سؤال ۱: ترکیب map و filter

با توجه به لیست اعداد زیر، با استفاده از ترکیب توابع map و filter ، لیستی جدید بسازید که شامل توان سوم اعداد فرد باشد.

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

# کد خود را در یک خط بنویسید
# result = ...

print(list(result))

# خروجی مورد انتظار:
# [1, 27, 125, 343]
```

سؤال ۲: محاسبه پیچیده با reduce

با توجه به لیستی از دیکشنری‌های زیر که اطلاعات محصولات را نشان می‌دهد، با استفاده از تابع reduce، مجموع کل قیمت تمام محصولات را محاسبه کنید. (قیمت هر محصول برابر است با price * quantity).

```
from functools import reduce

products = [
    {'name': 'Laptop', 'price': 1500, 'quantity': 2},
    {'name': 'Mouse', 'price': 25, 'quantity': 10},
    {'name': 'Keyboard', 'price': 75, 'quantity': 5}
]

# کد شما برای محاسبه مجموع کل
# total_value = ...

print(f"Total inventory value: {total_value}")

# خروجی مورد انتظار:
# Total inventory value: 3625
```

سؤال ۳: دکوراتور کنترل نوع ورودی

یک دکوراتور به نام enforce_types بنویسید که لیستی از انواع داده مثل (int, str) را به عنوان ورودی بگیرد و قبل از اجرای تابع اصلی، چک کند که آیا آرگومان‌های ورودی به تابع، از همان نوع مشخص شده هستند یا خیر. اگر نوع داده‌ها صحیح نبود، یک TypeError ایجاد کند.

```
@enforce_types(int, int)
def add(a, b):
    return a + b

@enforce_types(str, int)
def repeat_message(message, times):
    for _ in range(times):
        print(message)

# این فراخوانی باید به درستی کار کند
print(add(5, 10))

# این فراخوانی باید TypeError ایجاد کند
# add(5, "10")

# این فراخوانی باید به درستی کار کند
repeat_message("Hello", 3)

# خروجی مورد انتظار:
# 15
# Hello
# Hello
# Hello
# (یک TypeError را از کامنت خارج کنید، و اگر خطای add("5", 10))
```

سؤال ۴: ساخت یک Generator برای تولید دنباله فیبوناچی

یک تابع Generator به نام fib_generator بنویسید که یک عدد nرا به عنوان ورودی بگیرد و **nعدد اول** از دنباله فیبوناچی را یک به یک yield کند.

دنباله فیبوناچی... ,0, 1, 1, 2, 3, 5, 8 :

```
def fib_generator(n):
    # کد شما برای تولید دنباله
    pass

# ۱۰ عدد اول دنباله را چاپ کنید
for number in fib_generator(10):
    print(number)

# خروجی مورد انتظار:
# 0
# 1
# 1
# 2
```

سؤال ۵: گروه‌بندی داده‌ها با itertools.groupby سخت

با توجه به لیستی از دیکشنری‌ها که اطلاعات کارمندان را نشان می‌دهد، آن‌ها را بر اساس **دپارتمان (department)**گروه‌بندی کنید. سپس نام کارمندان هر دپارتمان را در یک لیست جداگانه چاپ کنید.

نکته مهم :برای اینکه groupby به درستی کار کند، لیست ورودی باید ابتدا بر اساس همان کلیدی که برای گروه‌بندی استفاده می‌شود (در اینجا department) مرتب شده باشد.

```python
from itertools import groupby

employees = [
    {'name': 'Ali', 'department': 'HR'},
    {'name': 'Sara', 'department': 'Engineering'},
    {'name': 'Reza', 'department': 'HR'},
    {'name': 'Maryam', 'department': 'Engineering'},
    {'name': 'Nima', 'department': 'Sales'},
    {'name': 'Hadi', 'department': 'Sales'}
]

# مرحله ۱: مرتب‌سازی لیست
# ...

# مرحله ۲: گروه‌بندی و چاپ
# ...

# خروجی مورد انتظار:
# Department: Engineering -> ['Sara', 'Maryam']
# Department: HR -> ['Ali', 'Reza']
# Department: Sales -> ['Nima', 'Hadi']
```

سؤال ۶: مرتب‌سازی چندسطحی با lambda

لیستی از دیکشنری‌ها در زیر وجود دارد که هر کدام اطلاعات یک محصول را نمایش می‌دهند. این لیست را با استفاده از تابع sortedو یک عبارت lambdaمرتب کنید. معیار مرتب‌سازی به این صورت است:

۱. ابتدا بر اساس **امتیاز (rating)** به صورت **نزولی**.

۲. اگر امتیازها یکسان بود، بر اساس **قیمت (price)** به صورت **صعودی**.

```
products = [
    {'name': 'A', 'price': 150, 'rating': 4.5},
    {'name': 'B', 'price': 200, 'rating': 4.2},
    {'name': 'C', 'price': 120, 'rating': 4.5},
    {'name': 'D', 'price': 250, 'rating': 4.8},
    {'name': 'E', 'price': 180, 'rating': 4.2}
]

# sorted_products = ...

for p in sorted_products:
    print(p)

# خروجی مورد انتظار:
# {'name': 'D', 'price': 250, 'rating': 4.8}
# {'name': 'C', 'price': 120, 'rating': 4.5}
# {'name': 'A', 'price': 150, 'rating': 4.5}
# {'name': 'E', 'price': 180, 'rating': 4.2}
# {'name': 'B', 'price': 200, 'rating': 4.2}
```

سؤال ۷: زنجیره پردازش داده با Generator

فرض کنید یک لاگ فایل چندخطی به صورت یک رشته در اختیار دارید. یک زنجیره (pipeline) از سه Generator بنویسید که:

۱. **read_lines** : خطوط رشته را یک به یک yield کند.

۲. **filter_errors** : از ورودی خود (که یک generator است) ، فقط خطوطی که شامل کلمه "ERROR"هستند را yield کند.

۳. **extract_message** : از ورودی خود، فقط پیام اصلی را استخراج و yieldکند (یعنی بخش بعد از "ERROR: ")

```
log_data = """
INFO: Starting process...
DEBUG: Connecting to database.
ERROR: Connection timeout.
INFO: Process finished.
ERROR: File not found.
"""

def read_lines(data):
    for line in data.strip().split('\n'):
        yield line

def filter_errors(lines_gen):
    # کد شما
    pass

def extract_message(errors_gen):
    # کد شما
    pass

# ساخت زنجیره
log_lines = read_lines(log_data)
error_lines = filter_errors(log_lines)
error_messages = extract_message(error_lines)

# اجرای زنجیره
for msg in error_messages:
    print(msg)

# خروجی مورد انتظار:
# Connection timeout.
# File not found.
```

سؤال ۸: مسطح کردن لیست تودرتو با تابع بازگشتی

یک تابع بازگشتی به نام flatten بنویسید که یک لیست تودرتو (با هر عمقی) را به عنوان ورودی دریافت کرده و آن را به یک لیست تک‌سطحی تبدیل کند.

```
def flatten(nested_list):
    flat_list = []
    # کد بازگشتی شما اینجا
    # برای هر آیتم در لیست، چک کنید که آیا خودش یک لیست است یا خیر
    pass

my_list = [1, [2, 3], [4, [5, 6]], 7]
result = flatten(my_list)
print(result)

# خروجی مورد انتظار:
# [1, 2, 3, 4, 5, 6, 7]
```

سؤال۹ : ارسال ایمیل به کاربران فعال

شما لیستی از دیکشنری‌ها را در اختیار دارید که هر کدام اطلاعات یک کاربر، شامل نام، ایمیل و وضعیت حساب (status)را نمایش می‌دهد.

با استفاده از ترکیب filterو mapدر یک خط، لیستی از رشته‌ها ایجاد کنید که فقط برای کاربران **فعال (active)** آماده شده باشد. هر رشته باید در قالب "Dear [NAME], welcome back!"باشد، به طوری که نام کاربر با حروف بزرگ نوشته شود.

```
users = [
    {'name': 'Ali', 'email': 'ali@example.com', 'status': 'active'},
    {'name': 'Sara', 'email': 'sara@example.com', 'status': 'inactive'},
    {'name': 'Reza', 'email': 'reza@example.com', 'status': 'active'},
    {'name': 'Maryam', 'email': 'maryam@example.com', 'status': 'active'},
    {'name': 'Nima', 'email': 'nima@example.com', 'status': 'inactive'}
]

# کد خود را در یک خط بنویسید
# welcome_messages = list(...)

print(welcome_messages)

# خروجی مورد انتظار:
# ["Dear ALI, welcome back!", "Dear REZA, welcome back!", "Dear MARYAM, welcome ba
```