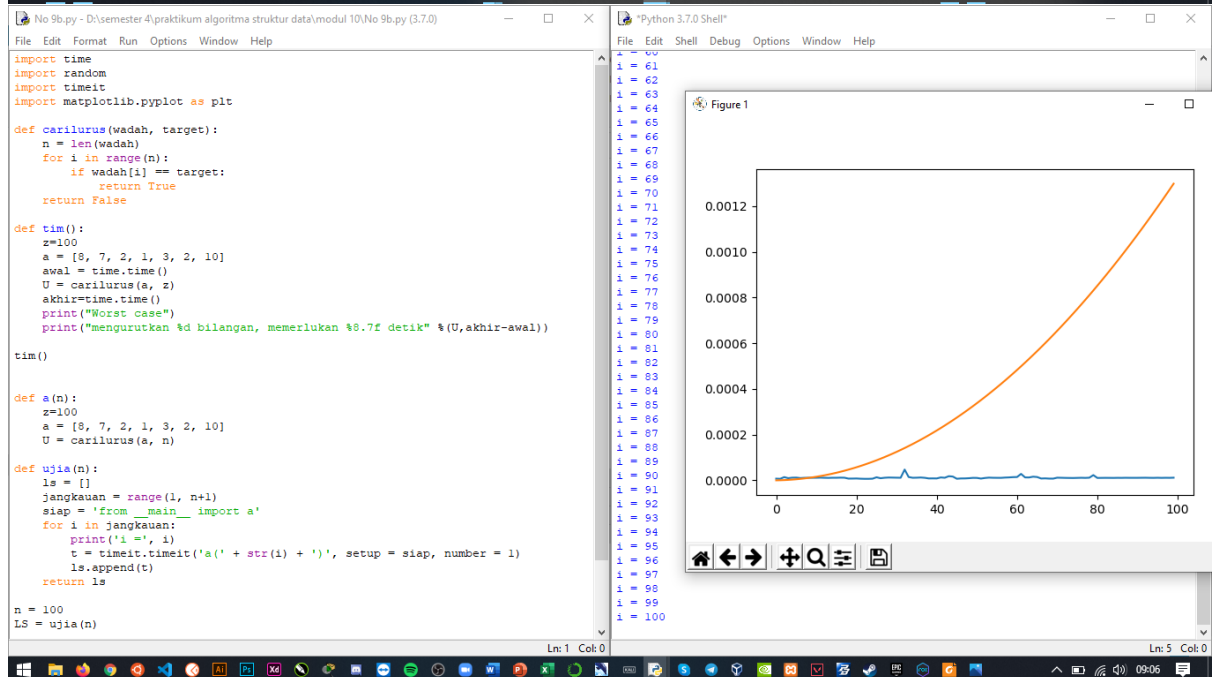
```

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel
)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\semester 4\praktikum algoritma struktur data\modul 10\No 9a.py =
Worst case
mengurutkan 0 bilangan, memerlukan 0.0000000 detik
>>>

```

Ln: 7 Col: 4

9.



10.

Operation	Average Case	Amortized Worst Case
k in d	$O(1)$	$O(n)$
Copy[2]	$O(n)$	$O(n)$
Get Item	$O(1)$	$O(n)$
Set Item[1]	$O(1)$	$O(n)$
Delete Item	$O(1)$	$O(n)$
Iteration[2]	$O(n)$	$O(n)$

11.

- Big O dilambangkan dengan notasi $O(\dots)$ merupakan keadaan terburuk (worst case). Kinerja sebuah algoritma biasanya diukur menggunakan patokan keadaan Big-O ini. Merupakan notasi asymptotic untuk batas fungsi dari atas dan bawah dengan Berperilaku mirip dengan \leq operator untuk tingkat pertumbuhan.
- Big Theta dilmbangkan dengan notasi $\Theta(\dots)$ 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.
- Big Omega dilambangkan dengan notasi $\Omega(\dots)$ merupakan notasi asymptotic untuk batas bawah dengan keadaan rata-rata(average case) yang berperilaku mirip dengan \geq operator untuk tingkat pertumbuhan.

12. 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 unyuk melakukan satu urutan operasi pada struktur data terhadap keseluruhan operasi yang dilakukan