

PRACTICUM REPORT
ALGORITHM AND DATA STRUCTURES
MODUL 1 : PYTHON REVIEW



Disusun Oleh :

JOHN DOE

L2002XXXXXX

X

INFORMATICS ENGINEERING
FACULTY OF COMMUNICATION AND INFORMATICS
UNIVERSITAS MUHAMMADIYAH SURAKARTA
YEARS 2024/2025

1.11 Questions

1. Regarding two-dimensional arrays, we will create a matrix data type that contains numbers. For that, create functions
 - to ensure that the contents and size of the matrix are consistent (because each member of the outer list may have a different size, and may even be a different type!),
 - to take the matrix size,
 - to add two matrices (make sure the sizes match),
 - to multiply two matrices (make sure the sizes match),
 - to calculate the determinant of a square matrix.

```
1  def is_matrix_consistent(matrix):
2      """
3      Memastikan bahwa isi dan ukuran matriks konsisten.
4      """
5      if not matrix or not isinstance(matrix, list):
6          return False
7      row_length = len(matrix[0])
8      for row in matrix:
9          if not isinstance(row, list) or len(row) != row_length:
10             return False
11     return True
12
13  def matrix_size(matrix):
14      """
15      Mengambil ukuran matriks.
16      """
17      if not is_matrix_consistent(matrix):
18          raise ValueError("Matriks tidak konsisten")
19      return len(matrix), len(matrix[0])
20
21  def add_matrices(matrix1, matrix2):
22      """
23      Menambahkan dua matriks (pastikan ukurannya sama).
24      """
25      if not (is_matrix_consistent(matrix1) and is_matrix_consistent(matrix2)):
26          raise ValueError("Salah satu atau kedua matriks tidak konsisten")
27
28      rows1, cols1 = matrix_size(matrix1)
29      rows2, cols2 = matrix_size(matrix2)
30
31      if rows1 != rows2 or cols1 != cols2:
32          raise ValueError("Ukuran matriks tidak sama")
33
34      result = []
35      for i in range(rows1):
36          row = []
37          for j in range(cols1):
38              row.append(matrix1[i][j] + matrix2[i][j])
39          result.append(row)
40
41      return result
```

```

43 def multiply_matrices(matrix1, matrix2):
44     """
45     Mengalikan dua matriks (pastikan ukurannya sesuai).
46     """
47     if not (is_matrix_consistent(matrix1) and is_matrix_consistent(matrix2)):
48         raise ValueError("Salah satu atau kedua matriks tidak konsisten")
49
50     rows1, cols1 = matrix_size(matrix1)
51     rows2, cols2 = matrix_size(matrix2)
52
53     if cols1 != rows2:
54         raise ValueError("Ukuran matriks tidak sesuai untuk perkalian")
55
56     result = [[0 for _ in range(cols2)] for _ in range(rows1)]
57
58     for i in range(rows1):
59         for j in range(cols2):
60             for k in range(cols1):
61                 result[i][j] += matrix1[i][k] * matrix2[k][j]
62
63     return result
64
65 def determinant(matrix):
66     """
67     Menghitung determinan dari matriks persegi.
68     """
69     if not is_matrix_consistent(matrix):
70         raise ValueError("Matriks tidak konsisten")
71
72     rows, cols = matrix_size(matrix)
73
74     if rows != cols:
75         raise ValueError("Matriks bukan persegi")
76
77     if rows == 1:
78         return matrix[0][0]
79
80     if rows == 2:
81         return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]
82
83     det = 0

```

```

84         for c in range(cols):
85             sub_matrix = [row[:c] + row[c+1:] for row in matrix[1:]]
86             det += ((-1) ** c) * matrix[0][c] * determinant(sub_matrix)
87
88         return det
89
90     # Contoh penggunaan
91     matrix1 = [
92         [1, 2],
93         [3, 4]
94     ]
95
96     matrix2 = [
97         [5, 6],
98         [7, 8]
99     ]
100
101     print("Matrix 1:", matrix1)
102     print("Matrix 2:", matrix2)
103     print("Penjumlahan Matriks:", add_matrices(matrix1, matrix2))
104     print("Perkalian Matriks:", multiply_matrices(matrix1, matrix2))
105     print("Determinan Matriks 1:", determinant(matrix1))

```

Picture 1.1 the code 1.py

```

PS D:\Semester 4\PrakAl_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAl_and_StrDat/Module_3/Question/1.py"
Matrix 1: [[1, 2], [3, 4]]
Matrix 2: [[5, 6], [7, 8]]
Penjumlahan Matriks: [[6, 8], [10, 12]]
Perkalian Matriks: [[19, 22], [43, 50]]
Determinan Matriks 1: -2

```

Picture 1.2 the output 1.py

2. Regarding the matrix and list comprehension, create (using list comprehension) functions

- to generate a matrix containing all zeros, given its size. calling: `buatNol(m,n)` and `buatNol(m)`. Calling the latter method will provide a square matrix of sizes $m \times m$.
- to generate an identity matrix, given its size. Call: `buatIdentitas(m)`.

```
Module_3 > Question > 2.py > ...
1  def buatNol(m, n=None):
2      """
3      Menghasilkan matriks yang berisi semua nol.
4      Jika hanya satu argumen diberikan, menghasilkan matriks persegi m x m.
5      """
6      if n is None:
7          n = m
8      return [[0 for _ in range(n)] for _ in range(m)]
9
10 def buatIdentitas(m):
11     """
12     Menghasilkan matriks identitas berukuran m x m.
13     """
14     return [[1 if i == j else 0 for j in range(m)] for i in range(m)]
15
16 # Contoh penggunaan
17 print("Matriks Nol 3x3:")
18 print(buatNol(3))
19 print("\nMatriks Nol 2x4:")
20 print(buatNol(2, 4))
21 print("\nMatriks Identitas 3x3:")
22 print(buatIdentitas(3))
```

Picture 2.1 the code 2.py

```
PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Module_3/Question/2.py"
Matriks Nol 3x3:
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

Matriks Nol 2x4:
[[0, 0, 0, 0], [0, 0, 0, 0]]

Matriks Identitas 3x3:
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
```

Picture 2.2 the output 2.py

3. Regarding the linked list, create a function for

- search for data with certain contents: cari(head,yang dicari)
- add a node at the beginning: tambahDepan(head)
- add a node at the end: tambahAkhir(head)
- insert a node anywhere: tambah(head,posisi)
- delete a node at the beginning, at the end, or anywhere: hapus(posisi)

Module_3 > Question > 3.py > ...

```
1 class Node:
2     def __init__(self, data=None):
3         self.data = data
4         self.next = None
5
6 class LinkedList:
7     def __init__(self):
8         self.head = None
9
10    def cari(self, yang_dicari):
11        current = self.head
12        while current is not None:
13            if current.data == yang_dicari:
14                return True
15            current = current.next
16        return False
17
18    def tambahDepan(self, data):
19        new_node = Node(data)
20        new_node.next = self.head
21        self.head = new_node
22
23    def tambahAkhir(self, data):
24        new_node = Node(data)
25        if self.head is None:
26            self.head = new_node
27            return
28        last = self.head
29        while last.next:
30            last = last.next
31        last.next = new_node
32
33    def tambah(self, data, posisi):
34        new_node = Node(data)
35        if posisi == 0:
36            new_node.next = self.head
37            self.head = new_node
38            return
39        current = self.head
40        for _ in range(posisi - 1):
41            if current is None:
42                raise IndexError("Posisi di luar jangkauan")
```

```

43         current = current.next
44     new_node.next = current.next
45     current.next = new_node
46
47     def hapus(self, posisi):
48         if self.head is None:
49             raise IndexError("List kosong")
50         if posisi == 0:
51             self.head = self.head.next
52             return
53         current = self.head
54         for _ in range(posisi - 1):
55             if current.next is None:
56                 raise IndexError("Posisi di luar jangkauan")
57             current = current.next
58         if current.next is None:
59             raise IndexError("Posisi di luar jangkauan")
60         current.next = current.next.next
61
62     def cetak(self):
63         current = self.head
64         while current:
65             print(current.data, end=" -> ")
66             current = current.next
67         print("None")
68
69     # Contoh penggunaan
70     ll = LinkedList()
71     ll.tambahDepan(3)
72     ll.tambahDepan(2)
73     ll.tambahDepan(1)
74     ll.cetak()
75     ll.tambahAkhir(4)
76     ll.cetak()
77     ll.tambah(1.5, 1)
78     ll.cetak()
79     ll.hapus(1)
80     ll.cetak()
81     print(ll.cari(3))
82     print(ll.cari(5))

```

Picture 3.1 the code 3.py


```

PS D:\Semester 4\PrakA1_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakA1_and_StrDat/Module_3/Question/3.py"
1 -> 2 -> 3 -> None
1 -> 2 -> 3 -> 4 -> None
1 -> 1.5 -> 2 -> 3 -> 4 -> None
1 -> 2 -> 3 -> 4 -> None
True
False

```

Picture 3.2 the output 3.py

4. Regarding the doubly linked list, create a function for

- visit and print data for each node from the front and from the back.
- add a node at the beginning
- add a node at the end

```

Module_3 > Question > 4.py > DoublyLinkedList
1  class Node:
2      def __init__(self, data=None):
3          self.data = data
4          self.next = None
5          self.prev = None
6
7  class DoublyLinkedList:
8      def __init__(self):
9          self.head = None
10         self.tail = None
11
12     def cetakDepan(self):
13         current = self.head
14         while current:
15             print(current.data, end=" <-> ")
16             current = current.next
17         print("None")
18
19     def cetakBelakang(self):
20         current = self.tail
21         while current:
22             print(current.data, end=" <-> ")
23             current = current.prev
24         print("None")
25
26     def tambahDepan(self, data):
27         new_node = Node(data)
28         if self.head is None:
29             self.head = self.tail = new_node
30         else:
31             new_node.next = self.head
32             self.head.prev = new_node
33             self.head = new_node
34
35     def tambahAkhir(self, data):
36         new_node = Node(data)
37         if self.tail is None:
38             self.head = self.tail = new_node
39         else:
40             new_node.prev = self.tail
41             self.tail.next = new_node
42             self.tail = new_node

```

```

43
44     # Contoh penggunaan
45     dll = DoublyLinkedList()
46     dll.tambahDepan(3)
47     dll.tambahDepan(2)
48     dll.tambahDepan(1)
49     dll.cetakDepan()
50     dll.cetakBelakang()
51
52     dll.tambahAkhir(4)
53     dll.cetakDepan()
54     dll.cetakBelakang()
55

```

Picture 4.1 the code 4.py

```

PS D:\Semester 4\PrakA1_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakA1_and_StrDat/Module_3/Question/4.py"
1 <-> 2 <-> 3 <-> None
3 <-> 2 <-> 1 <-> None
1 <-> 2 <-> 3 <-> 4 <-> None
4 <-> 3 <-> 2 <-> 1 <-> None

```

Picture 4.2 the output 4.py