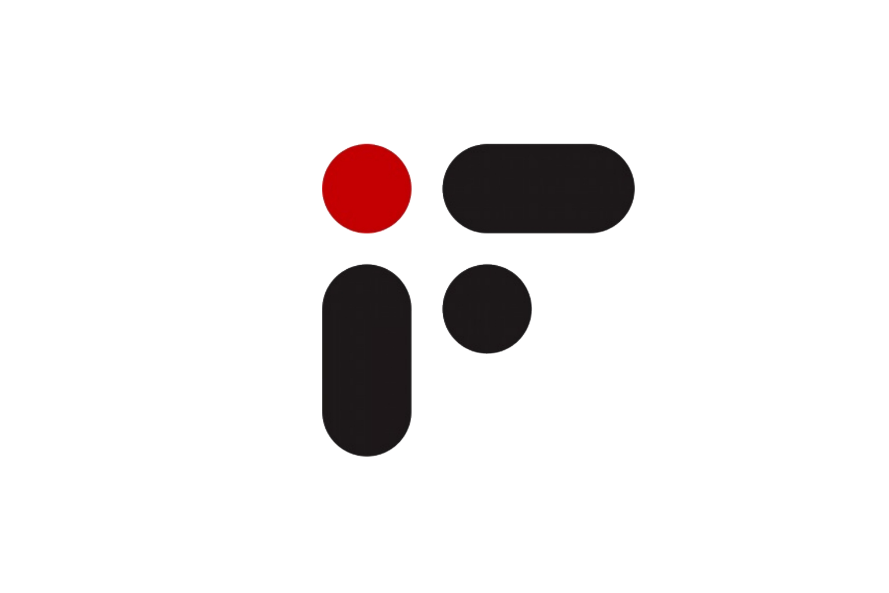
**LAPORAN PRAKTIKUM**

**PEMROGRAMAN DASAR**

**PERTEMUAN 5**

****

NAMA : Silvy Nur Azkia

NRP : 152022178

KELAS : EE

TANGGAL PENUGASAN : 26 Oktober 2023

**LABORATORIUM DASAR KOMPUTER**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**INSTITUT TEKNOLOGI NASIONAL**

**BANDUNG**

**2023**

1. Anda diberikan sebuah array dari gaji bilangan bulat unik di mana gaji[i] adalah gaji karyawan ke-i. Kembalikan gaji rata-rata karyawan tidak termasuk gaji minimum dan maksimum



=

* Source code

1. def averageSalary(salary): #fungsi averageSalary yang menerima satu parameter salary, yaitu

daftar gaji karyawan.

2. if len(salary) < 2: #jika panjang elemen dalam data salary kurang dari 2

3. return 0 #maka fungsi mengembalikan 0

4.

5. min\_salary = min(salary) #untuk menghitung gaji terendah menggunakan min

6. max\_salary = max(salary) #untuk menghitung gaji tertinggi menggunakan max

7. total = sum(salary) - min\_salary - max\_salary #menghitung total gaji semua karyawan

menggunakan sum dan kemudian mengurangkan

gaji terendah dan tertinggi untuk

menghilangkan nilai gaji minimum dan

maksimum dalam perhitungan rata-rata.

8.

9. num\_employees = len(salary) - 2 #menghitung jumlah karyawan yang ada dalam data salary

dengan mengurangkan 2 (selain gaji minimum dan

maksimum).

10.

11. average = total / num\_employees #menghitung rata-rata gaji karyawan dengan membagi total

gaji (setelah mengurangkan gaji minimum dan maksimum)

dengan jumlah karyawan

12. return average

13.

14.

15. salary = [] # inputan untuk memasukkan daftar gaji karyawan sebagai list

16.

17. n = int(input("Masukkan jumlah karyawan: ")) #inputan untuk jumlah karyawan

18. print("Masukkan gaji karyawan:")

19.

20. for i in range(n): #untuk i didalam range jumlah karyawan(n)

21. gaji = int(input(f"Gaji karyawan ke-{i+1}: ")) #digunakan untuk meminta inputan gaji

karyawan ke-(i+1).

22. salary.append(gaji) #Nilai gaji yang telah diinput akan ditambahkan ke dalam data salary

23.

24. rata\_rata\_min\_max = averageSalary(salary) #memanggil fungsi averageSalary dengan daftar

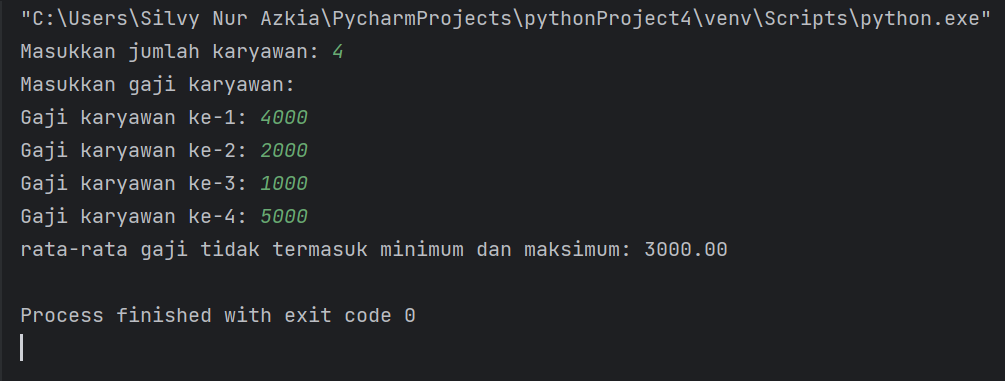
salary sebagai argumen dan menyimpan hasilnya

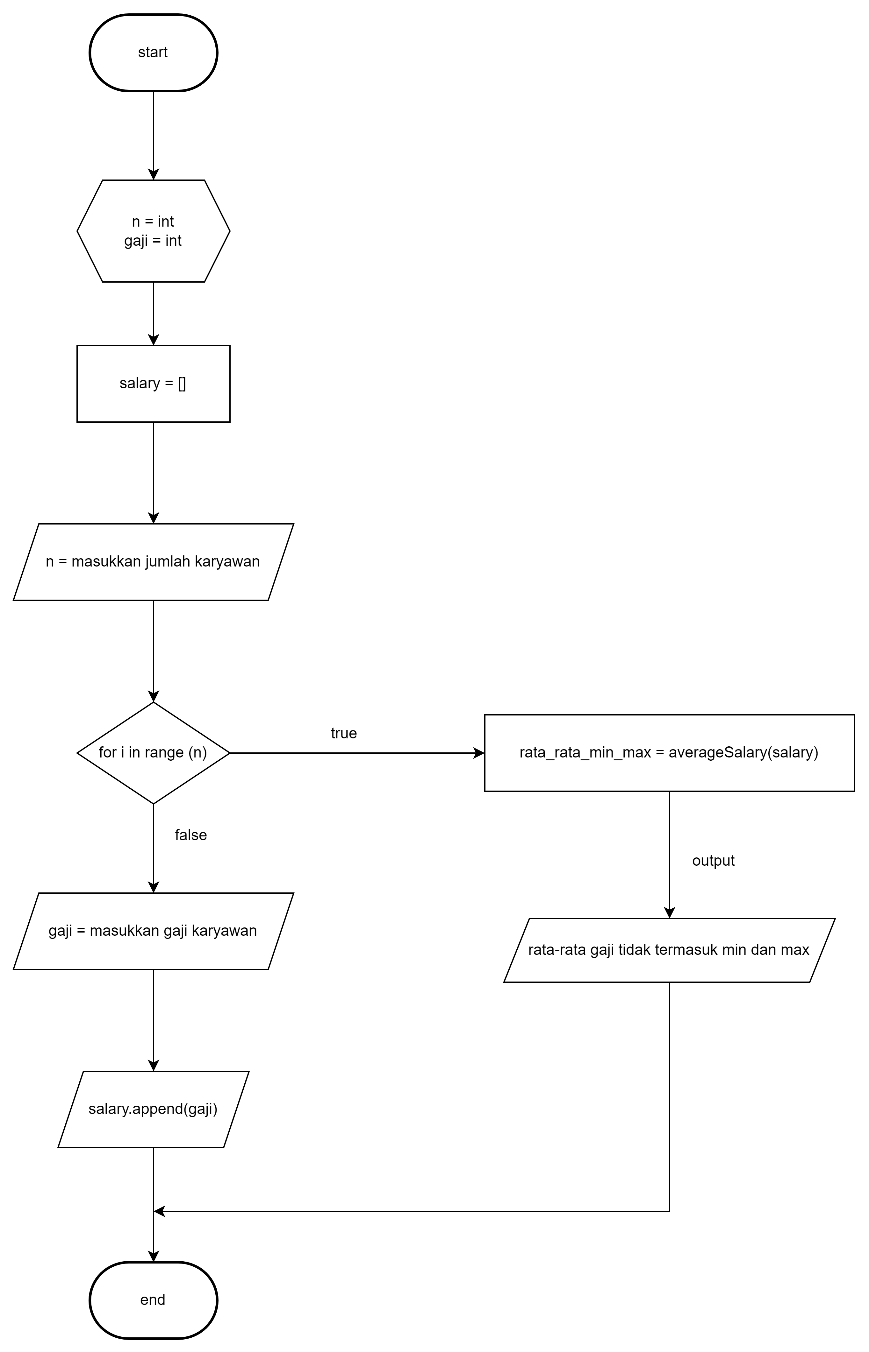
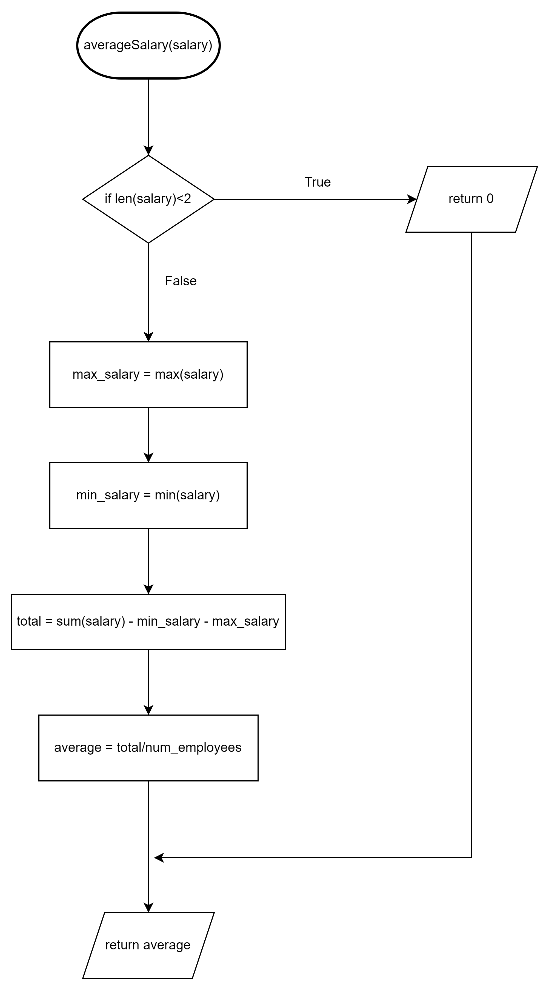
dalam variabel rata\_rata\_min\_max.

25. print("rata-rata gaji tidak termasuk minimum dan maksimum:{:.2f}".format(rata\_rata\_min\_max))

#Hasil rata-rata gaji yang tidak termasuk gaji minimum dan maksimum kemudian diprint

dengan dua angka desimal

* Output
* Flowchart



1. Diberikan sebuah matriks m x n berisi angka-angka yang berbeda, kembalikan semua angka keberuntungan dalam matriks tersebut dalam urutan apa pun. Angka keberuntungan adalah sebuah elemen dari matriks yang merupakan elemen minimum dalam baris dan maksimum dalam kolom.

=

* Source code

1. def luckyNumbers(matrix): #fungsi luckyNumbers yang berisi matriks sebagai argumen,

2. # digunakan untuk mencari lucky number pada sebuah matriks

3. luckyNums = [] #list ini digunakan untuk menyimpan lucky numbers yang ditemukan pada matriks

4.

5. for row in matrix: #Melakukan iterasi untuk setiap baris dalam matriks

6. min\_in\_row = min(row) # menghitung angka terkecil dalam baris

7. min\_index = row.index(min\_in\_row) # untuk mencari index angka terkecil dalam baris

8. col = [row[min\_index] for row in matrix] # mengambil seluruh kolom yang sesuai dengan indeks

9. max\_in\_col = max(col) # digunakan untuk mencari angka terbesar dalam kolom

10.

11. if min\_in\_row == max\_in\_col: #jika angka terkecil dalam baris sama dengan angka terbesar dalam kolom

12. luckyNums.append(min\_in\_row) #maka tambahkan angka terkecil(lucky number) ke dalam list

13.

14. return luckyNums

15.

16.

17. m = int(input("Masukkan jumlah baris: ")) #inputan untuk jumlah baris matriks

18. n = int(input("Masukkan jumlah kolom: ")) #inputan untuk jumlah kolom matriks

19.

20. matrix = [] #matriks dimasukkan kedalam list

21.

22. print("Masukkan angka pada matriks:") #inputan angka matriks

23.

24. for i in range(m): #untuk i dalam range baris

25. row = []

26. for j in range(n): #untuk j dalam range kolom

27. elemen = int(input(f"Masukkan elemen baris {i+1}, kolom {j+1}: "))#input elemen matriks

28. row.append(elemen) #menambahkan elemen kedalam baris

29. matrix.append(row) #menambahkan baris kedalam matriks

30.

31.

32. print("Masukkan Matrix: ", matrix) #print matriks yang tadi diinputkan

33.

34. for i in range(m):

35. for j in range(n):

36. print(matrix[i][j], end=" ") #menampilkan elemen-elemen pada matriks

37. print()

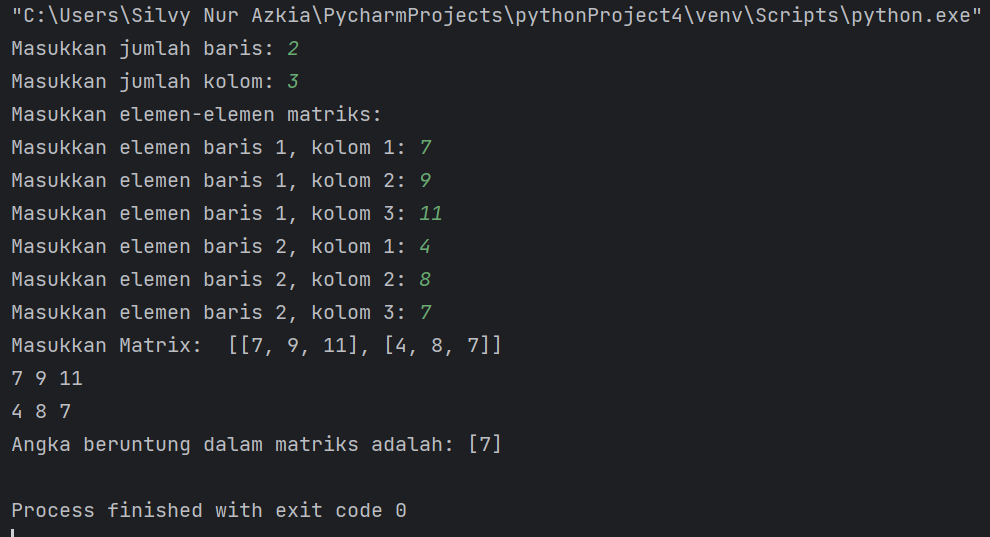
38.

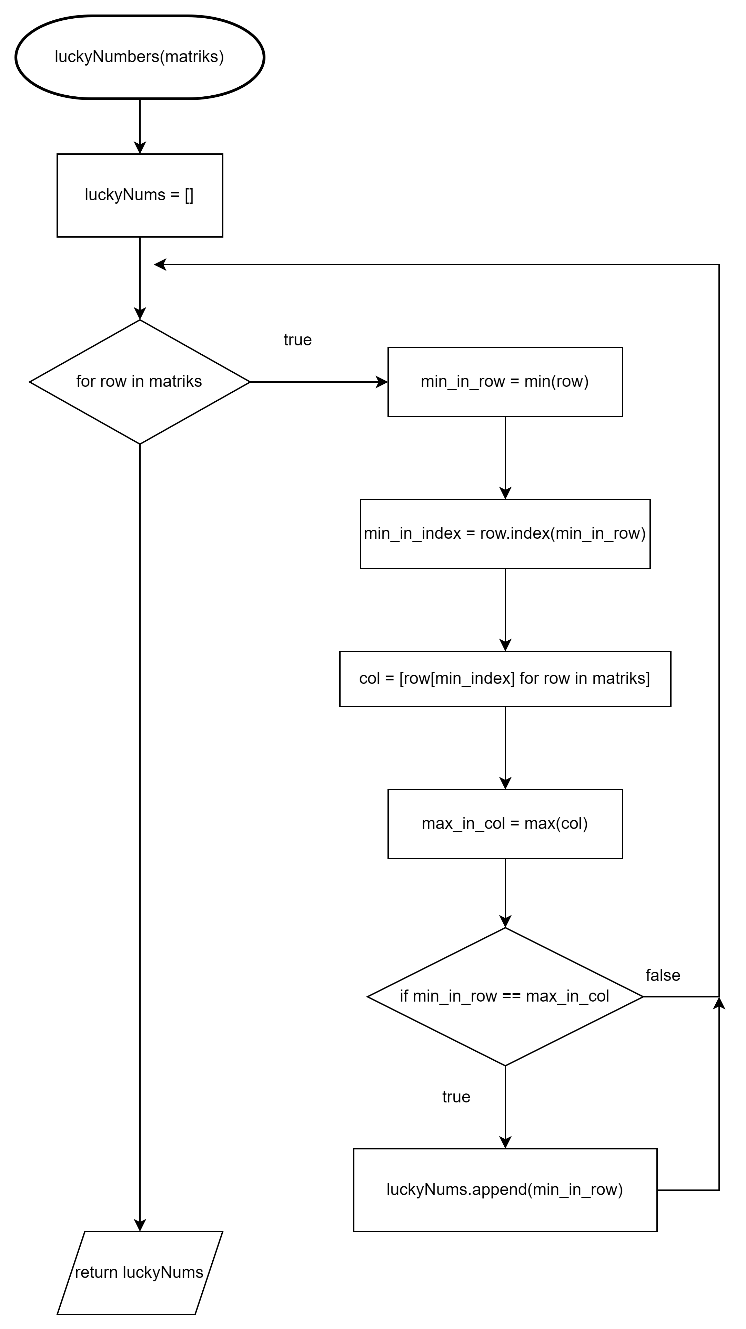
39.

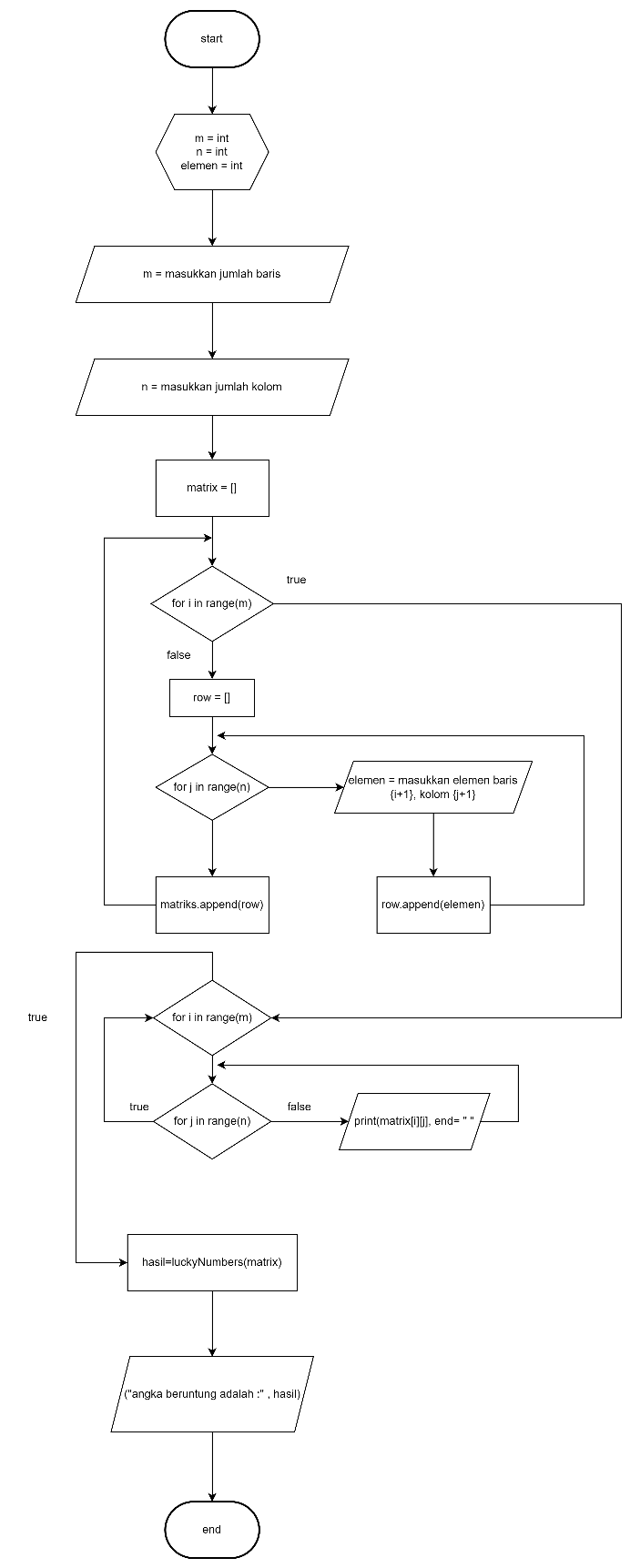
40. hasil = luckyNumbers(matrix) #memanggil fungsi luckyNumbers untuk mencari angka lucky number dalam matriks

41. print("Angka beruntung dalam matriks adalah:", hasil) #menampilkan lucky number yang ditemukan

* Output



* flowchart



1. Anda diberikan sebuah array harga di mana harga[i] adalah harga saham tertentu pada hari kei. Anda ingin memaksimalkan keuntungan dengan memilih satu hari untuk membeli satu saham dan memilih hari yang berbeda di masa depan untuk menjual saham tersebut. Kembalikan keuntungan maksimum yang dapat Anda raih dari transaksi ini. Jika Anda tidak dapat mencapai keuntungan apa pun, kembalikan 0.

=

* Source code

1. import matplotlib.pyplot as plt # Import library matplotlib untuk plotting

2.

3. def maxProfit(prices): #fungsi maxProfit yang menerima daftar harga saham sebagai argumen

4. max\_profit = 0 # Inisialisasi profit maksimum

5. buy\_day = 0 # Hari pembelian

6. sell\_day = 0 # Hari penjualan

7.

8. max\_price = max(prices) # Hitung harga saham tertinggi

9.

10. # Looping melalui setiap harga saham dengan indeks (i) dan harga (price)

11. for i, price in enumerate(prices):

12. if price != max\_price: # Jika harga saat ini bukan harga tertinggi, lanjutkan ke harga berikutnya

13. continue

14.

15. previous\_prices = prices[:i] # Ambil daftar harga sebelum harga tertinggi

16.

17. low\_price = min(previous\_prices, default=price) # Hitung harga terendah sebelum harga tertinggi

18. current\_profit = max\_price - low\_price # Hitung profit saat ini

19.

20. if max\_profit >= current\_profit: # Jika profit saat ini lebih kecil atau sama dengan profit maksimum sebelumnya

21. continue

22.

23. max\_profit = current\_profit # Perbarui profit maksimum

24. buy\_day = i # Perbarui hari pembelian

25.

26. if len(previous\_prices) > 0:

27. sell\_day = previous\_prices.index(low\_price) # Temukan indeks harga terendah sebelum harga tertinggi

28. else:

29. sell\_day = i # Jika tidak ada harga sebelumnya, hari penjualan adalah hari saat ini

30.

31. # Plot grafik harga saham

32. plt.figure(figsize=(10, 5)) # Atur ukuran plot

33. plt.plot(prices, label="Harga", marker='o', linestyle='-') # Plot harga saham

34.

35. # titik pembelian dan penjualan

36. plt.scatter([buy\_day, sell\_day], [prices[buy\_day], prices[sell\_day]], color='red', label='Beli / Jual')

37.

38. plt.title("Grafik Harga Saham dan Titik Pembelian/Jualan") # Judul plot

39. plt.xlabel("Hari") # Label sumbu X

40. plt.ylabel("Harga") # Label sumbu Y

41. plt.legend() # Menampilkan legenda

42. plt.grid() # Menampilkan grid

43.

44. plt.show() # Menampilkan plot

45.

46. return max\_profit # Kembalikan profit maksimum

47.

48. # input untuk memasukkan harga saham sebagai string

49. input\_price = input("Masukkan harga saham: ")

50.

51. input\_prices = input\_price.split() # Pisahkan angka-angka yang dimasukkan ke dalam daftar

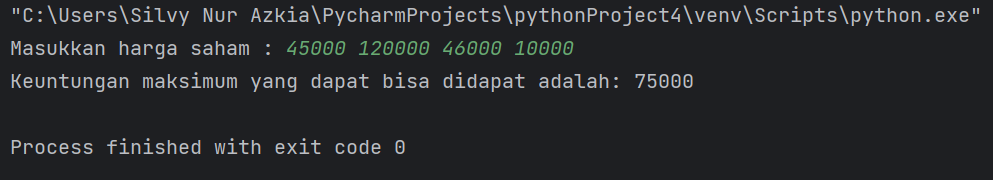
52.

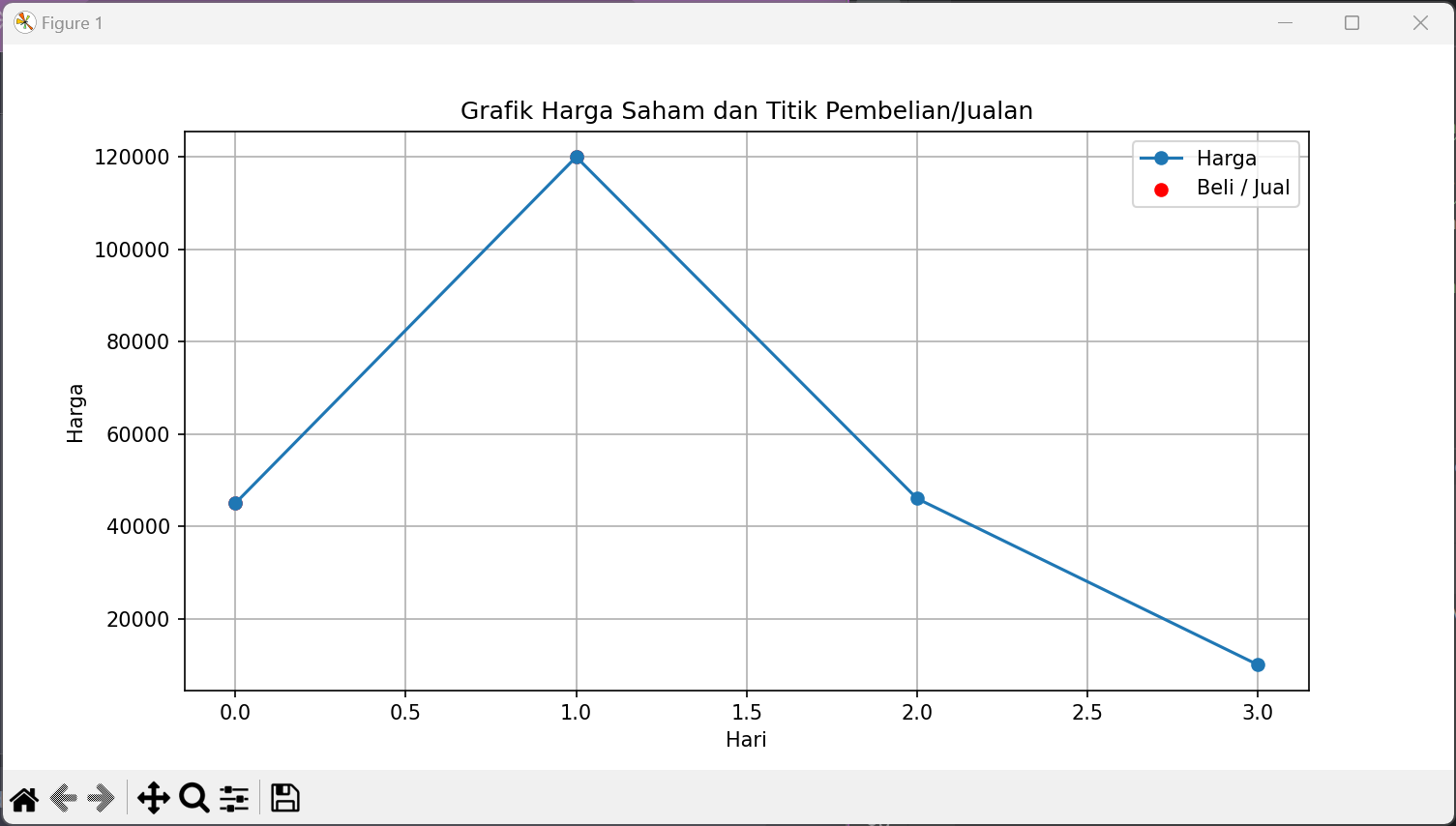
53. prices = list(map(int, input\_prices)) # Ubah string angka menjadi integer dan simpan dalam daftar

54.

55. profit = maxProfit(prices) # Hitung profit maksimum dengan memanggil fungsi maxProfit

56. print("Keuntungan maksimum yang dapat diperoleh adalah: ", profit) # Cetak profit maksimum ke layar

* Output



* Flowchart

