Python

Advanced Functions & File Handling

Alish Bista - 11/11/2025

Advanced Functions

Lambda Functions

One Line Anonymous

- Anonymous function: No name needed, disposable
- Used for simple one-line operations, quick transformations
- Examples: Applying discount rates, tax calculations, unit conversions

```
# Traditional function
def double(x):
    return x * 2

# Lambda version one line
double = lambda x: x * 2

# Real use case: Quick calculations
prices = [10, 20, 30, 40]
discounted = list(map(lambda x: x * 0.9, prices))
print(discounted)  # [9.0, 18.0, 27.0, 36.0]
```

Map Filter Reduce

The Power Trio

- **Map**: Transform every item (add tax, convert currency)
- **Filter**: Select items meeting criteria (high performers, flagged transactions)
- **Reduce**: Aggregate/combine items into one result (totals, max, min)

```
from functools import reduce

sales = [100, 250, 80, 300, 150]

# MAP: Apply operation to each item
with_tax = list(map(lambda x: x * 1.08, sales))

# FILTER: Keep only items that match condition
high_value = list(filter(lambda x: x >= 200, sales))

# REDUCE: Combine all items into single value
total_sales = reduce(lambda x, y: x + y, sales)
print(total_sales)  # 880

max_sale = reduce(lambda x, y: x if x > y else y,
sales)
print(max_sale)  # 300
```

List Comprehension

Convenience in List Handling

- Format: [what_to_do for item in list if condition]
- Cleaner, faster, more readable

```
# Old way (what you might do first)
squared = []
for num in [1, 2, 3, 4, 5]:
    squared.append(num ** 2)


# List comprehension way (Pythonic)
squared = [num ** 2 for num in [1, 2, 3, 4, 5]]


# With condition
sales = [100, 250, 80, 300, 150]
big_sales = [x for x in sales if x >= 200]  # [250, 300]
```

File Handling

File Handling

Handling Data in Files

- Two ways to open files: *with* context managers, *without context managers*
- with statement: Python’s safety net
- Automatically closes files even if errors occur

```
# Old way (can cause problems)
file = open('report.txt', 'r')
content = file.read()
file.close()  # Easy to forget


# Modern way (Python handles cleanup)
with open('report.txt', 'r') as file:
    content = file.read()
    print(content)
```

File Modes

Read Write Append

- **Read** (‘r’): Safe, can’t break anything
- **Write** (‘w’): Destructive - deletes old content
- **Append** (‘a’): Adds to end - safe for logs

```
# 'r' = Read (default) - just viewing
with open('data.txt', 'r') as f:
    content = f.read()

# 'w' = Write - CAREFUL! Overwrites everything
with open('output.txt', 'w') as f:
    f.write('New content')

# 'a' = Append - adds to end
with open('log.txt', 'a') as f:
    f.write('New log entry\n')
```

Reading Files

Different Ways To Read

- Small file: Use .read()
- Large file: Loop line by line (memory efficient)
- Need all lines: Use .readlines()

```
# Method 1: Read entire file (small files)
with open('sales.txt', 'r') as f:
    content = f.read()  # One big string

# Method 2: Read line by line (cleaner)
with open('sales.txt', 'r') as f:
    for line in f:
        print(line.strip())  # strip()
removes \n

# Method 3: Read all lines into list
with open('sales.txt', 'r') as f:
    lines = f.readlines()  # List of
strings
```

Writing Files

Now We Write

- /n = new line (press Enter)
- f-strings = easy formatting

```
# Calculate something
products = ['Laptop', 'Mouse', 'Keyboard']
sales = [50000, 5000, 8000]

# Write summary report
with open('summary_report.txt', 'w') as f:
    f.write('Q4 Sales Summary\n')
    f.write('=' * 30 + '\n')

    for product, amount in zip(products,
sales):
        f.write(f'{product}: ${amount:,}\n')

    total = sum(sales)
    f.write(f'\nTotal: ${total:,}\n')
```

Practice

Practice

File Handling

sales_data.txt
Monday: 5400
Tuesday: 6200
Wednesday: 4800
Thursday: 7100
Friday: 8200

Read the file and calculate total weekly sales
Write a summary report to a new file

Practice

File Handling

```
def analyze_sales(filename):
    sales_data = {}

    with open(filename, 'r') as f:
        for line in f:
            day, amount = line.strip().split(': ')
            sales_data[day] = int(amount)

    total = sum(sales_data.values())

    # Write report
    with open('weekly_report.txt', 'w') as f:
        f.write(f'Total Sales: ${total:,}\n')
    return total

analyze_sales('sales_data.txt')
```

Q&A