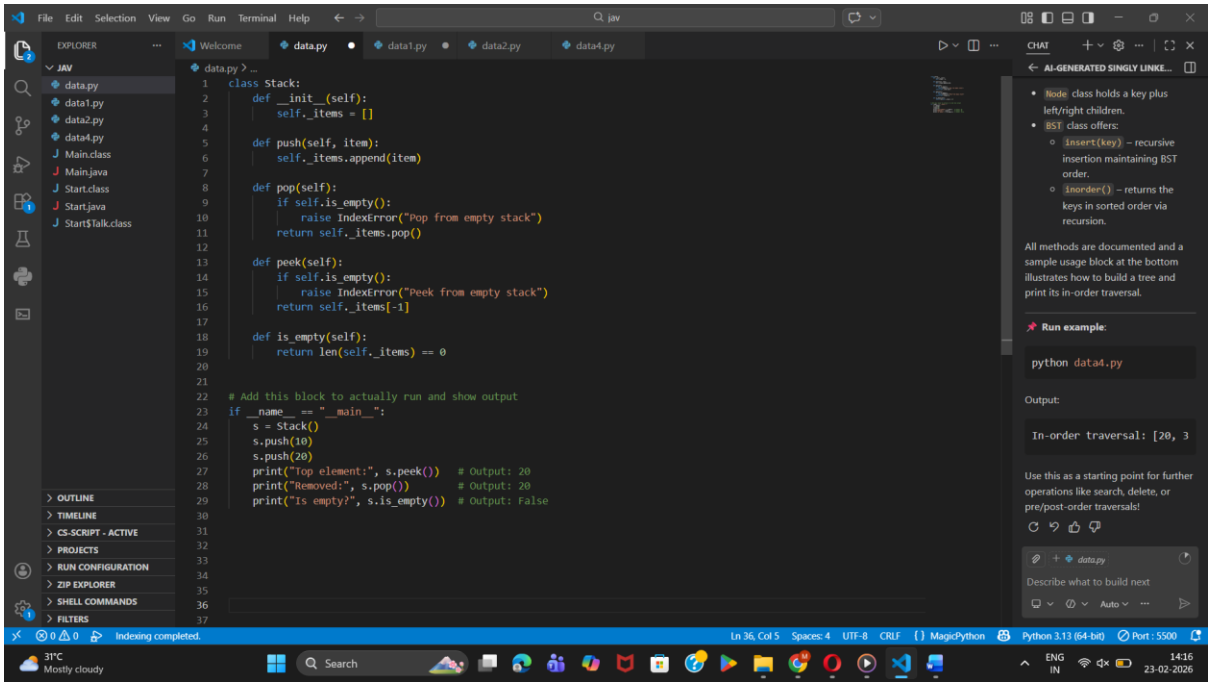
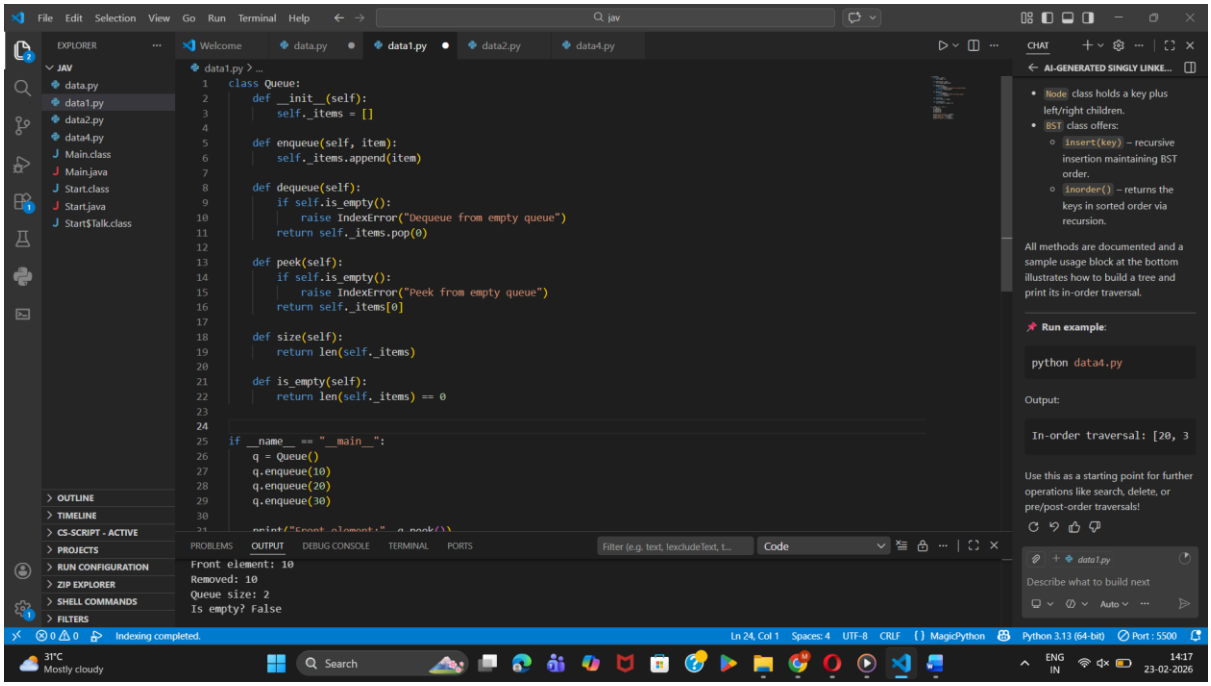
Assignment -11.1

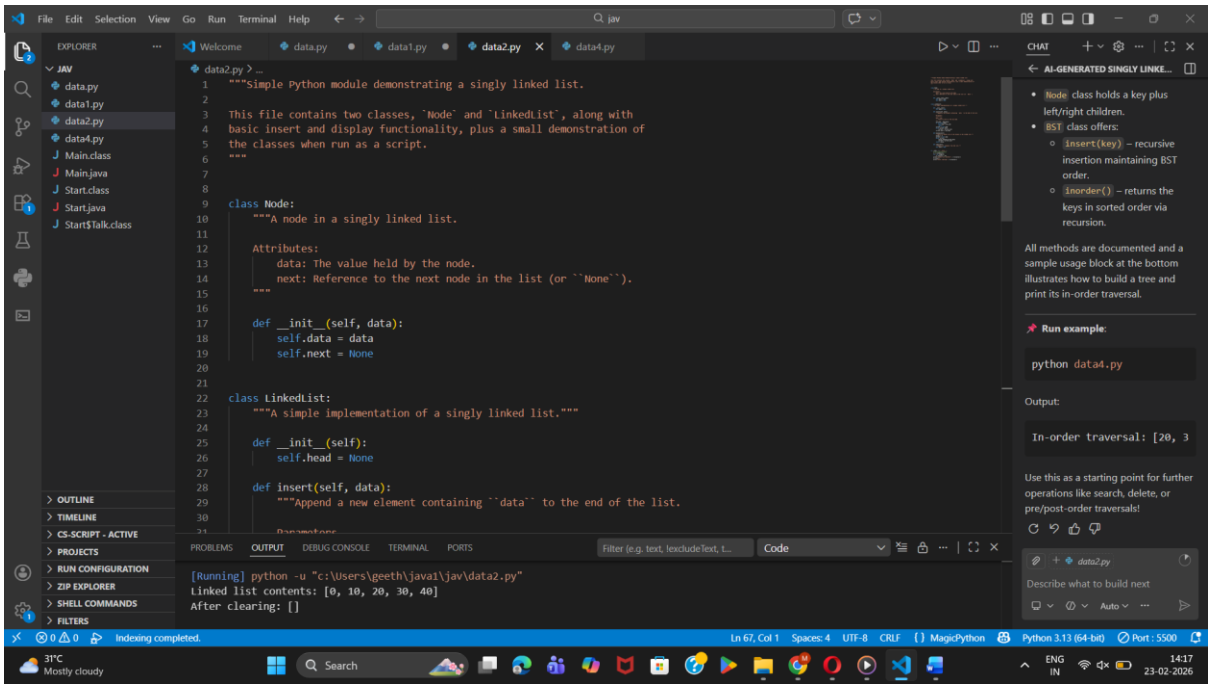
A.Harshita

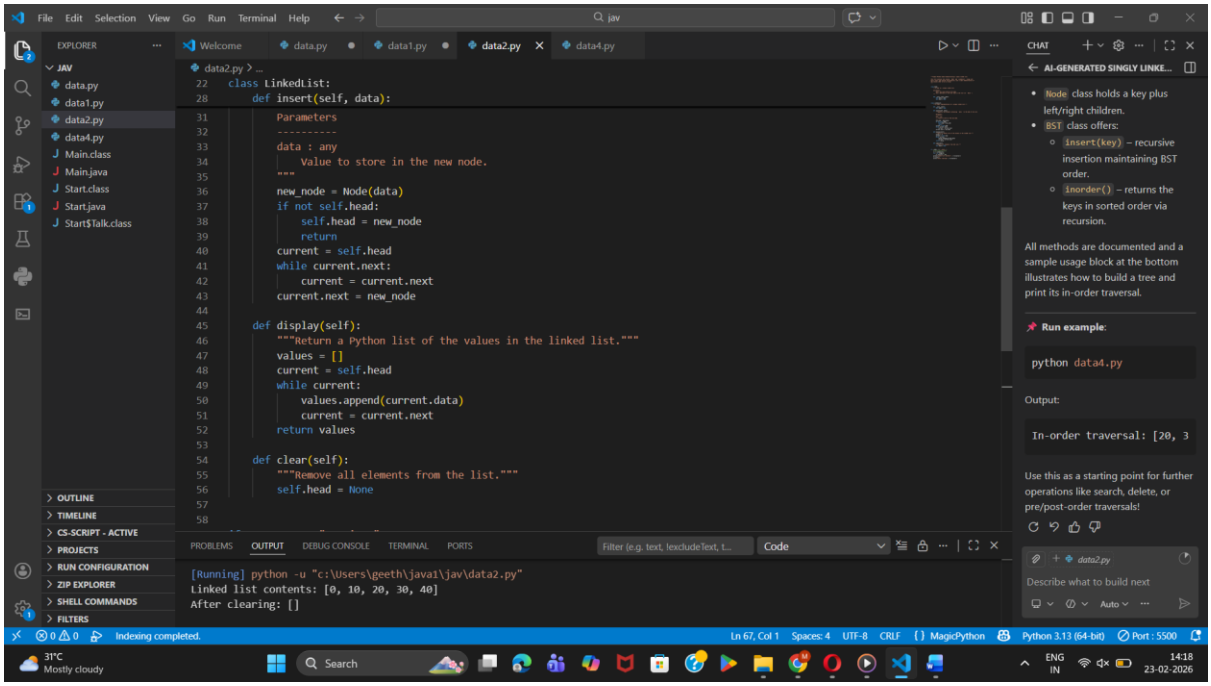
2303A52211

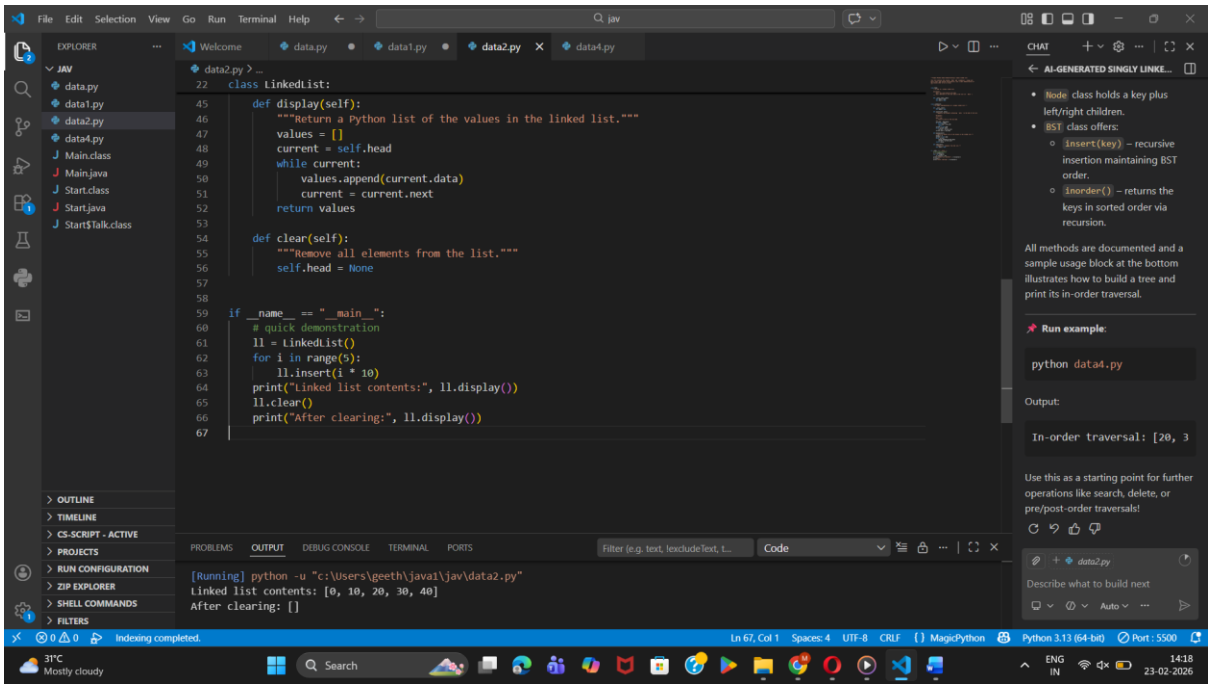
Batch-43

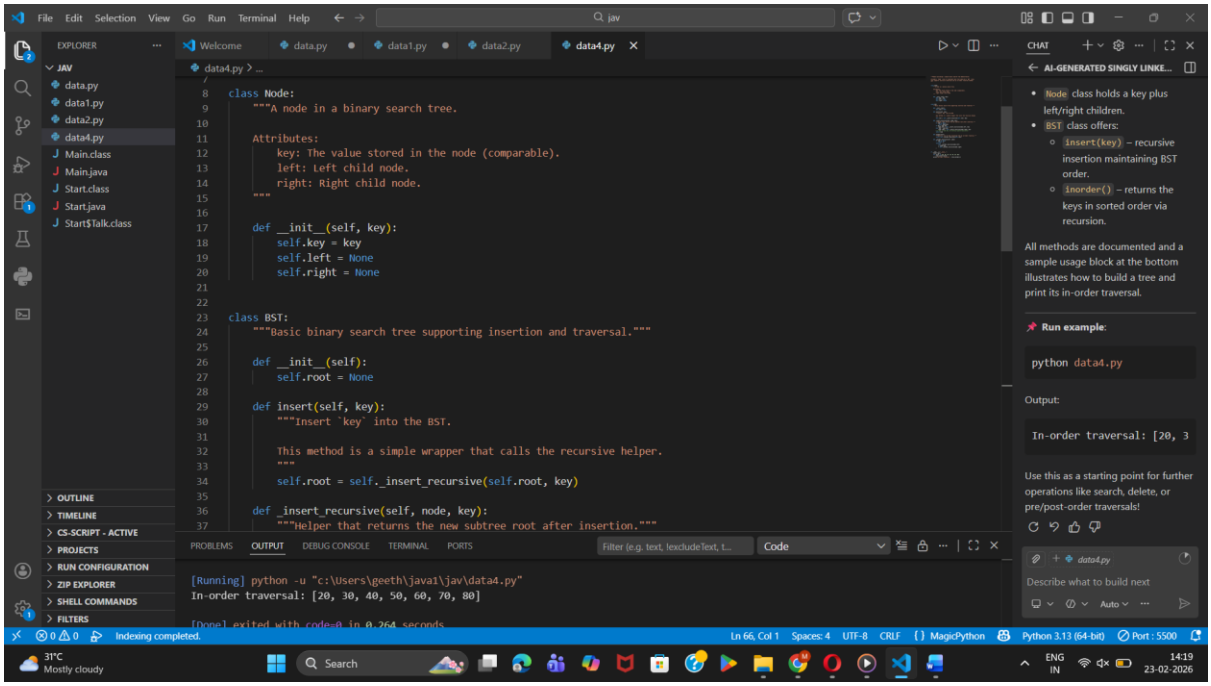


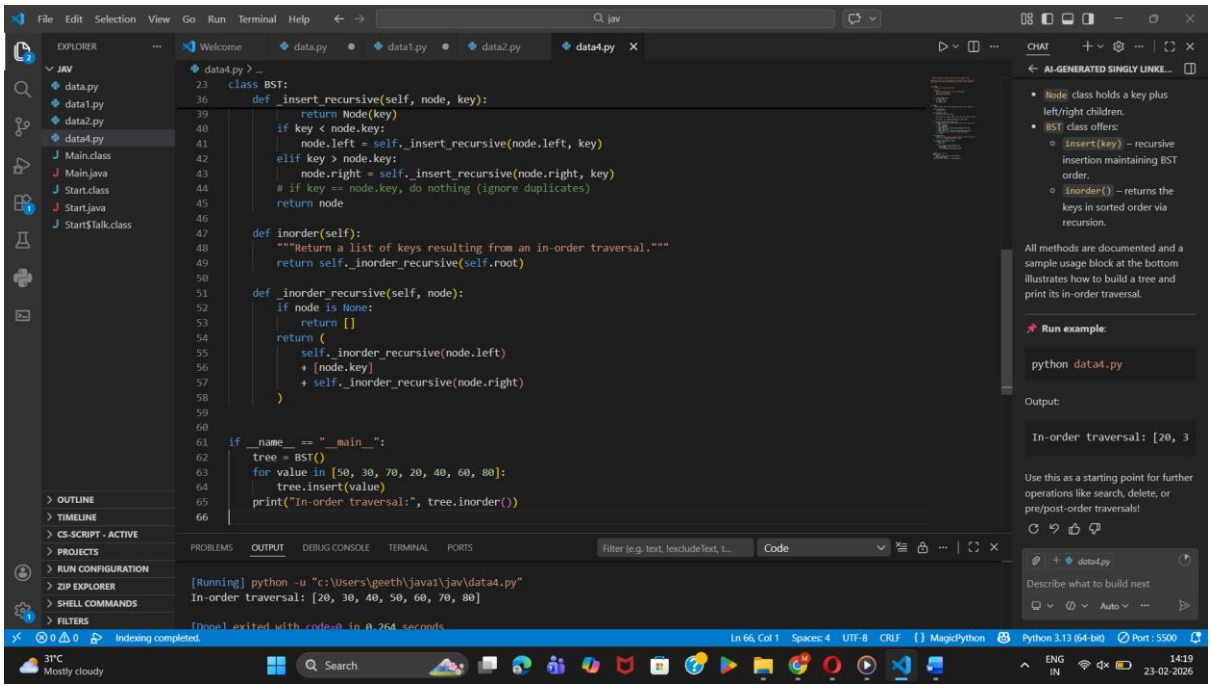

File Edit Selection View Go Run Terminal Help
jav
EXPLORER
JAV
data.py
data1.py
data2.py
data4.py
Main.class
Main.java
Start.class
Start.java
Start$Talk.class
Welcome
data.py
data1.py
data2.py
data4.py
data1.py > ...
class Queue:
def __init__(self):
self._items = []
def enqueue(self, item):
self._items.append(item)
def dequeue(self):
if self.is_empty():
raise IndexError("Dequeue from empty queue")
return self._items.pop(0)
def peek(self):
if self.is_empty():
raise IndexError("Peek from empty queue")
return self._items[0]
def size(self):
return len(self._items)
def is_empty(self):
return len(self._items) == 0
if __name__ == "__main__":
q = Queue()
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
PROJECTS
RUN CONFIGURATION
ZIP EXPLORER
SHELL COMMANDS
FILTERS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !excludeText, t...
Code
Front element: 10
Removed: 10
Queue size: 2
Is empty? False
CHAT
AI-GENERATED SINGLY LINKE...
Node class holds a key plus left/right children.
BST class offers:
insert(key) – recursive insertion maintaining BST order.
inorder() – returns the keys in sorted order via recursion.
All methods are documented and a sample usage block at the bottom illustrates how to build a tree and print its in-order traversal.
Run example:
python data4.py
Output:
In-order traversal: [20, 3
Use this as a starting point for further operations like search, delete, or pre/post-order traversals!
data1.py
Describe what to build next
Auto
Indexing completed.
Ln 24, Col 1
Spaces: 4
UTF-8
CRLF
MagicPython
Python 3.13 (64-bit)
Port : 5500
31°C
Mostly cloudy
Search
ENG
IN
14:17
23-02-2026

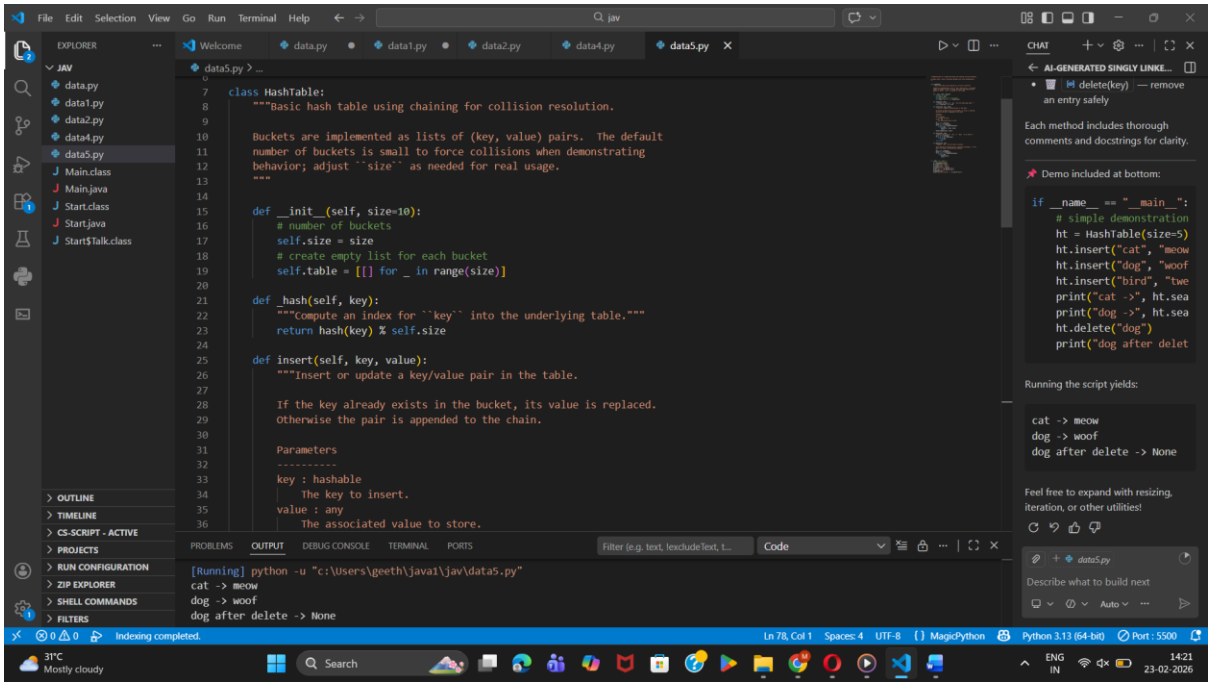

File Edit Selection View Go Run Terminal Help
jav
EXPLORER
JAV
data.py
data1.py
data2.py
data4.py
Main.class
Main.java
Start.class
Start.java
Start$Talk.class
Welcome
data.py
data1.py
data2.py
data4.py
data2.py > ...
"""Simple Python module demonstrating a singly linked list.
This file contains two classes, `Node` and `LinkedList`, along with
basic insert and display functionality, plus a small demonstration of
the classes when run as a script.
"""
class Node:
"""A node in a singly linked list.
Attributes:
data: The value held by the node.
next: Reference to the next node in the list (or ``None``).
"""
def __init__(self, data):
self.data = data
self.next = None
class LinkedList:
"""A simple implementation of a singly linked list."""
def __init__(self):
self.head = None
def insert(self, data):
"""Append a new element containing ``data`` to the end of the list.
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
PROJECTS
RUN CONFIGURATION
ZIP EXPLORER
SHELL COMMANDS
FILTERS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !excludeText, t...
Code
[Running] python -u "c:\Users\geeth\java1\jav\data2.py"
Linked list contents: [0, 10, 20, 30, 40]
After clearing: []
CHAT
AI-GENERATED SINGLY LINKE...
Node class holds a key plus left/right children.
BST class offers:
insert(key) – recursive insertion maintaining BST order.
inorder() – returns the keys in sorted order via recursion.
All methods are documented and a sample usage block at the bottom illustrates how to build a tree and print its in-order traversal.
Run example:
python data4.py
Output:
In-order traversal: [20, 3
Use this as a starting point for further operations like search, delete, or pre/post-order traversals!
data2.py
Describe what to build next
Auto
Indexing completed.
Ln 67, Col 1
Spaces: 4
UTF-8
CRLF
MagicPython
Python 3.13 (64-bit)
Port : 5500
31°C
Mostly cloudy
Search
ENG
IN
14:17
23-02-2026

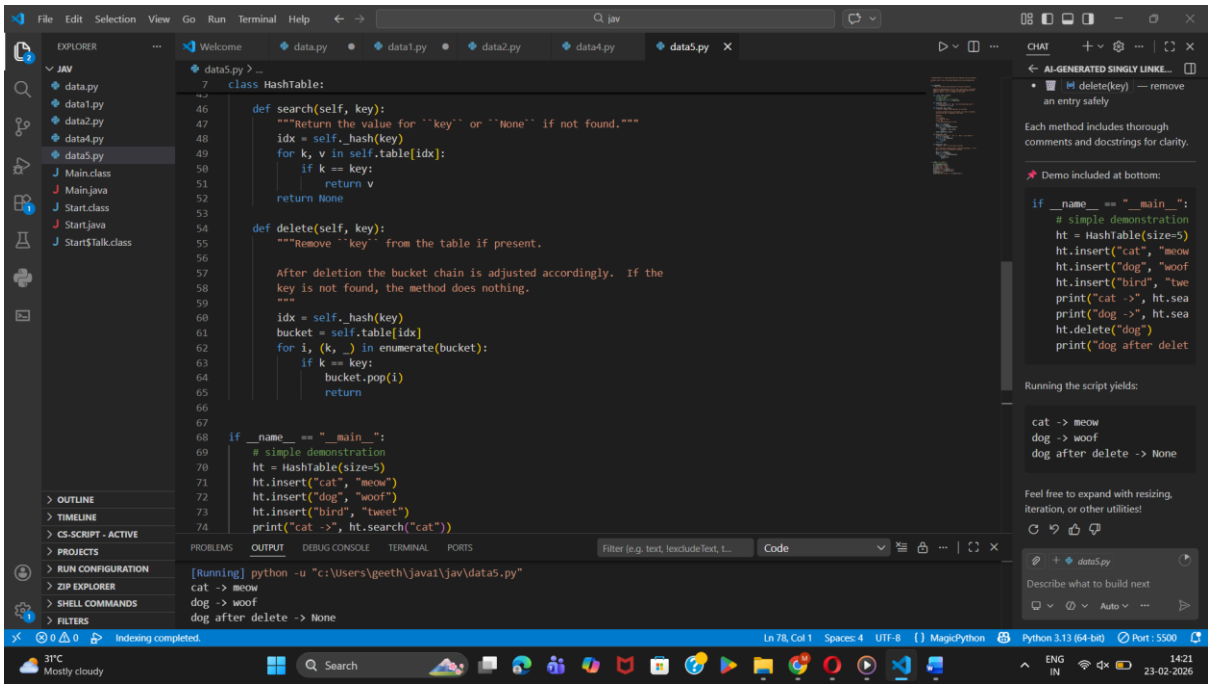

File Edit Selection View Go Run Terminal Help
jav
EXPLORER
JAV
data.py
data1.py
data2.py
data4.py
Main.class
Main.java
Start.class
Start.java
Start$Talk.class
Welcome
data.py
data1.py
data2.py
data4.py
data2.py > ...
class LinkedList:
def insert(self, data):
Parameters
data : any
Value to store in the new node.
new_node = Node(data)
if not self.head:
self.head = new_node
return
current = self.head
while current.next:
current = current.next
current.next = new_node
def display(self):
"""Return a Python list of the values in the linked list."""
values = []
current = self.head
while current:
values.append(current.data)
current = current.next
return values
def clear(self):
"""Remove all elements from the list."""
self.head = None
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
PROJECTS
RUN CONFIGURATION
ZIP EXPLORER
SHELL COMMANDS
FILTERS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !excludeText, t...
Code
[Running] python -u "c:\Users\geeth\java1\jav\data2.py"
Linked list contents: [0, 10, 20, 30, 40]
After clearing: []
CHAT
AI-GENERATED SINGLY LINKE...
Node class holds a key plus left/right children.
BST class offers:
insert(key) – recursive insertion maintaining BST order.
inorder() – returns the keys in sorted order via recursion.
All methods are documented and a sample usage block at the bottom illustrates how to build a tree and print its in-order traversal.
Run example:
python data4.py
Output:
In-order traversal: [20, 3
Use this as a starting point for further operations like search, delete, or pre/post-order traversals!
data2.py
Describe what to build next
Auto
Indexing completed.
Ln 67, Col 1
Spaces: 4
UTF-8
CRLF
MagicPython
Python 3.13 (64-bit)
Port : 5500
31°C
Mostly cloudy
Search
ENG
IN
14:18
23-02-2026

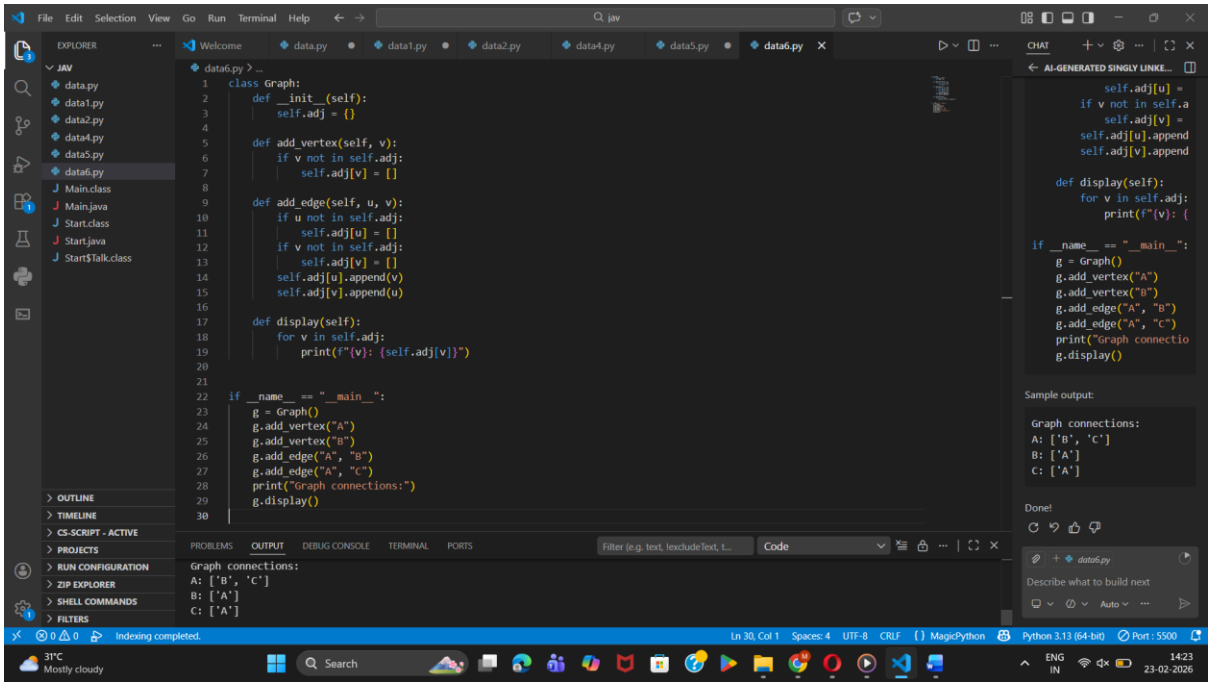

File Edit Selection View Go Run Terminal Help
jav
EXPLORER
JAV
data.py
data1.py
data2.py
data4.py
Main.class
Main.java
Start.class
Start.java
Start$Talk.class
Welcome
data.py
data1.py
data2.py
data4.py
data2.py > ...
class LinkedList:
def display(self):
"""Return a Python list of the values in the linked list."""
values = []
current = self.head
while current:
values.append(current.data)
current = current.next
return values
def clear(self):
"""Remove all elements from the list."""
self.head = None
if __name__ == "__main__":
# quick demonstration
ll = LinkedList()
for i in range(5):
ll.insert(i * 10)
print("Linked list contents:", ll.display())
ll.clear()
print("After clearing:", ll.display())
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
PROJECTS
RUN CONFIGURATION
ZIP EXPLORER
SHELL COMMANDS
FILTERS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !excludeText, t...
Code
[Running] python -u "c:\Users\geeth\java1\jav\data2.py"
Linked list contents: [0, 10, 20, 30, 40]
After clearing: []
CHAT
AI-GENERATED SINGLY LINKE...
Node class holds a key plus left/right children.
BST class offers:
insert(key) – recursive insertion maintaining BST order.
inorder() – returns the keys in sorted order via recursion.
All methods are documented and a sample usage block at the bottom illustrates how to build a tree and print its in-order traversal.
Run example:
python data4.py
Output:
In-order traversal: [20, 3
Use this as a starting point for further operations like search, delete, or pre/post-order traversals!
data2.py
Describe what to build next
Auto
Indexing completed.
Ln 67, Col 1
Spaces: 4
UTF-8
CRLF
MagicPython
Python 3.13 (64-bit)
Port : 5500
31°C
Mostly cloudy
Search
ENG
IN
14:18
23-02-2026

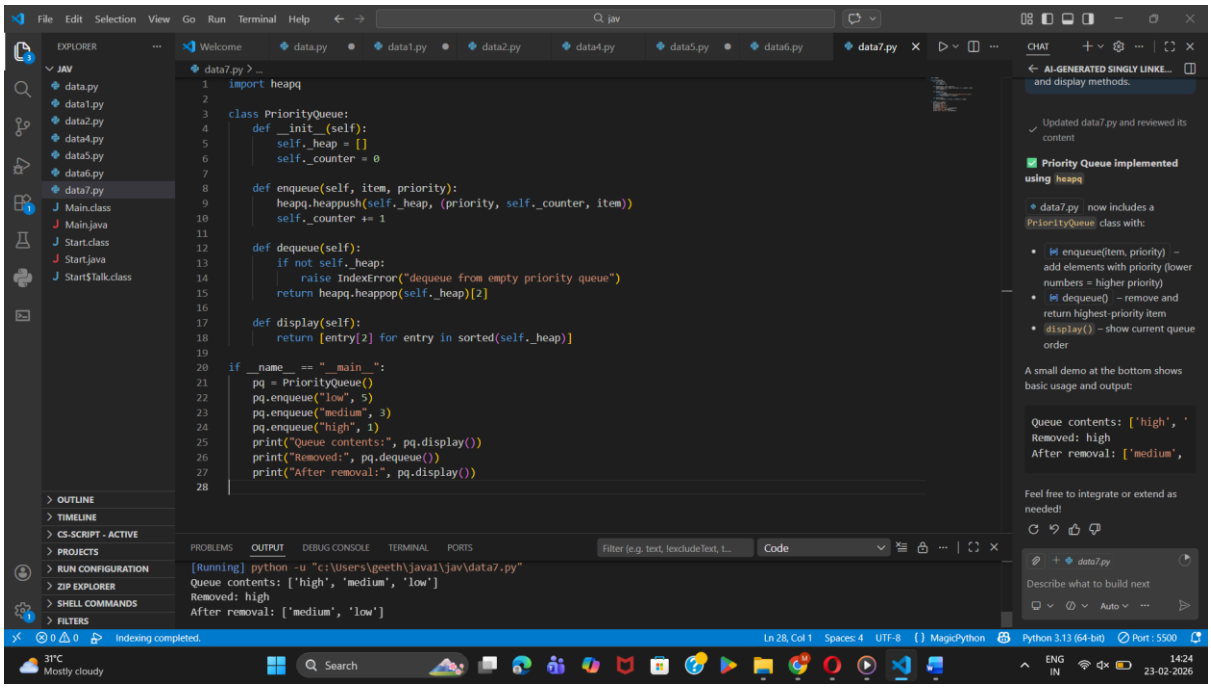

File Edit Selection View Go Run Terminal Help
jav
EXPLORER
JAV
data.py
data1.py
data2.py
data4.py
Main.class
Main.java
Start.class
Start.java
Start$Talk.class
Welcome
data.py
data1.py
data2.py
data4.py
data4.py > ...
class Node:
"""A node in a binary search tree.
Attributes:
key: The value stored in the node (comparable).
left: Left child node.
right: Right child node.
"""
def __init__(self, key):
self.key = key
self.left = None
self.right = None
class BST:
"""Basic binary search tree supporting insertion and traversal."""
def __init__(self):
self.root = None
def insert(self, key):
"""Insert `key` into the BST.
This method is a simple wrapper that calls the recursive helper.
"""
self.root = self._insert_recursive(self.root, key)
def _insert_recursive(self, node, key):
"""Helper that returns the new subtree root after insertion."""
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
PROJECTS
RUN CONFIGURATION
ZIP EXPLORER
SHELL COMMANDS
FILTERS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !excludeText, t...
Code
[Running] python -u "c:\Users\geeth\java1\jav\data4.py"
In-order traversal: [20, 30, 40, 50, 60, 70, 80]
[Done] exited with code=0 in 0.264 seconds
CHAT
AI-GENERATED SINGLY LINKE...
Node class holds a key plus left/right children.
BST class offers:
insert(key) – recursive insertion maintaining BST order.
inorder() – returns the keys in sorted order via recursion.
All methods are documented and a sample usage block at the bottom illustrates how to build a tree and print its in-order traversal.
Run example:
python data4.py
Output:
In-order traversal: [20, 3
Use this as a starting point for further operations like search, delete, or pre/post-order traversals!
data4.py
Describe what to build next
Auto
Indexing completed.
Ln 66, Col 1
Spaces: 4
UTF-8
CRLF
MagicPython
Python 3.13 (64-bit)
Port : 5500
31°C
Mostly cloudy
Search
ENG
IN
14:19
23-02-2026

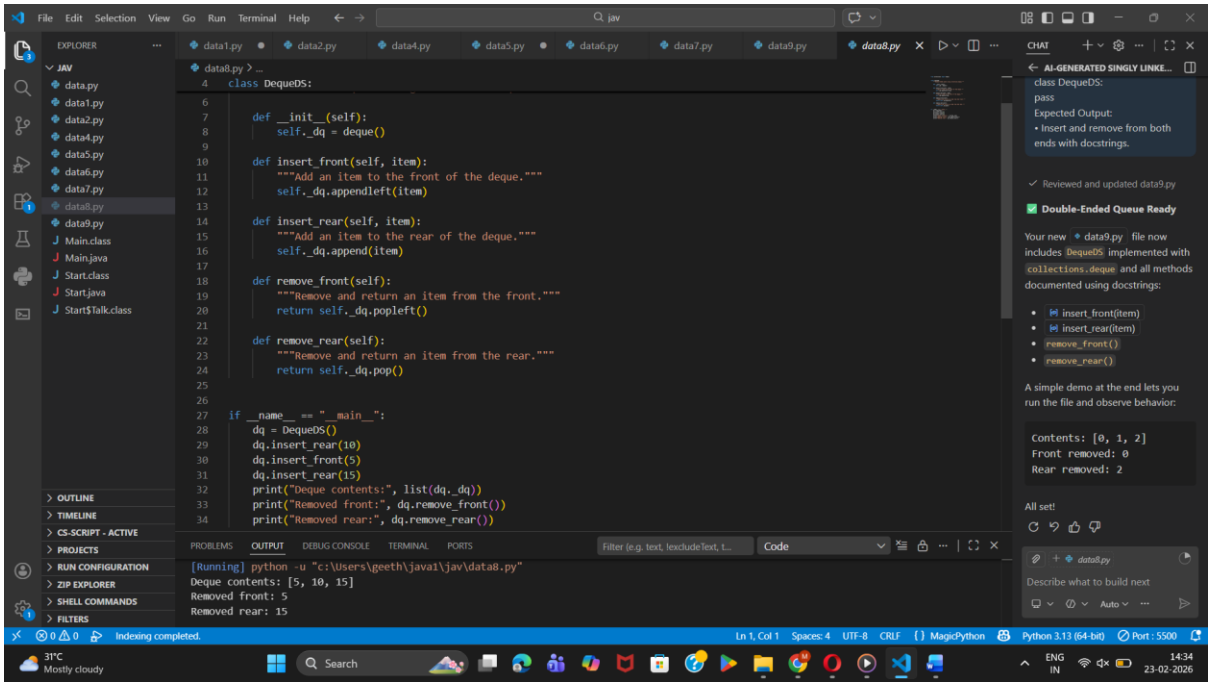

File Edit Selection View Go Run Terminal Help
jav
EXPLORER
JAV
data.py
data1.py
data2.py
data4.py
Main.class
Main.java
Start.class
Start.java
Start$Talk.class
Welcome
data.py
data1.py
data2.py
data4.py
data4.py > ...
class BST:
def _insert_recursive(self, node, key):
return Node(key)
if key < node.key:
node.left = self._insert_recursive(node.left, key)
elif key > node.key:
node.right = self._insert_recursive(node.right, key)
# if key == node.key, do nothing (ignore duplicates)
return node
def inorder(self):
"""Return a list of keys resulting from an in-order traversal."""
return self._inorder_recursive(self.root)
def _inorder_recursive(self, node):
if node is None:
return []
return (
self._inorder_recursive(node.left)
+ [node.key]
+ self._inorder_recursive(node.right)
)
if __name__ == "__main__":
tree = BST()
for value in [50, 30, 70, 20, 40, 60, 80]:
tree.insert(value)
print("In-order traversal:", tree.inorder())
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
PROJECTS
RUN CONFIGURATION
ZIP EXPLORER
SHELL COMMANDS
FILTERS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !excludeText, t...
Code
[Running] python -u "c:\Users\geeth\java1\jav\data4.py"
In-order traversal: [20, 30, 40, 50, 60, 70, 80]
[Done] exited with code=0 in 0.264 seconds
CHAT
AI-GENERATED SINGLY LINKE...
Node class holds a key plus left/right children.
BST class offers:
insert(key) – recursive insertion maintaining BST order.
inorder() – returns the keys in sorted order via recursion.
All methods are documented and a sample usage block at the bottom illustrates how to build a tree and print its in-order traversal.
Run example:
python data4.py
Output:
In-order traversal: [20, 3
Use this as a starting point for further operations like search, delete, or pre/post-order traversals!
data4.py
Describe what to build next
Auto
Indexing completed.
Ln 66, Col 1
Spaces: 4
UTF-8
CRLF
MagicPython
Python 3.13 (64-bit)
Port : 5500
31°C
Mostly cloudy
Search
ENG
IN
14:19
23-02-2026

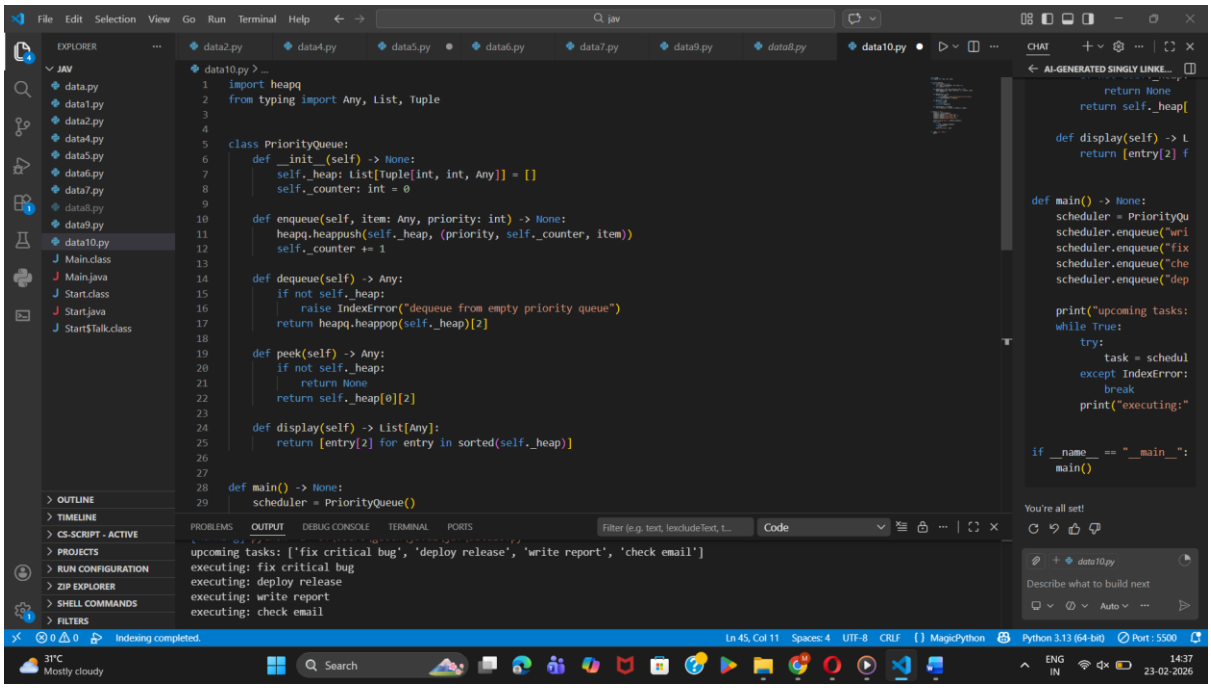

class HashTable:
"""Basic hash table using chaining for collision resolution.
Buckets are implemented as lists of (key, value) pairs. The default
number of buckets is small to force collisions when demonstrating
behavior; adjust ``size`` as needed for real usage.
"""
def __init__(self, size=10):
# number of buckets
self.size = size
# create empty list for each bucket
self.table = [[] for _ in range(size)]
def _hash(self, key):
"""Compute an index for ``key`` into the underlying table."""
return hash(key) % self.size
def insert(self, key, value):
"""Insert or update a key/value pair in the table.
If the key already exists in the bucket, its value is replaced.
Otherwise the pair is appended to the chain.
Parameters
key : hashable
The key to insert.
value : any
The associated value to store.
[Running] python -u "c:\Users\geeth\java1\jav\data5.py"
cat -> meow
dog -> woof
dog after delete -> None
Running the script yields:
cat -> meow
dog -> woof
dog after delete -> None
Feel free to expand with resizing, iteration, or other utilities!

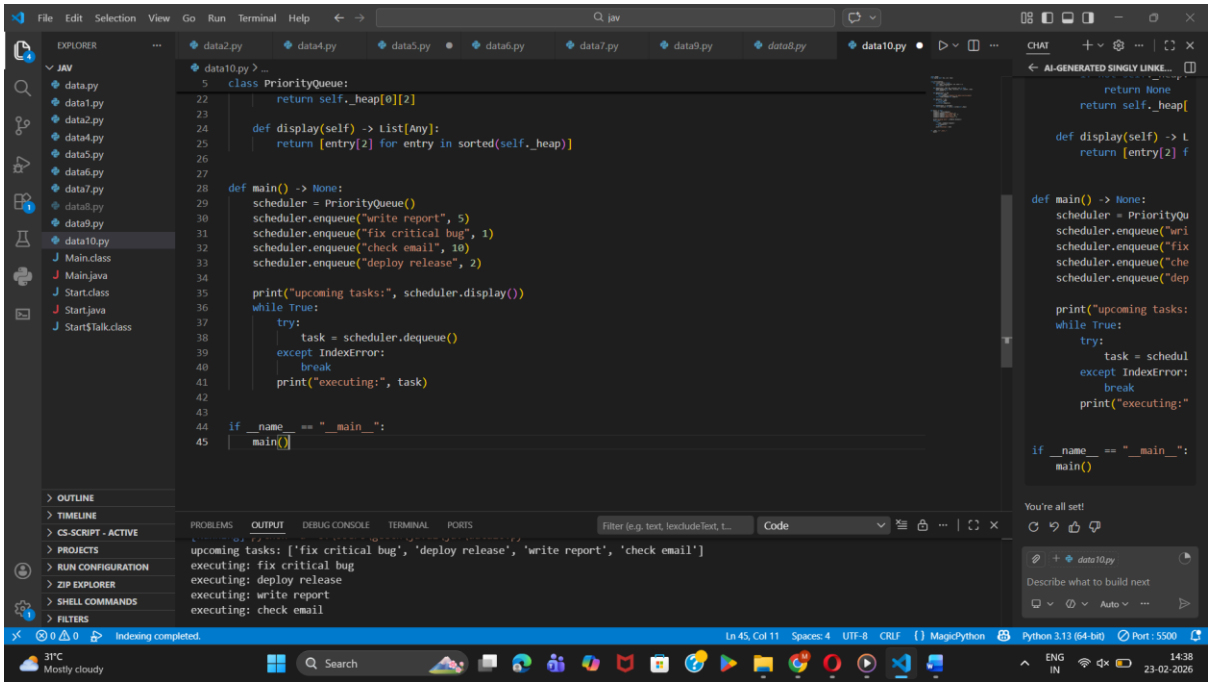

def search(self, key):
"""Return the value for ``key`` or ``None`` if not found."""
idx = self._hash(key)
for k, v in self.table[idx]:
if k == key:
return v
return None
def delete(self, key):
"""Remove ``key`` from the table if present.
After deletion the bucket chain is adjusted accordingly. If the
key is not found, the method does nothing.
"""
idx = self._hash(key)
bucket = self.table[idx]
for i, (k, _) in enumerate(bucket):
if k == key:
bucket.pop(i)
return
if __name__ == "__main__":
# simple demonstration
ht = HashTable(size=5)
ht.insert("cat", "meow")
ht.insert("dog", "woof")
ht.insert("bird", "tweet")
print("cat ->", ht.search("cat"))
[Running] python -u "c:\Users\geeth\java1\jav\data5.py"
cat -> meow
dog -> woof
dog after delete -> None
Running the script yields:
cat -> meow
dog -> woof
dog after delete -> None
Feel free to expand with resizing, iteration, or other utilities!


class Graph:
def __init__(self):
self.adj = {}
def add_vertex(self, v):
if v not in self.adj:
self.adj[v] = []
def add_edge(self, u, v):
if u not in self.adj:
self.adj[u] = []
if v not in self.adj:
self.adj[v] = []
self.adj[u].append(v)
self.adj[v].append(u)
def display(self):
for v in self.adj:
print(f"{v}: {self.adj[v]}")
if __name__ == "__main__":
g = Graph()
g.add_vertex("A")
g.add_vertex("B")
g.add_edge("A", "B")
g.add_edge("A", "C")
print("Graph connections:")
g.display()
Graph connections:
A: ['B', 'C']
B: ['A']
C: ['A']
Sample output:
Graph connections:
A: ['B', 'C']
B: ['A']
C: ['A']
Done!


import heapq
class PriorityQueue:
def __init__(self):
self._heap = []
self._counter = 0
def enqueue(self, item, priority):
heapq.heappush(self._heap, (priority, self._counter, item))
self._counter += 1
def dequeue(self):
if not self._heap:
raise IndexError("dequeue from empty priority queue")
return heapq.heappop(self._heap)[2]
def display(self):
return [entry[2] for entry in sorted(self._heap)]
if __name__ == "__main__":
pq = PriorityQueue()
pq.enqueue("low", 5)
pq.enqueue("medium", 3)
pq.enqueue("high", 1)
print("Queue contents:", pq.display())
print("Removed:", pq.dequeue())
print("After removal:", pq.display())
[Running] python -u "c:\Users\geeth\java1\jav\data7.py"
Queue contents: ['high', 'medium', 'low']
Removed: high
After removal: ['medium', 'low']
Priority Queue implemented using heapq
Sample output:
Queue contents: ['high', '
Removed: high
After removal: ['medium',
Done!


class DequeDS:
def __init__(self):
self._dq = deque()
def insert_front(self, item):
"""Add an item to the front of the deque."""
self._dq.appendleft(item)
def insert_rear(self, item):
"""Add an item to the rear of the deque."""
self._dq.append(item)
def remove_front(self):
"""Remove and return an item from the front."""
return self._dq.popleft()
def remove_rear(self):
"""Remove and return an item from the rear."""
return self._dq.pop()
if __name__ == "__main__":
dq = DequeDS()
dq.insert_rear(10)
dq.insert_front(5)
dq.insert_rear(15)
print("Deque contents:", list(dq._dq))
print("Removed front:", dq.remove_front())
print("Removed rear:", dq.remove_rear())
[Running] python -u "c:\Users\geeth\java1\jav\data8.py"
Deque contents: [5, 10, 15]
Removed front: 5
Removed rear: 15


import heapq
from typing import Any, List, Tuple
class PriorityQueue:
def __init__(self) -> None:
self._heap: List[Tuple[int, int, Any]] = []
self._counter: int = 0
def enqueue(self, item: Any, priority: int) -> None:
heapq.heappush(self._heap, (priority, self._counter, item))
self._counter += 1
def dequeue(self) -> Any:
if not self._heap:
raise IndexError("dequeue from empty priority queue")
return heapq.heappop(self._heap)[2]
def peek(self) -> Any:
if not self._heap:
return None
return self._heap[0][2]
def display(self) -> List[Any]:
return [entry[2] for entry in sorted(self._heap)]
def main() -> None:
scheduler = PriorityQueue()
upcoming tasks: ['fix critical bug', 'deploy release', 'write report', 'check email']
executing: fix critical bug
executing: deploy release
executing: write report
executing: check email


File Edit Selection View Go Run Terminal Help
jav
EXPLORER
JAV
data.py
data1.py
data2.py
data4.py
data5.py
data6.py
data7.py
data8.py
data9.py
data10.py
Main.class
Main.java
Start.class
Start.java
Start$Talk.class
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
PROJECTS
RUN CONFIGURATION
ZIP EXPLORER
SHELL COMMANDS
FILTERS
data2.py
data4.py
data5.py
data6.py
data7.py
data9.py
data8.py
data10.py
data10.py > ...
class PriorityQueue:
return self._heap[0][2]
def display(self) -> List[Any]:
return [entry[2] for entry in sorted(self._heap)]
def main() -> None:
scheduler = PriorityQueue()
scheduler.enqueue("write report", 5)
scheduler.enqueue("fix critical bug", 1)
scheduler.enqueue("check email", 10)
scheduler.enqueue("deploy release", 2)
print("upcoming tasks:", scheduler.display())
while True:
try:
task = scheduler.dequeue()
except IndexError:
break
print("executing:", task)
if __name__ == "__main__":
main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !excludeText, t...
Code
upcoming tasks: ['fix critical bug', 'deploy release', 'write report', 'check email']
executing: fix critical bug
executing: deploy release
executing: write report
executing: check email
CHAT
AI-GENERATED SINGLY LINKE...
You're all set!
data10.py
Describe what to build next
Auto
Indexing completed.
Ln 45, Col 11
Spaces: 4
UTF-8
CRLF
MagicPython
Python 3.13 (64-bit)
Port : 5500
31°C
Mostly cloudy
Search
ENG
IN
14:38
23-02-2026