```python
# Summary Example 1: Comprehensive Code Using All Concepts (with Comments)
import json
from datetime import datetime, timedelta

# 1. String manipulation
name = "    Alice Smith    ".strip().title().replace("Smith", "Walker")  # Clean, title-case, and replace last name

# 2. Conditionals and nested if
score = 95
if score >= 90:
    if score >= 95:
        performance = "Excellent"  # Nested condition true
    else:
        performance = "Very Good"
else:
    performance = "Needs Improvement"

# 3. Looping with enumerate and zip
subjects = ["Math", "Science", "English"]
marks = [88, 92, 85]
grades = {subject: mark for subject, mark in zip(subjects, marks)}  # Combine two lists into a dictionary

# 4. List comprehension
even_squares = [x**2 for x in range(10) if x % 2 == 0]  # Square even numbers from 0 to 9

# 5. Dictionary comprehension
subject_lengths = {k: len(k) for k in subjects}  # Create a map of subject names to their length

# 6. Function with default and keyword args
def greet(person="Guest", lang="en"):
    return f"Hello, {person}" if lang == "en" else f"Bonjour, {person}"

# 7. Function returning multiple values
def stats(nums):
    return min(nums), max(nums), sum(nums)/len(nums)

# 8. Nested function (closure)
def multiplier(factor):
    def multiply(x):
        return x * factor
    return multiply

double = multiplier(2)

# 9. Class with inheritance
class Person:
    def speak(self):
        return "Hello"
```

```
class Student(Person):  # Inherit from Person
    def speak(self):
        return "I'm a student"

# 10. Generator
def countdown(n):
    while n > 0:
        yield n
        n -= 1

# 11. Set operations
a = {1, 2, 3}
b = {2, 3, 4}
union = a | b
intersection = a & b

# 12. JSON parsing
info_json = '{"city": "Paris", "country": "France"}'
info = json.loads(info_json)

# 13. Datetime formatting
next_week = (datetime.now() + timedelta(days=7)).strftime("%Y-%m-%d")

# 14. Sorting with lambda
sorted_grades = sorted(grades.items(), key=lambda x: x[1], reverse=True)

# 15. Recursion
def factorial(n):
    if n == 1:
        return 1
    return n * factorial(n - 1)

# Displaying all results
print(f"Greeting: {greet(name)}")
print(f"Performance: {performance}")
print(f"Grades: {grades}")
print(f"Even Squares: {even_squares}")
print(f"Subject Lengths: {subject_lengths}")
print(f"Stats: {stats(marks)}")
print(f"Double 5: {double(5)}")
print(f"Student says: {Student().speak()}")
print(f"Countdown from 3: {list(countdown(3))}")
print(f"Union: {union}, Intersection: {intersection}")
print(f"City: {info['city']}, Country: {info['country']}")
print(f"Next Week: {next_week}")
print(f"Sorted Grades: {sorted_grades}")
print(f"Factorial 5: {factorial(5)}")
"""
Greeting: Hello, Alice Walker
Performance: Excellent
Grades: {'Math': 88, 'Science': 92, 'English': 85}
Even Squares: [0, 4, 16, 36, 64]
```

```
Subject Lengths: {'Math': 4, 'Science': 7, 'English': 7}
Stats: (85, 92, 88.33333333333333)
Double 5: 10
Student says: I'm a student
Countdown from 3: [3, 2, 1]
Union: {1, 2, 3, 4}, Intersection: {2, 3}
City: Paris, Country: France
Next Week: 2025-07-02
Sorted Grades: [('Science', 92), ('Math', 88), ('English', 85)]
Factorial 5: 120
"""
# Notes Section
# --------------------

# Summary Example 2: File Processing & Statistics
# This code simulates reading lines from a file and calculating word stats

sample_lines = [
    "Python is powerful and easy to learn.",
    "It is widely used in data science.",
    "Python supports multiple paradigms."
]

word_freq = {}  # Dictionary to hold word frequency

# Process each line
for line in sample_lines:
    words = line.lower().replace('.', '').split()  # Normalize words
    for word in words:
        word_freq[word] = word_freq.get(word, 0) + 1  # Count words

# Sort by frequency (descending)
sorted_words = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)

# Show top 3 most frequent words
print("Top 3 words:")
for word, freq in sorted_words[:3]:
    print(f"{word}: {freq}")
"""
Top 3 words:
python: 2
is: 2
and: 1
"""
# Notes Section
# --------------------

# Summary Example 3: Simulated API & Data Analysis
# This code simulates data from an API and performs filtering & aggregation

data_from_api = [
    {"name": "Alice", "role": "developer", "hours": 38},
    {"name": "Bob", "role": "designer", "hours": 42},
    {"name": "Charlie", "role": "developer", "hours": 45},
    {"name": "Diana", "role": "manager", "hours": 50},
]

# Filter only developers
devs = [emp for emp in data_from_api if emp["role"] == "developer"]

# Calculate average hours for developers
avg_hours = sum(emp["hours"] for emp in devs) / len(devs)

# Create a name -> hours dictionary
dev_hours = {emp["name"]: emp["hours"] for emp in devs}

# Output the result
print("Developer Work Hours:")
for name, hours in dev_hours.items():
    print(f"{name}: {hours} hrs")

print(f"Average Developer Hours: {avg_hours:.1f} hrs")
"""
Developer Work Hours:
Alice: 38 hrs
Charlie: 45 hrs
Average Developer Hours: 41.5 hrs
"""
# Notes Section
# --------------------
```

# Summary Example 4: Email Validator

```
# Validate a list of emails using basic string checks
emails = ["user@example.com", "invalid@.com", "admin@site.org"]

# Use list comprehension to filter valid emails
# Check if '@' exists and '.' appears after '@'
valid_emails = [e for e in emails if "@" in e and "." in e.split("@")[-1]]

print("Valid Emails:", valid_emails)
"""
Valid Emails: ['user@example.com', 'admin@site.org']
"""
# Notes Section
# --------------------
```

# Summary Example 5: Quiz Auto Grader

```
# Grade a student quiz by comparing answers

answers = ["A", "C", "B", "D"]   # Correct answers
student = ["A", "C", "A", "D"]   # Student's answers

# Use zip to pair answers and count matches
score = sum([1 for a, b in zip(answers, student) if a == b])

print(f"Score: {score}/4")
```

```python
"""
Score: 3/4
"""
# Notes Section
# --------------------

# Summary Example 6: Grouping Students by Grade
# Group students based on their marks into letter grades

students = [("Amit", 90), ("Sara", 75), ("Leo", 60)]

# Initialize grade categories
groups = {"A": [], "B": [], "C": []}

# Classify students using if-elif logic
for name, marks in students:
    if marks >= 85:
        groups["A"].append(name)
    elif marks >= 70:
        groups["B"].append(name)
    else:
        groups["C"].append(name)

print(groups)
"""
{'A': ['Amit'], 'B': ['Sara'], 'C': ['Leo']}
"""
# Notes Section
# --------------------

# Summary Example 7: Word Count
# Count the frequency of each word in a paragraph

text = "python python java python c++ java java python"
words = text.split()  # Split text into words

# Use dictionary comprehension to count each unique word
word_map = {w: words.count(w) for w in set(words)}
print(word_map)
"""
{'java': 3, 'c++': 1, 'python': 4}
"""
# Notes Section
# --------------------

# Summary Example 8: Palindrome Detector
# Detect words that read the same forward and backward

words = ["madam", "apple", "level", "banana"]

# A word is a palindrome if it equals its reverse
palindromes = [w for w in words if w == w[::-1]]

print("Palindromes:", palindromes)
```

```python
"""
Palindromes: ['madam', 'level']
"""
# Notes Section
# --------------------

# Summary Example 9: FizzBuzz
# Classic FizzBuzz using list comprehension

# Replace numbers divisible by 3 with 'Fizz',
# by 5 with 'Buzz', and by both with 'FizzBuzz'
fizzbuzz = ["FizzBuzz" if i%15==0 else "Fizz" if i%3==0 else "Buzz" if i%5==0 else
str(i) for i in range(1, 16)]

print(" ".join(fizzbuzz))
"""
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz
"""
# Notes Section
# --------------------

# Summary Example 10: Factorial Table
# Print factorial of numbers from 1 to 5

def factorial(n):
    return 1 if n == 1 else n * factorial(n - 1)

# Loop through 1 to 5 and display factorials
for i in range(1, 6):
    print(f"{i}! = {factorial(i)}")
"""
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
"""
# Notes Section
# --------------------
```