Həftə 1 — Python Basics & Data Structures: Tapşırıqlar

Gün 1 — Introduction, Control Structures, Functions

Mövzular: Python syntax, variables, data types, if-else, loops (for, while), functions

Assignment 1.1 — Elektrik Tarif Kalkulyatoru

Bir ev təsərrüfatının aylıq elektrik istehlakını (kWh) alaraq, pilləli tarif sisteminə əsasən ödəniş məbləğini hesablayan proqram yazın.

Tarif cədvəli:

Pillə	Aralıq (kWh)	Qiymət (qəpik/kWh)
1	0 – 150	6
2	151 – 300	8
3	301 – 500	12
4	500+	18

Tələblər:

1. `calculate_bill(kwh: float) -> float` funksiyası yazın ki, istehlaka görə ümumi məbləği (AZN-lə) qaytarsın. Hər pillə müstəqil hesablanmalıdır (ilk 150 kWh birinci tarifə, sonrakı 150 kW ikinci tarifə və s.)
2. `get_tariff_breakdown(kwh: float) -> str` funksiyası yazın ki, hər pillədə nə qədər kWh istehlak edildiyini və onun qiymətini detallı şəkildə göstərsin
3. İstifadəçidən daxil edilən mənfi və ya sıfır dəyərləri rədd edin
4. Proqram istifadəçidən dəfələrlə istehlak dəyəri qəbul etsin və `q` daxil edildikdə dayansın

Nümunə çıxış:

```
İstehlak (kWh): 350

Pillə 1: 150 kWh × 6 qəp = 9.00 AZN
Pillə 2: 150 kWh × 8 qəp = 12.00 AZN
Pillə 3:  50 kWh × 12 qəp = 6.00 AZN

Ümumi: 27.00 AZN
```

Assignment 1.2 — Matris Əməliyyatları (Funksiyanlarla)

Daxil edilmiş iki matrisin üzərində əməliyyatlar aparan proqram yazın. Matrislər iç-içə `list` olaraq təmsil olunacaq.

Initial Data:

```
matrix_a = [
    [3, 7, 1],
    [5, 2, 8],
    [9, 4, 6]
]

matrix_b = [
    [6, 4, 2],
    [1, 9, 3],
    [8, 5, 7]
]
```

Tələblər:

1. `add_matrices(a, b)` — iki matrisi toplayıb nəticəni qaytarsın
2. `scalar_multiply(matrix, scalar)` — matrisi skalyar ədədə vursun
3. `transpose(matrix)` — matrisin transpozunu qaytarsın
4. `row_sum(matrix)` — hər sətrin cəmini list olaraq qaytarsın
5. `is_symmetric(matrix)` — matrisin simmetrik olub-olmadığını yoxlasın
6. `display_matrix(matrix, title)` — matrisi gözəl formatda çap etsin

Əlavə şərtlər:

- Heç bir hazır kitabxana (numpy və s.) istifadə etməyin
- Matris ölçülərinin uyğunluğunu yoxlayan validasiya əlavə edin
- Hər funksiyanı test edən `main()` funksiyası yazın

Assignment 1.3 — Parkinq Sistemi Simulyatoru

Bir parkinq sahəsinin idarə olunmasını simulyasiya edən proqram yazın.

Sistem qaydaları:

Nəqliyyat növü	Saatlıq tarif (AZN)	Cərimə (hər əlavə saat üçün)
Minik	2.00	+50%
SUV	3.50	+50%
Motosiklet	1.00	+50%

- Parkinqdə **20 yer** var
- İcazə verilən maksimum müddət: **8 saat**
- 8 saatdan artıq hər saat üçün **cərimə** tətbiq olunur (tarif +50%)

- Gecə saatlarında (22:00 – 06:00) əlavə **+30%** tutur

Tələblər:

1. `calculate_parking_fee(vehicle_type: str, hours: float, is_night: bool) -> float` — ödəniş hesablaması
2. `check_availability(current_count: int, capacity: int) -> bool` — yer olub-olmadığını yoxlasın
3. `generate_ticket(vehicle_type: str, hours: float, is_night: bool) -> str` — bilet (receipt) formatında çıxış
4. Proqram loop ilə işləsin: istifadəçi nəqliyyat növü, saat sayı və vaxt aralığı daxil etsin
5. Hər giriş-çıxışda mövcud boş yer sayını yeniləyin
6. Boş yer qalmadıqda xəbərdarlıq verin

Bonus Assignment 1.B — Refaktor: Data Structure ilə Genişləndirmə

> *Bu tapşırığı Gün 2-nin mövzularını öyrəndikdən sonra edin.*

Assignment 1.3 (Parkinq Sistemi) kodunu aşağıdakı kimi refaktor edin:

1. Hər nəqliyyatı **dictionary** olaraq saxlayın: `{"type": "Minik", "plate": "10-AB-123", "entry_hour": 14, "is_night": False}`
2. Bütün aktiv maşınları bir **list**-də saxlayın
3. Tariflər üçün **dictionary** istifadə edin: `tariffs = {"Minik": 2.0, "SUV": 3.5, "Motosiklet": 1.0}`
4. Çıxış edən maşınları axtarmaq üçün plaka nömrəsinə görə axtarış əlavə edin
5. Gün sonunda bütün gəlirləri, nəqliyyat növünə görə qruplaşdırılmış hesabat verin (**dictionary** ilə)

Gün 2 — Lists, Tuples, Dictionaries, Sets

Mövzular: List (indexing, slicing, methods), Tuple (immutability), Dictionary (key-value, methods), Set (union, intersection, etc.)

Assignment 2.1 — Tələbə Qiymətləndirmə Sistemi

Tələbələrin imtahan nəticələrini idarə edən sistem yazın.

Initial Data:

```
students_data = [
    ("Əli Həsənov", "CS101", [85, 92, 78, 90, 88]),
    ("Leyla Məmmədova", "CS101", [95, 89, 92, 97, 91]),
    ("Tural Əliyev", "CS102", [60, 55, 70, 65, 58]),
    ("Nigar Hüseynova", "CS101", [45, 52, 38, 41, 50]),
    ("Rəşad Quliyev", "CS102", [72, 68, 75, 80, 71]),
    ("Səbinə İsmayılova", "CS102", [88, 91, 85, 90, 87]),
    ("Orxan Nəsirov", "CS103", [30, 25, 40, 35, 28]),
    ("Günel Babayeva", "CS103", [78, 82, 75, 80, 79]),
    ("Kamran Rzayev", "CS103", [93, 95, 98, 91, 96]),
    ("Fidan Əsgərova", "CS102", [67, 72, 61, 58, 70])
]

grade_scale = {
    "A": (90, 100),
    "B": (80, 89),
    "C": (70, 79),
    "D": (60, 69),
    "F": (0, 59)
}
```

Tələblər:

1. Hər tələbə üçün orta qiyməti hesablayın. Ən aşağı qiyməti silərək (*drop lowest*) orta hesablayan variant da təqdim edin
2. Hər tələbəyə `grade_scale`-ə uyğun hərf qiyməti təyin edin
3. Kurs koduna görə tələbələri qruplaşdırın — **dictionary** istifadə edin: `{"CS101": [...], "CS102": [...]}`
4. Hər kurs üçün: ən yüksək, ən aşağı, orta bal, keçid faizi (D və yuxarı) hesablayın
5. "F" alan tələbələri **set** olaraq toplayın
6. İki kursun tələbələrini müqayisə edin: hər iki kursda eyni hərf qiymətini alan tələbələr varmı? (**set intersection** istifadə edin — hərf qiymətləri set-inə görə)
7. Nəticələri `tuple` olaraq qaytaran `get_student_summary(name, course, grades)` funksiyası yazın: `(ad, kurs, orta, hərf_qiyməti, keçdi_mi)`

Assignment 2.2 — İnventar İdarəetmə Sistemi

Bir anbar/mağaza üçün inventar sistemini idarə edən proqram yazın.

Initial Data:

```
inventory = {
    "SKU001": {"name": "Mexanik Klaviatura", "price": 89.99, "qty": 45, "category": "Elektronika", "supplier": "TechCo"},
    "SKU002": {"name": "Simsiz Siçan", "price": 34.50, "qty": 120, "category": "Elektronika", "supplier": "TechCo"},
    "SKU003": {"name": "USB-C Hub", "price": 52.00, "qty": 0, "category": "Aksessuar", "supplier": "GadgetPro"},
    "SKU004": {"name": "Monitor Standı", "price": 45.00, "qty": 18, "category": "Aksessuar", "supplier": "OfficePlus"},
    "SKU005": {"name": "Noutbuk Çantası", "price": 29.99, "qty": 200, "category": "Aksessuar", "supplier": "BagWorld"},
    "SKU006": {"name": "Webcam HD", "price": 75.00, "qty": 8, "category": "Elektronika", "supplier": "GadgetPro"},
    "SKU007": {"name": "Qulaqlıq", "price": 120.00, "qty": 32, "category": "Elektronika", "supplier": "AudioMax"},
    "SKU008": {"name": "Elektrik Ştekeri", "price": 12.50, "qty": 500, "category": "Aksessuar", "supplier": "OfficePlus"},
    "SKU009": {"name": "SSD 1TB", "price": 95.00, "qty": 15, "category": "Elektronika", "supplier": "TechCo"},
    "SKU010": {"name": "Ergonomik Kreslo", "price": 350.00, "qty": 5, "category": "Mebel", "supplier": "OfficePlus"}
}

transactions = [
    ("SKU001", "sell", 5),
    ("SKU002", "sell", 30),
    ("SKU003", "restock", 50),
    ("SKU006", "sell", 8),
    ("SKU005", "sell", 15),
    ("SKU009", "restock", 25),
    ("SKU007", "sell", 10),
    ("SKU004", "sell", 3),
    ("SKU001", "restock", 20),
    ("SKU010", "sell", 2),
    ("SKU008", "sell", 100),
    ("SKU003", "sell", 25),
    ("SKU006", "restock", 30),
    ("SKU002", "sell", 50),
]
```

Tələblər:

1. `process_transaction(inventory, sku, action, quantity)` — əməliyyatı icra etsin. Stokda kifayət qədər məhsul yoxdursa, xətanı idarə etsin
2. Bütün `transactions` list-ini icra edin
3. Kateqoriyaya görə qruplaşdırılmış hesabat verin: hər kateqoriyanın ümumi dəyəri (price × qty)
4. Stoku bitmiş (`qty == 0`) məhsulları tapın
5. Unikal təchizatçılar (**set**) siyahısı çıxarın
6. Hər təchizatçının neçə məhsul tədarük etdiyini göstərin
7. Ən bahalı 3 məhsulu (vahid qiymətinə görə) **tuple** listində qaytarın: `[(sku, name, price), ...]`
8. İki kateqoriyanın məhsul SKU-larını **set** əməliyyatları ilə müqayisə edin

---

Assignment 2.3 — Şəhər Hava Məlumatları Analizi

Bir neçə şəhərin bir həftəlik hava məlumatlarını analiz edən proqram yazın.

Initial Data:

```
weather_data = {
    "Bakı": {
        "temps": [12, 14, 11, 15, 13, 16, 14],
        "humidity": [65, 70, 68, 60, 72, 58, 63],
        "conditions": ["Buludlu", "Günəşli", "Yağışlı", "Günəşli", "Buludlu", "Günəşli", "Günəşli"]
    },
    "Gəncə": {
        "temps": [8, 10, 7, 12, 9, 11, 10],
        "humidity": [75, 80, 78, 70, 82, 68, 73],
        "conditions": ["Yağışlı", "Buludlu", "Yağışlı", "Günəşli", "Buludlu", "Günəşli", "Buludlu"]
    },
    "Lənkəran": {
        "temps": [15, 17, 14, 18, 16, 19, 17],
        "humidity": [80, 85, 82, 78, 88, 75, 81],
        "conditions": ["Yağışlı", "Yağışlı", "Buludlu", "Günəşli", "Yağışlı", "Buludlu", "Günəşli"]
    },
    "Şəki": {
        "temps": [5, 7, 3, 9, 6, 8, 7],
        "humidity": [70, 72, 75, 65, 78, 62, 69],
        "conditions": ["Qarlı", "Buludlu", "Qarlı", "Günəşli", "Buludlu", "Günəşli", "Buludlu"]
    }
}

days = ("Bazar ertəsi", "Çərşənbə axşamı", "Çərşənbə", "Cümə axşamı", "Cümə", "Şənbə", "Bazar")
```

Tələblər:

1. Hər şəhər üçün: orta temperatur, min/max temperatur, orta rütubət hesablayın
2. Həftə ərzində ən isti və ən soyuq günü tapın (hansı şəhərdə, hansı gün)
3. Hər şəhərdə hava şəraitinin tezliyini **dictionary** ilə hesablayın: `{"Günəşli": 4, "Buludlu": 2, ...}`
4. Günəşli günləri olan şəhərləri **set** olaraq toplayın, yağışlı günləri olan şəhərləri də ayrı set-də
5. **Set intersection** ilə həm günəşli, həm yağışlı günlər yaşayan şəhərləri tapın
6. `days` tuple-ından istifadə edərək gün adı ilə nəticələri formatlaşdırın
7. Bütün şəhərlərin temperaturlarını birləşdirərək ümumi həftəlik statistika çıxarın
8. Temperatur dəyişkənliyini hesablayın (hər şəhər üçün max - min fərqi) və ən dəyişkən şəhəri tapın

---

Bonus Assignment 2.B — Refaktor: List Comprehension ilə Sadələşdirmə

> **Bu tapşırığı Gün 3-ün mövzularını öyrəndikdən sonra edin.**

Assignment 2.3 (Hava Məlumatları) kodunu aşağıdakı kimi refaktor edin:

1. Bütün `for` loop ilə hazırlanan list-ləri **list comprehension** ilə əvəz edin
2. Temperatur filtrləmələri üçün **conditional list comprehension** istifadə edin (məs: `[t for t in temps if t > 10]`)
3. Nested dictionary-lərdə axtarış üçün **dict comprehension** istifadə edin
4. Böyük məlumat dəstləri üçün **generator** istifadə edin (məsələn, bütün şəhərlərin temperaturlarını birləşdirən generator)
5. Statistik hesablamaları **decorator** ilə ölçün — hər funksiyanın nə qədər vaxt çəkdiyini göstərən `@timer` decorator yazın

---

Gün 3 — List Comprehensions, Generators, Decorators

Mövzular: List/dict/set comprehensions, generators (yield), decorators (function decorators, stacking)

Assignment 3.1 — Log Faylı Analiz Sistemi

Bir server log faylını analiz edən proqram yazın. Böyük log fayllarını yaddaşda saxlamadan generator ilə emal edin.

Initial Data:

```
raw_logs = [
    "2026-03-10 08:15:22 INFO  UserService Login successful user=eli.hasanov ip=192.168.1.10",
    "2026-03-10 08:15:45 ERROR DatabaseService Connection timeout db=main retry=3",
    "2026-03-10 08:16:01 WARN  AuthService Failed login attempt user=unknown ip=10.0.0.55",
    "2026-03-10 08:16:30 INFO  UserService Profile updated user=leyla.mammadova ip=192.168.1.22",
    "2026-03-10 08:17:05 ERROR PaymentService Transaction failed order=ORD-445 amount=150.00",
    "2026-03-10 08:17:20 INFO  UserService Login successful user=tural.aliyev ip=192.168.1.35",
    "2026-03-10 08:18:00 WARN  AuthService Suspicious activity ip=10.0.0.55 attempts=5",
    "2026-03-10 08:18:45 INFO  OrderService Order created order=ORD-446 user=eli.hasanov total=89.99",
    "2026-03-10 08:19:10 ERROR DatabaseService Query timeout db=analytics duration=30s",
    "2026-03-10 08:19:55 INFO  UserService Login successful user=nigar.huseynova ip=192.168.1.41",
    "2026-03-10 08:20:15 WARN  PaymentService Low balance user=tural.aliyev balance=12.50",
    "2026-03-10 08:20:40 ERROR AuthService Brute force detected ip=10.0.0.55 blocked=true",
    "2026-03-10 08:21:00 INFO  OrderService Order shipped order=ORD-440 carrier=AzPost",
    "2026-03-10 08:21:30 INFO  UserService Logout user=eli.hasanov session_duration=65min",
    "2026-03-10 08:22:00 ERROR PaymentService Refund failed order=ORD-430 reason=expired",
    "2026-03-10 08:22:45 INFO  UserService Login successful user=sabina.ismayilova ip=192.168.1.50",
    "2026-03-10 08:23:10 WARN  DatabaseService High load db=main connections=95/100",
    "2026-03-10 08:23:55 INFO  OrderService Order created order=ORD-447 user=nigar.huseynova total=245.00",
    "2026-03-10 08:24:20 ERROR DatabaseService Deadlock detected db=main table=orders",
    "2026-03-10 08:25:00 INFO  UserService Password changed user=leyla.mammadova ip=192.168.1.22",
]
```

Tələblər:

1. `parse_log(line: str) -> dict` — hər log sətrini dict-ə çevirin: `{"timestamp": ..., "level": ..., "service": ..., "message": ...}`
2. **Generator** `log_reader(logs)` yazın ki, log sətrlərini bir-bir parse edib `yield` etsin
3. **List comprehension** ilə yalnız `ERROR` səviyyəli logları çıxarın
4. **Dict comprehension** ilə hər service-in neçə dəfə log yazdığını hesablayın
5. **Generator** `filter_by_level(logs, level)` yazın ki, yalnız verilmiş səviyyədəki logları qaytarsın
6. **Generator chaining**: `log_reader` → `filter_by_level` → analiz — generatorları bir-birinə bağlayın
7. IP ünvanlarına görə `WARN` və `ERROR` loglarını qruplaşdırın (dict comprehension + set)

Assignment 3.2 — Funksiya Decorator Dəsti (Decorator Toolkit)

Müxtəlif məqsədlər üçün istifadə olunan decorator-lar dəsti yazın və onları real funksiyalar üzərində tətbiq edin.

Tələblər:

1. `@timer` — funksiyanın icra müddətini ölçsün və çap etsin
2. `@call_counter` — funksiyanın neçə dəfə çağırıldığını saysın (funksiyanın `.call_count` atributunda saxlasın)
3. `@cache_result` — funksiyanın nəticəsini cache-ləsin: eyni arqumentlərlə çağırıldıqda cache-dən qaytarsın, əks halda hesablasın
4. `@validate_types(*types)` — funksiyanın arqumentlərinin tiplərini yoxlasın (parametrik decorator). Məsələn: `@validate_types(int, int)` ilə yalnız int qəbul edən funksiya
5. `@retry(max_attempts=3, delay=1)` — funksiya xəta verərsə, verilmiş sayda təkrar cəhd etsin (parametrik decorator)
6. **Stacked decorators**: bir funksiya üzərində birdən çox decorator tətbiq edin. Məsələn:

```
@timer
@call_counter
@cache_result
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)
```

1. Hər decorator-u test edin — `fibonacci`, `factorial`, və aşağıdakı funksiya ilə:

```
def process_data(data):
    """Datanı emal edib nəticə qaytarır — bilərəkdən yavaş işləyir."""
    import time
    time.sleep(0.5)
    return [x ** 2 for x in data if x % 2 == 0]
```

Assignment 3.3 — Data Pipeline (Comprehension + Generator ilə)

E-ticarət sifarişlərini analiz edən data pipeline qurun. Bütün pipeline generator-lardan ibarət olmalıdır.

Initial Data:

```
orders = [
      {"id": "ORD-1001", "customer": "Əli", "items": [("Laptop", 2500, 1), ("Siçan", 35, 2)], "city": "Bakı", "status":
"delivered", "date": "2026-03-01"},
     {"id": "ORD-1002", "customer": "Leyla", "items": [("Telefon", 1800, 1), ("Qabıq", 25, 1)], "city": "Gəncə", "status":
"delivered", "date": "2026-03-02"},
      {"id": "ORD-1003", "customer": "Tural", "items": [("Qulaqlıq", 120, 3), ("Kabel", 10, 5)], "city": "Bakı", "status":
"cancelled", "date": "2026-03-02"},
      {"id": "ORD-1004", "customer": "Nigar", "items": [("Monitor", 800, 2), ("Klaviatura", 90, 1)], "city": "Sumqayıt",
"status": "delivered", "date": "2026-03-03"},
      {"id": "ORD-1005", "customer": "Rəşad", "items": [("Printer", 450, 1), ("Kağız", 15, 10)], "city": "Bakı", "status":
"shipped", "date": "2026-03-03"},
     {"id": "ORD-1006", "customer": "Səbinə", "items": [("Tablet", 650, 1), ("Stylus", 40, 1)], "city": "Lənkəran", "status":
"delivered", "date": "2026-03-04"},
    {"id": "ORD-1007", "customer": "Orxan", "items": [("SSD", 95, 2), ("RAM", 70, 4)], "city": "Bakı", "status": "delivered",
"date": "2026-03-05"},
     {"id": "ORD-1008", "customer": "Günel", "items": [("Webcam", 75, 1), ("Mikrofon", 110, 1)], "city": "Gəncə", "status":
"returned", "date": "2026-03-05"},
      {"id": "ORD-1009", "customer": "Kamran", "items": [("GPU", 1200, 1), ("PSU", 180, 1)], "city": "Bakı", "status":
"delivered", "date": "2026-03-06"},
    {"id": "ORD-1010", "customer": "Fidan", "items": [("Router", 85, 1), ("Ethernet Kabel", 8, 3)], "city": "Şəki", "status":
"shipped", "date": "2026-03-07"},
      {"id": "ORD-1011", "customer": "Əli", "items": [("USB Hub", 52, 2), ("Adapter", 18, 3)], "city": "Bakı", "status":
"delivered", "date": "2026-03-07"},
    {"id": "ORD-1012", "customer": "Leyla", "items": [("Smart Saat", 320, 1)], "city": "Gəncə", "status": "delivered", "date":
"2026-03-08"},
]
```

Tələblər:

1. `calculate_order_total(order)` — **list comprehension** ilə sifarişin ümumi məbləğini hesablasın (`price × qty` cəmi)
2. **Generator** `active_orders(orders)` — yalnız `delivered` və `shipped` sifarişləri yield etsin
3. **Generator** `high_value_orders(orders, threshold)` — məbləği threshold-dan yuxarı sifarişləri yield etsin
4. **Dict comprehension** ilə şəhərlərə görə sifariş sayını hesablayın
5. **Generator pipeline** qurun:

```
orders → active_orders → high_value_orders(500) → calculate totals → group by city
```

1. **Set comprehension** ilə unikal müştəriləri tapın
2. Hər müştərinin ümumi xərclərini **dict comprehension** ilə hesablayın
3. `@timer` decorator yazıb pipeline-ın hər mərhələsinin vaxtını ölçün
4. Pipeline-ın nəticəsini gözəl formatda çap edən `report_generator(pipeline_result)` generator yazın

---

Həftəlik Assignment — Kitabxana İdarəetmə Sistemi

> *Bu tapşırıq bütün həftənin mövzularını əhatə edir: control structures, functions, data structures (list, tuple, dict, set), list/dict comprehensions, generators, və decorators.*

Sistemin Təsviri

Şəhər kitabxanası üçün tam funksional idarəetmə sistemi yazın.

Initial Data:

```
books = [
    {"isbn": "978-1-001", "title": "Python Proqramlaşdırma", "author": "A. Məmmədov", "genre": "Texnologiya", "year": 2023,
"copies": 5, "available": 5},
    {"isbn": "978-1-002", "title": "Data Elmi Əsasları", "author": "L. Əliyeva", "genre": "Texnologiya", "year": 2024,
"copies": 3, "available": 3},
    {"isbn": "978-1-003", "title": "Süni İntellekt", "author": "R. Hüseynov", "genre": "Texnologiya", "year": 2022, "copies":
4, "available": 2},
    {"isbn": "978-1-004", "title": "Azərbaycan Tarixi", "author": "T. Qasımov", "genre": "Tarix", "year": 2020, "copies": 6,
"available": 6},
    {"isbn": "978-1-005", "title": "Modern Fizika", "author": "N. Babayev", "genre": "Elm", "year": 2021, "copies": 2,
"available": 0},
    {"isbn": "978-1-006", "title": "Psixologiya Giriş", "author": "S. Vəliyeva", "genre": "Psixologiya", "year": 2023,
"copies": 4, "available": 4},
    {"isbn": "978-1-007", "title": "Riyaziyyat Təhlili", "author": "K. Nəsirov", "genre": "Elm", "year": 2019, "copies": 3,
"available": 1},
    {"isbn": "978-1-008", "title": "İqtisadiyyata Giriş", "author": "G. Əsgərova", "genre": "İqtisadiyyat", "year": 2022,
"copies": 5, "available": 5},
    {"isbn": "978-1-009", "title": "Web Development", "author": "A. Məmmədov", "genre": "Texnologiya", "year": 2024, "copies":
2, "available": 2},
    {"isbn": "978-1-010", "title": "Dünya Ədəbiyyatı", "author": "F. Rəhimova", "genre": "Ədəbiyyat", "year": 2021, "copies":
7, "available": 5},
]

members = [
    {"id": "MEM-001", "name": "Əli Həsənov", "membership": "Premium", "borrowed": [], "history": []},
    {"id": "MEM-002", "name": "Leyla Məmmədova", "membership": "Standard", "borrowed": [], "history": []},
    {"id": "MEM-003", "name": "Tural Əliyev", "membership": "Premium", "borrowed": [], "history": []},
    {"id": "MEM-004", "name": "Nigar Hüseynova", "membership": "Standard", "borrowed": [], "history": []},
    {"id": "MEM-005", "name": "Rəşad Quliyev", "membership": "Premium", "borrowed": [], "history": []},
]

membership_rules = {
    "Premium": {"max_books": 5, "loan_days": 30, "can_reserve": True},
    "Standard": {"max_books": 3, "loan_days": 14, "can_reserve": False},
}

borrow_log = [
    ("MEM-001", "978-1-003", "2026-03-01"),
    ("MEM-001", "978-1-005", "2026-03-01"),
    ("MEM-003", "978-1-003", "2026-03-02"),
    ("MEM-002", "978-1-007", "2026-03-03"),
    ("MEM-004", "978-1-010", "2026-03-04"),
    ("MEM-005", "978-1-005", "2026-03-04"),
    ("MEM-004", "978-1-010", "2026-03-05"),
    ("MEM-001", "978-1-007", "2026-03-06"),
]
```

Tələblər

Hissə 1 — Əsas Əməliyyatlar (Functions + Control Structures)

1. `find_book(books, **kwargs)` — ISBN, ad, müəllif, janr ilə kitab axtarışı (birdən çox kriteriya dəstəkləsin)
2. `borrow_book(member, book, date)` — kitab verilməsi. Yoxlasın:
   - Üzvün götürə biləcəyi maks. kitab sayını keçib-keçmədiyini
   - Kitabın mövcud olub-olmadığını
   - Kitabı artıq götürüb-götürmədiyini
1. `return_book(member, book, date)` — kitabın qaytarılması, gecikmə hesablaması (loan_days-dən artıq)
2. `process_borrow_log(books, members, log)` — `borrow_log` üzərində əməliyyatları icra edin

Hissə 2 — Analiz və Hesabatlar (Data Structures)

1. Janra görə kitabları qruplaşdırın (**dict**)
2. Hər müəllifin kitab sayını tapın (**dict**)
3. Hazırda mövcud olan kitabların ISBN-lərini **set** olaraq verin
4. Heç bir nüsxəsi qalmamış kitabları **set** olaraq tapın
5. Premium və Standard üzvlərin götürdüyü kitabların ISBN **set**-lərini müqayisə edin (**intersection**, **difference**)
6. Hər üzvün kitab götürmə tarixçəsini **list of tuples** olaraq saxlayın: `[(isbn, borrow_date, return_date), ...]`

Hissə 3 — Advanced (Comprehensions + Generators + Decorators)

1. **List comprehension** ilə mövcud kitabları filter edin, **dict comprehension** ilə janra görə statistika çıxarın
2. **Generator** `overdue_checker(members, current_date)` — gecikmiş kitabları yield etsin
3. **Generator** `popular_books(borrow_log)` — ən çox götürülən kitabları tezliyə görə yield etsin
4. **Generator pipeline** qurun: bütün kitablar → mövcud olanlar → janra görə filtr → sorted

Decorators:

1. `@log_action` — hər kitab əməliyyatını log-lasın (kim, nə vaxt, hansı kitab, hansı əməliyyat)
2. `@permission_check` — üzvün membership tipinə görə əməliyyata icazə verib-vermədiyini yoxlasın
3. `@performance_monitor` — funksiyanın icra vaxtını və yaddaş istifadəsini ölçsün

Nəticə hesabatı:

1. Sistem aşağıdakı hesabatı çıxara bilməlidir:

```
       KİTABXANA HESABATI — 2026-03-17


 Ümumi kitab sayı: 10
 Mövcud kitablar: 7
 Stokda olmayan: 2
 Üzv sayı: 5 (Premium: 3, Standard: 2)

 Janra Görə Statistika
Texnologiya: 4 kitab (11 ədəd mövcud)
Tarix:       1 kitab (6 ədəd mövcud)
...

 Ən Populyar Kitablar
1. Modern Fizika — 2 dəfə götürülüb
2. Riyaziyyat Təhlili — 2 dəfə götürülüb
...

 Gecikmiş Kitablar
 Əli Həsənov — "Süni İntellekt" (16 gün gecikib)
...
```