تمرینات عملی فصل ۱:

تمرین ۱User Management System :

```
class UserManager:
    def __init__(self):
        self.users = []

    def add_user(self, username: str, email: str,
                 role: str = 'user') -> dict:
        user = {
            'id': len(self.users) + ١,
            'username': username,
            'email': email,
            'role': role,
            'is_active': True
        }
        self.users.append(user)
        return user

    def get_active_users(self):
        return [u for u in self.users if u['is_active']]
```

```
    def get_users_by_role(self, role: str):
        return filter(lambda u: u['role'] == role, self.users)

# استفاده
manager = UserManager()
manager.add_user('john', 'john@example.com', 'admin')
manager.add_user('jane', 'jane@example.com', 'user')

print(manager.get_active_users())
```

تمرین ۲: Decorator برای Logging

```
python
import functools
import logging

def log_execution(func):
    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        logging.info(f"Executing {func.__name__}")
        try:
            result = func(*args, **kwargs)
```

```
            logging.info(f"{func.__name__} completed successfully")
            return result
        except Exception as e:
            logging.error(f"{func.__name__} failed: {str(e)}")
            raise
    return wrapper

@log_execution
def create_post(title, content):
    # شبیه‌سازی ساخت post
    return {'title': title, 'content': content}
```

تمرین ۳: Data Processing Pipeline :

```
python
def read_csv_data(filename):
    """Generator برای خواندن CSV"""
    with open(filename) as f:
        for line in f:
            yield line.strip().split(',')

def validate_data(rows):
    """Validation داده‌ها"""
```

```
    for row in rows:
        if len(row) == ۳:  # فرض: ۳ ستون
            yield row

def transform_data(rows):
    """تبدیل داده‌ها به dictionary"""
    for row in rows:
        yield {
            'name': row[۰],
            'email': row[۱],
            'age': int(row[۲])
        }

# Pipeline
data = read_csv_data('users.csv')
valid_data = validate_data(data)
transformed = transform_data(valid_data)

for user in transformed:
    print(user)
```

پروژه عملی Mini Blog System :

python

```python
from typing import List, Dict, Optional
from datetime import datetime

class BlogPost:
    def __init__(self, title: str, content: str, author: str):
        self.id = None
        self.title = title
        self.content = content
        self.author = author
        self.created_at = datetime.now()
        self.tags = []

    def add_tag(self, tag: str):
        if tag not in self.tags:
            self.tags.append(tag)

    def to_dict(self) -> Dict:
        return {
            'id': self.id,
            'title': self.title,
```

```
            'content': self.content,
            'author': self.author,
            'created_at': self.created_at.isoformat(),
            'tags': self.tags
        }

class BlogManager:
    def __init__(self):
        self.posts: List[BlogPost] = []
        self.next_id = 1

    def create_post(self, title: str, content: str,
                    author: str) -> BlogPost:
        post = BlogPost(title, content, author)
        post.id = self.next_id
        self.next_id += 1
        self.posts.append(post)
        return post

    def get_posts_by_author(self, author: str) -> List[BlogPost]:
        return [p for p in self.posts if p.author == author]
```

```
    def get_posts_by_tag(self, tag: str) -> List[BlogPost]:
        return [p for p in self.posts if tag in p.tags]

    def search_posts(self, keyword: str) -> List[BlogPost]:
        return [p for p in self.posts
                if keyword.lower() in p.title.lower()
                or keyword.lower() in p.content.lower()]

# استفاده
blog = BlogManager()
post۱ = blog.create_post("Django Tutorial",
                         "Learn Django...",
                         "John")
post۱.add_tag("django")
post۱.add_tag("python")

post۲ = blog.create_post("Python Tips",
                         "Advanced Python...",
                         "Jane")
post۲.add_tag("python")
```

```
# جستجو
python_posts = blog.get_posts_by_tag("python")
print(f"Found {len(python_posts)} posts with 'python' tag")
```