Object-Oriented Programming (OOP)

English: Object-oriented programming relies on objects and classes. It includes concepts like inheritance, polymorphism, encapsulation, and abstraction.
Arabic: البرمجة الكائنية تعتمد على الكائنات والفئات. تتضمن مفاهيم مثل الوراثة، تعدد الأشكال، التغليف، والتجريد.

```
**Example:**
```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return f"{self.name} says Woof!"

class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow!"

dog = Dog("Buddy")
cat = Cat("Kitty")

print(dog.speak())  # Buddy says Woof!
print(cat.speak())  # Kitty says Meow!
```

SOLID Principles

English: Software design principles to facilitate development and maintenance.
Arabic: مبادئ تصميم البرمجيات لتسهيل التطوير والصيانة.

Single Responsibility Principle (SRP)

English: A class should have only one reason to change.
Arabic: يجب أن يكون للفئة سبب واحد فقط للتغيير.

Open/Closed Principle (OCP)

English: Software entities should be open for extension but closed for modification.
Arabic: يجب أن تكون الكيانات البرمجية مفتوحة للتوسع ولكن مغلقة للتعديل.

Liskov Substitution Principle (LSP)

English: Subtypes must be substitutable for their base types.
Arabic: يجب أن تكون الأنواع الفرعية قابلة للاستبدال بأنواعها الأساسية.

Interface Segregation Principle (ISP)

English: Many client-specific interfaces are better than one general-purpose interface.
Arabic: العديد من الواجهات الخاصة بالعميل أفضل من واجهة واحدة لأغراض عامة.
```

Dependency Inversion Principle (DIP)

English: Depend on abstractions, not concretions.
Arabic: يعتمد على التجريدات، وليس على التطبيقات الفعلية.

Example:

```
# SRP Example
class Report:
    def __init__(self, data):
        self.data = data

    def generate_report(self):
        return f"Report: {self.data}"

class ReportPrinter:
    def __init__(self, report):
        self.report = report

    def print_report(self):
        print(self.report.generate_report())

report = Report("Sales Data")
printer = ReportPrinter(report)
printer.print_report()
```

ACID Properties

English: Database properties ensuring data integrity.
Arabic: خصائص قاعدة البيانات التي تضمن سلامة البيانات.

Atomicity

English: Transactions are all-or-nothing.
Arabic: المعاملات إما أن تتم بالكامل أو لا تتم على الإطلاق.

Consistency

English: Transactions bring the database from one valid state to another.
Arabic: المعاملات تنقل قاعدة البيانات من حالة صحيحة إلى أخرى.

Isolation

English: Transactions do not interfere with each other.
Arabic: المعاملات لا تتداخل مع بعضها البعض.

Durability

English: Once a transaction is committed, it remains so.
Arabic: بمجرد تأكيد المعاملة، تبقى كذلك.

Design Patterns

English: Ready-made solutions for common design problems such as Singleton, Factory, Observer.
Arabic: حلول جاهزة لمشاكل التصميم الشائعة مثل النمط الفردي، المصنع، المراقب.

Example:

```
# Singleton Example
class Singleton:
    _instance = None

    def __new__(cls, *args, **kwargs):
        if not cls._instance:
            cls._instance = super(Singleton, cls).__new__(cls, *args, **kwargs)
        return cls._instance

singleton1 = Singleton()
singleton2 = Singleton()

print(singleton1 is singleton2)  # True
```

Python Data Types

English: Data types in Python such as int, float, str, list, dict.
Arabic: أنواع البيانات في بايثون مثل int، float، str، list، dict.

Example:

```
# List
my_list = [1, 2, 3, 4]
print(my_list)

# Tuple
my_tuple = (1, 2, 3, 4)
print(my_tuple)

# Dictionary
my_dict = {'name': 'John', 'age': 30}
print(my_dict)

# Set
my_set = {1, 2, 3, 4}
print(my_set)
```

List Comprehension

English: A concise way to create lists based on existing lists with conditional expressions.
Arabic: طريقة مختصرة لإنشاء القوائم استنادًا إلى القوائم الموجودة مع تعبيرات شرطية.

Example:

```python
# List comprehension to create a list of squares
squares = [x**2 for x in range(10)]
print(squares)  # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Python Generators

English: A function that yields a series of values using 'yield'.
Arabic: دالة تنتج سلسلة من القيم باستخدام 'yield'.

Example:

```python
def my_generator():
    yield 1
    yield 2
    yield 3

gen = my_generator()
print(next(gen))  # 1
print(next(gen))  # 2
print(next(gen))  # 3
```

Python Metaclasses

English: A class that controls the behavior of other classes.
Arabic: فئة تتحكم في سلوك الفئات الأخرى.

Example:

```python
class Meta(type):
    def __new__(cls, name, bases, dct):
        print(f"Creating class {name}")
        return super().__new__(cls, name, bases, dct)

class MyClass(metaclass=Meta):
    pass

# Output: Creating class MyClass
```

Functional Programming

English: Programming paradigm that relies on functions and mathematical expressions.
Arabic: نموذج برمجي يعتمد على الدوال والتعبيرات الرياضية.

Example:

```python
# Using map and lambda for functional programming
numbers = [1, 2, 3, 4]
squared = list(map(lambda x: x**2, numbers))
print(squared)  # [1, 4, 9, 16]
```

2. Django Topics

Django

English: A Python web framework.
Arabic: إطار عمل ويب مبني على بايثون.

Signals

English: Mechanism for alerting between application components.
Arabic: آلية للتنبيه بين مكونات التطبيق.

Views

English: Logic for handling requests.
Arabic: منطق معالجة الطلبات.

ViewSet

English: Set of views in REST framework.
Arabic: مجموعة من العروض في إطار عمل REST.

Generic View

English: Ready-made views for common operations.
Arabic: عروض جاهزة للعمليات الشائعة.

Mixins

English: Add-ons to extend view functionality.
Arabic: إضافات لتوسيع وظيفة العروض.

Docker, Celery in Django, Redis

English: Docker is a tool for running applications in isolated containers. Celery is a library for asynchronous task execution. Redis is an in-memory database used for caching.
Arabic: Docker هو أداة لتشغيل التطبيقات في حاويات معزولة. Celery هو مكتبة لتنفيذ المهام بشكل غير متزامن. Redis هو قاعدة بيانات في الذاكرة تستخدم للتخزين المؤقت.

How to Deploy

English: Deploying applications using tools like Docker, Kubernetes, or cloud services like AWS.
Arabic: نشر التطبيقات باستخدام أدوات مثل Docker، Kubernetes، أو خدمات السحابة مثل AWS.

CRUD in Django and REST Framework API

English: Creating CRUD interfaces using Django and Django REST framework.
Arabic: إنشاء واجهات CRUD باستخدام Django و إطار عمل Django REST.

Database

English: DB refers to databases. Query in Django refers to writing queries using ORM. Filtering F, Q refers to using F and Q classes for complex data filtering.
Arabic: DB يشير إلى قواعد البيانات. Query in Django يشير إلى كتابة الاستعلامات باستخدام ORM. Filtering F، Q يشير إلى استخدام فئات F و Q لتصفية البيانات المعقدة.

MVT, Load Balancer

English: MVT (Model-View-Template) is a design pattern used in Django. A Load Balancer is used for distributing load across multiple servers for better performance.
Arabic: MVT (Model-View-Template) هو نمط تصميم مستخدم في Django. Load Balancer يستخدم لتوزيع الحمل عبر عدة خوادم لتحسين الأداء.

MVT, Load Balancer

MVT (Model-View-Template)

- **English:** Design pattern used in Django.
- **Arabic:** نمط التصميم المستخدم في Django.

```python
# Example: Django View using MVT
from django.shortcuts import render
from django.http import HttpResponse
from .models import MyModel

def my_view(request):
    queryset = MyModel.objects.all()
    return render(request, 'template.html', {'queryset': queryset})
```

Load Balancer

- **English:** Distributing load across multiple servers for better performance.
- **Arabic:** توزيع الحمل عبر عدة خوادم لتحسين الأداء.

```
# Example: Setting up a Load Balancer in AWS
# Configuration file (nginx.conf)
http {
    upstream myapp {
        server 10.0.0.1;
        server 10.0.0.2;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://myapp;
        }
```

```
    }
}
```

AWS Intro, EC2, S3, Auto-scaling

AWS Intro

- **English:** Introduction to Amazon Web Services.
- **Arabic:** مقدمة عن خدمات الويب من أمازون.

EC2

- **English:** Cloud computing service.
- **Arabic:** خدمة الحوسبة السحابية.

S3

- **English:** File storage service.
- **Arabic:** خدمة تخزين الملفات.

Auto-scaling

- **English:** Automatically adjusting resources based on demand.
- **Arabic:** ضبط الموارد تلقائياً استناداً إلى الطلب.

```
# Example: Auto-scaling configuration in AWS
# CloudFormation template
Resources:
  MyAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchConfigurationName: MyLaunchConfiguration
      MinSize: '2'
      MaxSize: '6'
      DesiredCapacity: '4'
```

Nginx and How to Deploy to AWS

Nginx

- **English:** Web server and reverse proxy.
- **Arabic:** خادم ويب وبروكسي عكسي.

```
# Example: Nginx configuration for a Django application
server {
    listen 80;
    server_name example.com;

    location / {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

How to Deploy to AWS

- **English:** Deploying applications on AWS using EC2, S3, and other services.
- **Arabic:** نشر التطبيقات على AWS باستخدام EC2 و S3 وخدمات أخرى.

```
# Example: Deploying Django app to AWS using Elastic Beanstalk
# ebextensions/myapp.config
option_settings:
  aws:elasticbeanstalk:container:python:
    WSGIPath: myapp.wsgi:application
```

API, Auth in REST API, Method API

REST API

- **English:** Design pattern using HTTP.
- **Arabic:** نمط التصميم باستخدام HTTP.

```
# Example: Creating a REST API endpoint in Django
from rest_framework.decorators import api_view
from rest_framework.response import Response

@api_view(['GET'])
def hello_world(request):
    return Response({"message": "Hello, world!"})
```

Auth in REST API

- **English:** Authentication methods like Token, OAuth.
- **Arabic:** طرق المصادقة مثل التوكن و OAuth.

```python
# Example: Token authentication in Django REST framework
from rest_framework.authentication import TokenAuthentication
from rest_framework.permissions import IsAuthenticated

class MyView(APIView):
    authentication_classes = [TokenAuthentication]
    permission_classes = [IsAuthenticated]

    def get(self, request, format=None):
        content = {
            'message': 'Hello, World!'
        }
        return Response(content)
```

Method API

- **English:** APIs based on function calls.
- **Arabic:** واجهات برمجة التطبيقات استناداً إلى استدعاءات الدوال.

```python
# Example: Creating a Method API in Django
from django.http import JsonResponse

def hello(request):
    return JsonResponse({'message': 'Hello, world!'})
```

Microservice and Monolithic

Microservice

- **English:** Breaking applications into small, independent services.
- **Arabic:** تقسيم التطبيقات إلى خدمات صغيرة ومستقلة.

```python
# Example: Microservice architecture using Flask
# Service 1
@app.route('/service1')
def service1():
    return 'Service 1'

# Service 2
@app.route('/service2')
def service2():
    return 'Service 2'
```

Monolithic

- **English:** Single large application containing all functionalities.
- **Arabic:** تطبيق كبير واحد يحتوي على جميع الوظائف.

```python
# Example: Monolithic application in Django
# views.py
from django.shortcuts import render
from .models import MyModel

def my_view(request):
    queryset = MyModel.objects.all()
    return render(request, 'template.html', {'queryset': queryset})
```

ASP, SSR

ASP

- **English:** Application single page (API).
- **Arabic:** تطبيق صفحة واحدة (API).

SSR (Server-Side Rendering)

- **English:** Generating HTML pages on the server.
- **Arabic:** إنشاء صفحات HTML على الخادم.

Other Topics

What's Different Between Library, Package, Module, Framework

Library

- **English:** Collection of reusable functions.
- **Arabic:** مجموعة من الدوال قابلة لإعادة الاستخدام.

Package

- **English:** Unit of distribution containing libraries.
- **Arabic:** وحدة التوزيع تحتوي على المكتبات.

Module

- **English:** Python file containing code.
- **Arabic:** ملف بايثون يحتوي على الشفرة.

Framework

- **English:** Work environment providing a foundation for application development.
- **Arabic:** بيئة عمل توفر أساساً لتطوير التطبيقات.

```
# Example: Django as a web framework
# settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Git Commands and Shortcut Concepts in Programming

Git Commands

- **English:** Commands like 'git clone', 'git commit', 'git push'.
- **Arabic:** أوامر مثل 'git clone', 'git commit', 'git push'.

```
# Example: Basic Git commands
git clone https://github.com/repository.git
git commit -m "Commit message"
git push origin master
```

Shortcut Concepts

- **English:** Concepts to improve code such as DRY (Don't Repeat Yourself), KISS (Keep It Simple, Stupid).
- **Arabic:** مفاهيم لتحسين الشفرة مثل DRY، KISS.

```
# Example: Applying DRY principle in Python
def calculate_sum(numbers):
    total = 0
    for num in numbers:
        total += num
    return total
```