==
==========
APL BOOK - COMPLETE SOLUTIONS AND ANSWERS MASTER FILE
==
==========

This file contains all solutions and answers organized by chapter.
Format: FILENAME | FILE CONTENT

==
==========
CHAPTER 1: FUNDAMENTALS - EXERCISES
==
==========

FILENAME: Chapter1_Fundamentals_Exercise2_Bytecode_Inspection.py
```
def square(x):
    return x * x

import dis
dis.dis(square)

def multiply(a, b):
    return a * b

dis.dis(multiply)
```

FILENAME: Chapter1_Fundamentals_Exercise3_Dynamic_Typing_in_Action.py
```
data = 42
print(type(data))

data = [1, 2, 3]
print(type(data))

def my_func():
    pass

data = my_func
print(type(data))
```

FILENAME: Chapter1_Fundamentals_Exercise5_AST_Exploration.py
```
import ast

code = "y = (4*5)-3"
```

```
tree = ast.parse(code)
print(ast.dump(tree, indent=4))
```

FILENAME: Chapter1_Fundamentals_Exercise6_Mutability_and_Object_Identity.py

```
my_list = [10, 20, 30]
print(f"Initial address: {id(my_list)}")

my_list.append(40)
print(f"After append: {id(my_list)}")
```

==

CHAPTER 2: DATA MODEL - EXERCISES

==

FILENAME:
Chapter2_DataModel_Problem1_Vector3D_Class_with_Operator_Overloading.py

```
class Vector3D:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def __add__(self, other):
        return Vector3D(self.x + other.x, self.y + other.y, self.z + other.z)

    def __sub__(self, other):
        return Vector3D(self.x - other.x, self.y - other.y, self.z - other.z)

    def __mul__(self, other):
        return self.x * other.x + self.y * other.y + self.z * other.z

    def __repr__(self):
        return f"Vector3D({self.x}, {self.y}, {self.z})"

v1 = Vector3D(1, 2, 3)
v2 = Vector3D(4, 5, 6)
print(v1 + v2)
print(v1 - v2)
print(v1 * v2)
```

FILENAME: Chapter2_DataModel_Problem2_Positive_Number_Descriptor.py

```
class Positive:
    def __init__(self, name):
        self.name = name

    def __get__(self, obj, objtype=None):
        return obj.__dict__.get(self.name, 0)

    def __set__(self, obj, value):
        if value < 0:
            raise ValueError(f"{self.name} cannot be negative")
        obj.__dict__[self.name] = value

class BankAccount:
    balance = Positive('balance')

    def __init__(self, initial_balance):
        self.balance = initial_balance

account = BankAccount(100)
print(account.balance)
account.balance = 50
print(account.balance)
```

FILENAME: Chapter2_DataModel_Problem3_Point_Class_with_slots.py

```
class Point:
    __slots__ = ['x', 'y']

    def __init__(self, x, y):
        self.x = x
        self.y = y

p = Point(1, 2)
print(f"x: {p.x}, y: {p.y}")

try:
    p.z = 5
except AttributeError as e:
    print(f"Error: {e}")
```

FILENAME: Chapter2_DataModel_Problem4_Disassembling_a_Simple_Function.py

```
import dis
```

```
def calculate_sum(a, b):
    return a + b

dis.dis(calculate_sum)
```

==
==========
CHAPTER 3: FUNCTIONAL PROGRAMMING - EXERCISES + CHAPTER 4: REGEX
==
==========

FILENAME: Chapter3_FunctionalProgramming_Exercise1_remove_vowels.py

```
def remove_vowels(text):
    vowels = "aeiouAEIOU"
    return ''.join([c for c in text if c not in vowels])

print(remove_vowels("Hello World"))
```

FILENAME: Chapter3_FunctionalProgramming_Exercise2_map_filter_squares.py

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
result = list(map(lambda x: x**2, filter(lambda x: x % 2 == 1, numbers)))
print(result)
```

FILENAME: Chapter3_FunctionalProgramming_Exercise3_fibonacci_with_memoization.py

```
from functools import lru_cache
import time

@lru_cache(maxsize=None)
def fib_memo(n):
    if n <= 1:
        return n
    return fib_memo(n-1) + fib_memo(n-2)

def fib_no_memo(n):
    if n <= 1:
        return n
    return fib_no_memo(n-1) + fib_no_memo(n-2)

start = time.time()
print(f"With memoization: {fib_memo(35)}")
print(f"Time: {time.time() - start}")
```

```
start = time.time()
print(f"Without memoization: {fib_no_memo(30)}")
print(f"Time: {time.time() - start}")
```

FILENAME: Chapter3_FunctionalProgramming_Exercise4_closure_make_adder.py

```
def make_adder(n):
    def adder(x):
        return n + x
    return adder

add_5 = make_adder(5)
print(add_5(10))
print(add_5(3))
```

FILENAME: Chapter3_FunctionalProgramming_Exercise5_apply_twice.py

```
def apply_twice(func, value):
    return func(func(value))

result = apply_twice(lambda x: x + 1, 5)
print(result)
```

FILENAME: Chapter3_FunctionalProgramming_Exercise6_functional_ETL_pipeline.py

```
from collections import Counter
import re

def functional_etl(texts):
    stopwords = {"the", "a", "is", "and", "or", "in", "on", "at", "to", "for"}

    words = []
    for text in texts:
        words.extend(re.findall(r'\b\w+\b', text.lower()))

    filtered = [w for w in words if w not in stopwords]
    return dict(Counter(filtered))

texts = ["the quick brown fox", "the lazy dog", "a quick fox"]
result = functional_etl(texts)
print(result)
```

FILENAME: Chapter3_FunctionalProgramming_Exercise7_custom_reduce.py

```
from functools import reduce as functools_reduce

def reduce(func, iterable, initial=None):
    iterator = iter(iterable)
    if initial is None:
        try:
            accumulator = next(iterator)
        except StopIteration:
            raise TypeError("reduce() of empty sequence with no initial value")
    else:
        accumulator = initial

    for value in iterator:
        accumulator = func(accumulator, value)

    return accumulator

result = reduce(lambda x, y: x + y, [1, 2, 3, 4, 5])
print(result)
```

FILENAME: Chapter3_FunctionalProgramming_Exercise8_decorator_log_call.py

```
def log_call(func):
    def wrapper(*args, **kwargs):
        print(f"Calling {func.__name__} with args={args}, kwargs={kwargs}")
        result = func(*args, **kwargs)
        print(f"{func.__name__} returned {result}")
        return result
    return wrapper

@log_call
def add(a, b):
    return a + b

add(3, 5)
```

FILENAME: Chapter3_Regex_Problem1_validate_email.py

```
import re
emails = ["user@example.com", "bad-email", "test@domain.org"]
pattern = re.compile(r'^[A-Za-z0-9._]+@[A-Za-z0-9.-]+\.(?:com|org|edu)$')
print([e for e in emails if pattern.match(e)])
```

```
FILENAME: Chapter3_Regex_Problem2_extract_hashtags.py
import re
text = "I love #Python and #AI"
print(re.findall(r'#\w+', text))

---

FILENAME: Chapter3_Regex_Problem3_validate_phone_numbers.py
import re
pattern = re.compile(r'^(?:\+\d-\d{3}-\d{4}|\d{3}-\d{3}-\d{4})$')
tests = ['+1-555-1234','123-456-7890','5551234']
print([t for t in tests if pattern.match(t)])

---

FILENAME: Chapter3_Regex_Problem4_word_frequency.py
import re
from collections import Counter
text = "Python, Python! AI is great; Python AI."
words = re.findall(r"\b[A-Za-z+#+\+]+\b", text)
print(dict(Counter(words)))

---

FILENAME: Chapter3_Regex_Problem5_find_duplicate_words.py
import re
text = "This is is a test test"
print(re.findall(r'\b(\w+)\s+\1\b', text))

---

FILENAME: Chapter3_Regex_Problem6_extract_dates.py
import re
text = "The events are on 2023-05-12 and 2024-01-01."
print(re.findall(r'\b\d{4}-\d{2}-\d{2}\b', text))

---

FILENAME: Chapter3_Regex_Problem7_mask_sensitive_data.py
import re
text = "Card: 1234-5678-9012-3456"
print(re.sub(r'\d(?=\d{4})', '*', text))

---

FILENAME: Chapter3_Regex_Problem8_extract_programming_languages.py
import re
text = "I know Python, Java, and C++ but not Ruby."
```

```
langs = re.findall(r'\bC\+\+\b|\b[A-Za-z]+\b', text)
print([l for l in langs if l.lower() in {'python','java','c++','ruby'}])
```

==

CHAPTER 5: OBJECT-ORIENTED PROGRAMMING

==

FILENAME: Chapter5_OOP_Problem1_Rectangle_Class.py

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

    def perimeter(self):
        return 2 * (self.width + self.height)

r = Rectangle(5, 10)
print(f"Area: {r.area()}")
print(f"Perimeter: {r.perimeter()}")
```

FILENAME: Chapter5_OOP_Problem2_Employee_Class_with_Alternative_Constructor.py

```
class Employee:
    def __init__(self, name, employee_id, salary):
        self.name = name
        self.employee_id = employee_id
        self.salary = salary

    @classmethod
    def from_string(cls, employee_str):
        parts = employee_str.split(',')
        return cls(parts[0], parts[1], float(parts[2]))

    def display_employee_info(self):
        print(f"Name: {self.name}, ID: {self.employee_id}, Salary: {self.salary}")

emp = Employee.from_string("John Doe,E123,50000")
emp.display_employee_info()
```

FILENAME: Chapter5_OOP_Problem3_Vehicle_Hierarchy.py

```
class Vehicle:
    def move(self):
        print("Vehicle is moving")

class Car(Vehicle):
    def move(self):
        print("Car is driving")

class Bike(Vehicle):
    def move(self):
        print("Bike is cycling")

vehicles = [Vehicle(), Car(), Bike()]
for v in vehicles:
    v.move()
```

FILENAME: Chapter5_OOP_Problem4_Vector_Class_with_Operator_Overloading.py

```
class Vector:
    def __init__(self, values):
        self.values = values

    def __sub__(self, other):
        return Vector([a - b for a, b in zip(self.values, other.values)])

    def __mul__(self, other):
        return sum(a * b for a, b in zip(self.values, other.values))

v1 = Vector([1, 2, 3])
v2 = Vector([4, 5, 6])
print(f"Subtraction: {(v1 - v2).values}")
print(f"Dot Product: {v1 * v2}")
```

FILENAME: Chapter5_OOP_Problem5_Shape_Polymorphism_Function.py

```
import math

class Shape:
    def area(self):
        return 0

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

def print_shape_area(shape):
    print(f"Area: {shape.area()}")

shapes = [Circle(5), Rectangle(4, 6)]
for s in shapes:
    print_shape_area(s)
```

==
==========

CHAPTER 6: DATA FORMATS (CSV/JSON/EXCEL)

==
==========

FILENAME: Chapter6_DataFormats_Problem1_CSV_Handling.py

```
import csv

with open('students.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerows([['ID', 'Name', 'Grade'], [1, 'Ali', 85], [2, 'Mona', 92], [3, 'Omar', 78]])

with open('students.csv', 'r') as f:
    reader = csv.DictReader(f)
    for row in reader:
        if int(row['Grade']) > 80:
            print(row['Name'])
```

FILENAME: Chapter6_DataFormats_Problem2_JSON_Handling.py

```
import json

data = {"course": "Python", "duration": "3 months", "students": ["Ali", "Sara"]}
```

```
with open('course.json', 'w') as f:
    json.dump(data, f)

with open('course.json', 'r') as f:
    loaded = json.load(f)
    print(loaded['students'])
```

FILENAME: Chapter6_DataFormats_Problem3_Excel_Handling.py

```
import pandas as pd

df = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie'], 'Salary': [50000, 60000, 70000]})
df.to_excel('employees.xlsx', index=False)

df_read = pd.read_excel('employees.xlsx')
print(df_read[['Name', 'Salary']])
```

FILENAME: Chapter6_DataFormats_Problem4_Data_Transformation.py

```
import csv
import json

def csv_to_json(csv_file, json_file):
    data = {"people": []}
    with open(csv_file, 'r') as f:
        reader = csv.DictReader(f)
        for row in reader:
            row['Age'] = int(row['Age'])
            data["people"].append(row)
    with open(json_file, 'w') as f:
        json.dump(data, f)

with open('data.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerows([['Name', 'Age', 'City'], ['Ali', 25, 'Cairo'], ['Mona', 30, 'Alex']])

csv_to_json('data.csv', 'data.json')
```

==

CHAPTER 7: DATABASES (SQLITE & SQLALCHEMY)

==

FILENAME: Chapter7_Databases_Problem1_Basic_SQLite_CRUD.py

```
import sqlite3

conn = sqlite3.connect('school.db')
c = conn.cursor()

c.execute('''CREATE TABLE students (id INTEGER PRIMARY KEY, name TEXT, grade
REAL)''')
c.execute("INSERT INTO students VALUES (1, 'Ali', 85.5)")
c.execute("INSERT INTO students VALUES (2, 'Sara', 92.0)")
c.execute("INSERT INTO students VALUES (3, 'Mohamed', 78.3)")

conn.commit()

c.execute("SELECT * FROM students")
for row in c.fetchall():
    print(row)

conn.close()
```

FILENAME: Chapter7_Databases_Problem2_Parameterized_Queries.py

```
import sqlite3

conn = sqlite3.connect('school.db')
c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS students (id INTEGER PRIMARY KEY, name
TEXT, grade REAL)''')
c.execute("INSERT INTO students VALUES (1, 'Ali', 85.5)")
c.execute("INSERT INTO students VALUES (2, 'Sara', 92.0)")
c.execute("INSERT INTO students VALUES (3, 'Mohamed', 78.3)")
conn.commit()

name = input("Enter name: ")
grade = float(input("Enter grade: "))
c.execute("INSERT INTO students (name, grade) VALUES (?, ?)", (name, grade))
conn.commit()

print("Updated Records")
c.execute("SELECT * FROM students")
for row in c.fetchall():
    print(row)
```

```python
conn.close()
```

FILENAME: Chapter7_Databases_Problem3_Transactions.py

```python
import sqlite3

conn = sqlite3.connect('school.db')
c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS students (id INTEGER PRIMARY KEY, name
TEXT, grade REAL)''')
c.execute("INSERT INTO students VALUES (1, 'Ali', 85.5)")
c.execute("INSERT INTO students VALUES (2, 'Sara', 92.0)")
c.execute("INSERT INTO students VALUES (3, 'Mohamed', 78.3)")
conn.commit()

try:
    c.execute("INSERT INTO students (name, grade) VALUES (?, ?)", ('Fatima', 88.0))
    c.execute("INSERT INTO students (name, grade) VALUES (?, ?)", ('Hassan', 90.0))
    x = 1 / 0
    conn.commit()
except Exception as e:
    print(f"Error occurred: {e}")
    conn.rollback()

print("Final Records:")
c.execute("SELECT * FROM students")
for row in c.fetchall():
    print(row)

conn.close()
```

FILENAME: Chapter7_Databases_Problem4_ORM_with_SQLAlchemy.py

```python
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

Base = declarative_base()

class Book(Base):
    __tablename__ = 'books'
    id = Column(Integer, primary_key=True)
    title = Column(String)
    author = Column(String)
```

```
engine = create_engine('sqlite:///library.db')
Base.metadata.create_all(engine)

Session = sessionmaker(bind=engine)
session = Session()

book1 = Book(title='Python Basics', author='Guido')
book2 = Book(title='AI with Python', author='Mohamed')
session.add(book1)
session.add(book2)
session.commit()

for book in session.query(Book).all():
    print(f"Book (id={book.id}, title='{book.title}', author='{book.author}')")

session.close()
```

==

CHAPTER 8: DATA SCIENCE (NUMPY/PANDAS/MATPLOTLIB/FLASK/PYTORCH)

==

FILENAME: Chapter8_DataScience_Problem1_NumPy_Operations.py

```
import numpy as np

arr = np.arange(1, 11)
print(f"Mean: {np.mean(arr)}")
print(f"Median: {np.median(arr)}")
print(f"Standard Deviation: {np.std(arr)}")
```

FILENAME: Chapter8_DataScience_Problem2_Pandas_Filtering.py

```
import pandas as pd

df = pd.DataFrame({'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 28], 'Score': [85, 75, 92]})
print(df[df['Score'] > 80])
```

FILENAME: Chapter8_DataScience_Problem3_Visualization_with_Matplotlib.py

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
```

```
y = [1, 4, 9, 16, 25]

plt.plot(x, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Line Graph')
plt.show()

---

FILENAME: Chapter8_DataScience_Problem4_Flask_Application.py
from flask import Flask

app = Flask(__name__)

@app.route('/hello')
def hello():
    return "Hello, Advanced Python!"

if __name__ == '__main__':
    app.run()

---

FILENAME: Chapter8_DataScience_Problem5_PyTorch_Tensor_Operations.py
import torch

t1 = torch.tensor([1, 2, 3])
t2 = torch.tensor([4, 5, 6])

dot_product = torch.dot(t1, t2)
element_wise = t1 * t2

print(f"Dot Product: {dot_product}")
print(f"Element-wise Multiplication: {element_wise}")

---


================================================================================
CHAPTER 9: WEB SCRAPING (REQUESTS/BEAUTIFULSOUP/SELENIUM)
================================================================================

FILENAME: Chapter9_WebScraping_Problem1_Fetch_Web_Page_Title.py
import requests
from bs4 import BeautifulSoup
```

```
response = requests.get('https://example.com')
soup = BeautifulSoup(response.text, 'html.parser')
title = soup.find('title')
print(f"Page Title: {title.text}")
```

FILENAME: Chapter9_WebScraping_Problem2_Extract_All_Links.py

```
import requests
from bs4 import BeautifulSoup

response = requests.get('https://example.com')
soup = BeautifulSoup(response.text, 'html.parser')
links = soup.find_all('a')
for link in links:
    if link.get('href'):
        print(link.get('href'))
```

FILENAME: Chapter9_WebScraping_Problem3_Extract_Table.py

```
from bs4 import BeautifulSoup

html = '''<table>
<tr><th>Name</th><th>Age</th></tr>
<tr><td>Alice</td><td>25</td></tr>
<tr><td>Bob</td><td>30</td></tr>
</table>'''

soup = BeautifulSoup(html, 'html.parser')
rows = soup.find_all('tr')
for row in rows:
    cells = row.find_all(['th', 'td'])
    print([cell.text for cell in cells])
```

FILENAME: Chapter9_WebScraping_Problem4_Automate_Google_Search.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

driver = webdriver.Chrome()
driver.get('https://www.google.com')
search_box = driver.find_element(By.NAME, 'q')
search_box.send_keys('Python Web Scraping')
search_box.send_keys(Keys.RETURN)
time.sleep(2)
print(driver.title)
driver.quit()
```

FILENAME: Chapter9_WebScraping_Problem5_Save_Scraped_Data_to_CSV.py

```
import csv
from bs4 import BeautifulSoup

html = '''<ul>
<li>Apple</li>
<li>Banana</li>
<li>Cherry</li>
</ul>'''

soup = BeautifulSoup(html, 'html.parser')
fruits = [li.text for li in soup.find_all('li')]

with open('fruits.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['Fruit'])
    for fruit in fruits:
        writer.writerow([fruit])
```

==

CHAPTER 10: DESIGN PATTERNS & ADVANCED

==

FILENAME: Chapter10_DesignPatterns_Problem1_Context_Manager.py

```
import time

class Timer:
    def __enter__(self):
        self.start = time.time()
        return self

    def __exit__(self, *args):
        self.end = time.time()
        print(f"Execution took {self.end - self.start:.2f} seconds")

with Timer():
    for i in range(1000000):
        pass
```

FILENAME: Chapter10_DesignPatterns_Problem2_Generator.py

```
def even_numbers(n):
    for i in range(2, n+1, 2):
        yield i

for num in even_numbers(10):
    print(num)
```

FILENAME: Chapter10_DesignPatterns_Problem3_Coroutine.py

```
def filter_positive():
    while True:
        value = yield
        if value > 0:
            print(f"Positive number: {value}")

co = filter_positive()
next(co)
co.send(-3)
co.send(5)
co.send(0)
```

FILENAME: Chapter10_DesignPatterns_Problem4_Factory_Pattern.py

```
class Shape:
    def draw(self):
        pass

class Circle(Shape):
    def draw(self):
        print("Drawing a Circle")

class Square(Shape):
    def draw(self):
        print("Drawing a Square")

def shape_factory(shape_type):
    if shape_type == "circle":
        return Circle()
    elif shape_type == "square":
        return Square()
```

```python
shape = shape_factory("circle")
shape.draw()
```

FILENAME: Chapter10_DesignPatterns_Problem5_Observer_Pattern.py

```python
class Subject:
    def __init__(self):
        self.observers = []

    def attach(self, observer):
        self.observers.append(observer)

    def notify(self, message):
        for obs in self.observers:
            obs.update(message)

class Observer:
    def update(self, message):
        print(f"Received update: {message}")

subject = Subject()
obs1, obs2 = Observer(), Observer()
subject.attach(obs1)
subject.attach(obs2)
subject.notify("Update available!")
```

==
==========
ALL CHAPTERS: MULTIPLE CHOICE QUESTIONS ANSWERS
==
==========

Chapter 1 (Fundamentals) MCQs:
(No MCQs in Chapter 1 - practical exercises only)

Chapter 2 (Data Model) MCQs:
(No MCQs in Chapter 2 - practical exercises only)

Chapter 3 (Functional Programming) MCQs:
9. C
10. B
11. C
12. C

13. B

Chapter 4 (Regex) MCQs:
1. A
2. B
3. B
4. D
5. B
6. C
7. B
8. B

Chapter 5 (OOP) MCQs:
(No MCQs in Chapter 5 - practical exercises only)

Chapter 6 (CSV/JSON/Excel) MCQs:
1. B
2. B
3. C
4. B
5. B

Chapter 7 (Databases) MCQs:
1. C
2. B
3. C
4. C
5. A

Chapter 8 (Data Science/Frameworks) MCQs:
1. C
2. C
3. B
4. A
5. C
6. A
7. A
8. C

Chapter 9 (Web Scraping) MCQs:
1. B
2. C
3. C
4. C
5. D
6. C
7. D

Chapter 10 (Design Patterns/Advanced) MCQs:
1. B
2. B
3. B
4. B
5. C

==
ALL CHAPTERS: TRUE/FALSE QUESTIONS ANSWERS
==

Chapter 3 (Regex) True/False:
1. re.match() checks for a pattern only at the start of the string.
2. True
3. \w+ matches letters, digits, and underscores.
4. True
5. True
6. True
7. re.findall() returns all non-overlapping matches in a list.
8. True

Chapter 6 (CSV/JSON/Excel) True/False:
1. The csv module reads values as strings and does not automatically convert numbers.
2. True
3. True
4. True
5. Writing Excel files with pandas typically requires an external engine like openpyxl.

Chapter 7 (Databases) True/False:
1. SQLite databases are usually stored in files and can also be in-memory.
2. True
3. True
4. True
5. cursor.execute() executes a statement; fetch methods return result sets.

Chapter 8 (Data Science/Frameworks) True/False:
1. NumPy arrays are more efficient than Python lists for numerical computations.
2. True
3. True
4. Flask is lighter and less opinionated than Django.
5. True
6. True

Chapter 9 (Web Scraping) True/False:
1. True
2. BeautifulSoup parses HTML but does not fetch pages; use requests to retrieve content.

3. True
4. True
5. Saving data into JSON format uses the json module, not the csv module.

Chapter 10 (Design Patterns/Advanced) True/False:
1. True
2. Generators yield values lazily one at a time using yield.
3. True
4. The Factory pattern abstracts object creation and does not notify observers.
5. True

==
END OF FILE
==