numpy-part3

September 12, 2024

[1]:

```python
import numpy as np

# Array of names
names = np.array(['Alice', 'Bob', 'Charlie', 'David', 'Eve'])

# Filter names starting with 'D'
filtered_names = names[np.char.startswith(names, 'D')]
print("Names starting with 'D':", filtered_names)
```

```
Names starting with 'D': ['David']
```

[2]:

```python
import numpy as np

# Flat array of data
data = np.array(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'])

# Reshape into a 2x4 grid
reshaped_data = data.reshape(2, 4)
print("Reshaped Data (2x4 grid):\n", reshaped_data)
```

```
Reshaped Data (2x4 grid):
 [['A' 'B' 'C' 'D']
 ['E' 'F' 'G' 'H']]
```

[3]:

```python
import numpy as np

# Array with some missing data represented as NaN
data = np.array([1, 2, np.nan, 4, 5])

# Replace NaN with a specific value (e.g., 0)
cleaned_data = np.nan_to_num(data, nan=0)
print("Data with NaNs replaced by 0:", cleaned_data)
```

```
Data with NaNs replaced by 0: [1. 2. 0. 4. 5.]
```

[4]:

```python
import numpy as np

# Array of text data
```

```python
texts = np.array(['hello', 'world', 'numpy'])

# Convert all text to uppercase
uppercase_texts = np.char.upper(texts)
print("Uppercase Texts:", uppercase_texts)
```

```
Uppercase Texts: ['HELLO' 'WORLD' 'NUMPY']
```

[5]:

```python
import numpy as np

# Array of unsorted numbers
numbers = np.array([4, 1, 7, 3, 9])

# Sort the array
sorted_numbers = np.sort(numbers)
print("Sorted Numbers:", sorted_numbers)

# Array of unsorted names
names = np.array(['Charlie', 'Alice', 'Bob'])

# Sort the names alphabetically
sorted_names = np.sort(names)
print("Sorted Names:", sorted_names)
```

```
Sorted Numbers: [1 3 4 7 9]
Sorted Names: ['Alice' 'Bob' 'Charlie']
```

[6]:

```python
import numpy as np

# Create a 4x4 array
array = np.arange(16).reshape(4, 4)
print("Original Array:\n", array)

# Extract a 2x2 subarray from the top-left corner
subarray = array[:2, :2]
print("Top-Left 2x2 Subarray:\n", subarray)
```

```
Original Array:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
Top-Left 2x2 Subarray:
 [[0 1]
 [4 5]]
```

[2]:

```python
import numpy as np

# Arrays of different pieces of information
names = np.array(['Alice', 'Bob', 'Charlie'])
ages = np.array([30, 25, 35])

# Combine into a 2D array
combined_data = np.column_stack((names, ages))
print("Combined Data:\n", combined_data)
```

```
Combined Data:
 [['Alice' '30']
 ['Bob' '25']
 ['Charlie' '35']]
```

[8]:

```python
import numpy as np

# Array of strings
sentences = np.array(['hello world', 'world of numpy', 'numpy is great'])

# Replace 'world' with 'universe'
modified_sentences = np.char.replace(sentences, 'world', 'universe')
print("Modified Sentences:", modified_sentences)
```

```
Modified Sentences: ['hello universe' 'universe of numpy' 'numpy is great']
```

[9]:

```python
import numpy as np

# Create an array of temperatures in Celsius
celsius = np.array([0, 10, 20, 30])

# Convert Celsius to Fahrenheit using broadcasting
fahrenheit = celsius * 9/5 + 32
print("Temperatures in Fahrenheit:", fahrenheit)
```

```
Temperatures in Fahrenheit: [32. 50. 68. 86.]
```

[10]:

```python
import numpy as np

# Array of strings with different lengths
strings = np.array(['short', 'medium', 'a very long string'])

# Apply a function to get the length of each string
lengths = np.char.str_len(strings)
print("Lengths of Strings:", lengths)
```

```
Lengths of Strings: [ 5  6 18]
```

[11]:

```python
import numpy as np

# Create an array with some missing values
array = np.array([10, 20, np.nan, 40, 50])

# Replace missing values with the median of non-missing values
median_value = np.nanmedian(array)
array[np.isnan(array)] = median_value
print("Array with Missing Values Replaced by Median:", array)
```

```
Array with Missing Values Replaced by Median: [10. 20. 30. 40. 50.]
```

[1]:

```python
# Seating arrangement in a classroom (2D array)
import numpy as np

seating = np.array([['A1', 'A2', 'A3'],
                    ['B1', 'B2', 'B3'],
                    ['C1', 'C2', 'C3']])

# Accessing the seat in the second row, third column
seat = seating[1, 2]
print("Seat at second row, third column:", seat)
```

```
Seat at second row, third column: B3
```

[2]:

```python
# Survey responses (2D array)
import numpy as np

responses = np.array([['Yes', 'No', 'Yes'],
                      ['No', 'Yes', 'No'],
                      ['Yes', 'Yes', 'No']])

# Count the number of 'Yes' responses
yes_count = np.count_nonzero(responses == 'Yes')
print("Total 'Yes' responses:", yes_count)
```

```
Total 'Yes' responses: 5
```

[3]:

```python
# Library book titles (2D array)
import numpy as np

library = np.array([['Python Basics', 'Data Science'],
                    ['Machine Learning', 'Deep Learning']])

# Retrieve the title in the second row, first column
book_title = library[1, 0]
print("Book in second row, first column:", book_title)
```

```
Book in second row, first column: Machine Learning
```

[4]:

```python
# Student grades (2D array)
import numpy as np

grades = np.array([['Alice', 'B', 'A'],
                   ['Bob', 'C', 'B'],
                   ['Charlie', 'A', 'A']])

# Update a student's grade (Bob's second subject from 'B' to 'A')
grades[1, 2] = 'A'
print("Updated grades:\n", grades)
```

```
Updated grades:
 [['Alice' 'B' 'A']
 ['Bob' 'C' 'A']
 ['Charlie' 'A' 'A']]
```

[6]:

```python
# Inventory of items (2D array)
import numpy as np

inventory = np.array([['Item1', '10'],
                      ['Item2', '15'],
                      ['Item3', '8']])

# Update stock quantity for a specific item (increase Item2 by 5 units)
inventory[1, 1] = str(int(inventory[1, 1]) + 5)
print("Updated inventory:\n", inventory)
```

```
Updated inventory:
 [['Item1' '10']
 ['Item2' '20']
 ['Item3' '8']]
```

[7]:

```python
# Seating arrangement (2D array)
import numpy as np

seating = np.array([['A1', 'A2', 'A3'],
                    ['B1', 'B2', 'B3'],
                    ['C1', 'C2', 'C3']])

# Swap the seats of A1 and C3
seating[0, 0], seating[2, 2] = seating[2, 2], seating[0, 0]
print("Updated seating arrangement:\n", seating)
```

```
Updated seating arrangement:
 [['C3' 'A2' 'A3']
 ['B1' 'B2' 'B3']
 ['C1' 'C2' 'A1']]
```

[8]:

```python
# Survey responses (2D array)
import numpy as np

responses = np.array([['Yes', 'No', 'Yes'],
                      ['No', 'Yes', 'No'],
                      ['Yes', 'No', 'Yes']])

# Sort each row alphabetically
sorted_responses = np.sort(responses, axis=1)
print("Sorted responses:\n", sorted_responses)
```

```
Sorted responses:
 [['No' 'Yes' 'Yes']
 ['No' 'No' 'Yes']
 ['No' 'Yes' 'Yes']]
```

[9]:

```python
# Task assignment (2D array)
import numpy as np

tasks = np.array([['Task1', 'Alice'],
                  ['Task2', 'Bob'],
                  ['Task3', 'Charlie']])

# Reassign Task2 to a new employee (David)
tasks[1, 1] = 'David'
print("Updated task assignment:\n", tasks)
```

```
Updated task assignment:
 [['Task1' 'Alice']
 ['Task2' 'David']
 ['Task3' 'Charlie']]
```

[10]:

```python
# Stock availability (2D array)
import numpy as np

stock = np.array([['Product1', 'In Stock'],
                  ['Product2', 'Out of Stock'],
                  ['Product3', 'In Stock']])

# Find all products that are in stock
in_stock_products = stock[stock[:, 1] == 'In Stock']
print("Products in stock:\n", in_stock_products)
```

```
Products in stock:
 [['Product1' 'In Stock']
 ['Product3' 'In Stock']]
```

[11]:

```python
# Seat assignment (2D array)
import numpy as np

seats = np.array([['A1', 'A2', 'A3'],
                  ['B1', 'B2', 'B3'],
                  ['C1', 'C2', 'C3']])

# Shuffle the seat assignment randomly
np.random.shuffle(seats)
print("Shuffled seat assignment:\n", seats)
```

```
Shuffled seat assignment:
 [['C1' 'C2' 'C3']
 ['B1' 'B2' 'B3']
 ['A1' 'A2' 'A3']]
```

[12]:

```python
# Initial shift assignment for employees (2D array)
import numpy as np

shifts = np.array([['John', 'Day', 'Room1'],
                   ['Alice', 'Night', 'Room2'],
                   ['Bob', 'Evening', 'Room3']])

# Operations:
# 1. Swap shifts between Alice and Bob.
shifts[1, 1], shifts[2, 1] = shifts[2, 1], shifts[1, 1]

# 2. Assign a new room to John.
shifts[0, 2] = 'Room4'

# 3. Add a new employee with their shift (concatenate a new row).
new_employee = np.array([['David', 'Day', 'Room5']])
shifts = np.vstack([shifts, new_employee])

# 4. Sort the employees alphabetically by their names.
shifts = shifts[np.argsort(shifts[:, 0])]

# 5. Replace 'Evening' shifts with 'Afternoon' for all employees.
shifts[shifts[:, 1] == 'Evening', 1] = 'Afternoon'

print("Updated shift assignment:\n", shifts)
```

```
Updated shift assignment:
 [['Alice' 'Afterno' 'Room2']
 ['Bob' 'Night' 'Room3']
 ['David' 'Day' 'Room5']
 ['John' 'Day' 'Room4']]
```

[13]:

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

numbers = [4, 2, 9, 1, 5, 6]
bubble_sort(numbers)
print("Sorted list:", numbers)
```

```
Sorted list: [1, 2, 4, 5, 6, 9]
```

[17]:

```python
def add_inventory(inventory, item_name, quantity):
    """Add or update the quantity of an item in the inventory."""
    if item_name in inventory:
        inventory[item_name] += quantity
    else:
        inventory[item_name] = quantity

def remove_inventory(inventory, item_name, quantity):
    """Remove a quantity of an item from the inventory, ensuring no negative␣
 ↪values."""
    if item_name in inventory:
        inventory[item_name] = max(0, inventory[item_name] - quantity)

def check_inventory(inventory, item_name):
    """Check the current quantity of an item in the inventory."""
    return inventory.get(item_name, 0)

def list_inventory(inventory):
    """Return a sorted list of all items in the inventory."""
    return sorted(inventory.items())

def main():
    import sys
    input = sys.stdin.read
    data = input().strip().split('\n')

    inventory = {}
    n = int(data[0])

    results = []

    for i in range(1, n + 1):
        line = data[i].split()
        command = line[0]
        if command == "ADD":
            item_name = line[1]
            quantity = int(line[2])
            add_inventory(inventory, item_name, quantity)

        elif command == "REMOVE":
            item_name = line[1]
            quantity = int(line[2])
            remove_inventory(inventory, item_name, quantity)

        elif command == "CHECK":
            item_name = line[1]
            results.append(str(check_inventory(inventory, item_name)))

        elif command == "LIST":
            for item_name, quantity in list_inventory(inventory):
                results.append(f"{item_name} {quantity}")

    print("\n".join(results))

if __name__ == "__main__":
    main()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[17], line 56
     53     print("\n".join(results))
     55 if __name__ == "__main__":
---> 56     main()

Cell In[17], line 27, in main()
     24 data = input().strip().split('\n')
     26 inventory = {}
---> 27 n = int(data[0])
     29 results = []
     31 for i in range(1, n + 1):

ValueError: invalid literal for int() with base 10: ''
```

[26]:

```
def print_fibonacci_series(n: int) -> None:
    if n <= 0:
        raise ValueError("Input should be a positive integer.")

    # Edge cases
    if n == 1:
        print(0)
        return
    elif n == 2:
        print(0)
        print(1)
        return

    a, b = 0, 1
    print(a)  # Print the first Fibonacci number
    print(b)  # Print the second Fibonacci number

    for _ in range(n - 2):
        a, b = b, a + b
        print(b)  # Print the next Fibonacci number
print(print_fibonacci_series(10))
```

```
0
1
1
2
3
5
8
13
21
34
None
```

[27]:

```
def display_tasks(tasks):
    if not tasks:
        print("Your to-do list is empty.")
    else:
        print("Your To-Do List:")
        for i, task in enumerate(tasks, start=1):
            print(f"{i}. {task}")

def add_task(tasks):
    task = input("Enter the task: ").strip()
    if task:
        tasks.append(task)
        print(f"Task '{task}' added.")
    else:
        print("Task cannot be empty.")

def remove_task(tasks):
    display_tasks(tasks)
    try:
```

```python
        index = int(input("Enter the number of the task to remove: ")) - 1
        if 0 <= index < len(tasks):
            removed_task = tasks.pop(index)
            print(f"Task '{removed_task}' removed.")
        else:
            print("Invalid task number.")
    except ValueError:
        print("Invalid input. Please enter a number.")

def main():
    tasks = []

    while True:
        print("\nTo-Do List Application")
        print("1. View tasks")
        print("2. Add a task")
        print("3. Remove a task")
        print("4. Exit")

        choice = input("Choose an option: ").strip()

        if choice == '1':
            display_tasks(tasks)
        elif choice == '2':
            add_task(tasks)
        elif choice == '3':
            remove_task(tasks)
        elif choice == '4':
            print("Exiting the application.")
            break
        else:
            print("Invalid option. Please choose a valid option.")

if __name__ == "__main__":
    main()
```

```
To-Do List Application
1. View tasks
2. Add a task
3. Remove a task
4. Exit

Choose an option:  1

Your to-do list is empty.


To-Do List Application
```

```
1. View tasks
2. Add a task
3. Remove a task
4. Exit
Choose an option:  2
Enter the task:  work
Task 'work' added.

To-Do List Application
1. View tasks
2. Add a task
3. Remove a task
4. Exit
Choose an option:  1
Your To-Do List:
1. work

To-Do List Application
1. View tasks
2. Add a task
3. Remove a task
4. Exit
Choose an option:  2
Enter the task:  coding
Task 'coding' added.

To-Do List Application
1. View tasks
2. Add a task
3. Remove a task
4. Exit
Choose an option:  1
Your To-Do List:
1. work
2. coding

To-Do List Application
1. View tasks
2. Add a task
3. Remove a task
4. Exit
Choose an option:  4
Exiting the application.
```

[ ]:

```
```