장고 대비 답지

1. web framework - T/F를 판별하시오

- **Static web page(정적 웹 페이지)**
  - 서버가 정적 웹 페이지에 대한 요청을 받은 경우, 서버는 추가적인 처리 과정 없이 클라이언트에게 응답을 보낸다 (O)
  - **모든** 상황에서, **모든** 사용자에게 **동일한** 정보를 표시한다 (O)
  - 일반적으로 HTML, CSS, JavaScript로 작성된다 (O)
  - flat page라고도 불리운다 (O)
  - 클라이언트는 서버에 요청을하고, 서버는 클라이언트에 응답을 하는 과정을 거친다.
    - 여기서 클라이언트는 Django, 서버는 웹브라우저(크롬)에 해당한다 (X)

- **Dynamic web page(동적 웹 페이지)**
  - 웹 페이지에 대한 요청을 받은 경우 서버는 추가적인 처리 과정 없이 클라이언트에게 응답을 보내게 된다 (X)
  - 동적 페이지는 방문자와 상호작용하기 때문에 페이지 내용은 그때그때마다 다르다 (O)
  - 마찬가지로, HTML, CSS, JavaScript가 주력언어로 사용된다 (X)

2. Frameworkd Architecture - T/F를 판별하시오

- Django는 Python Web Framework이다 (O)
- Django는 MTV Design Pattern을 채용하고 있다 (O)
- Django는 사용자 인터페이스로부터 프로그램 로직을 분리하여, 애플리케이션의 시각적 요소나 이면에서 실행되는 부분을 독립적으로, 개별적으로 쉽게 고칠 수 있도록 한다 (

O)

- Django의 **Model**은 응용프로그램의 데이터 구조를 정의하고 데이터베이스의 기록을 관리만 한다 (X)
- Django의 **Template**는 파일의 구조나 레이아웃을 정의한다 (O)
- Django의 **View**는 컨트롤 타우의 역할을 하는데,
  - HTTP 요청을 수신하고, HTTP응답을 반환하는 역할을 한다 (O)
  - Model을 통해 요청을 충족시키기 위해 필요한 데이터에 접근한다 (O)

---

3. Django에 대한 설명 - T/F를 판별하시오

- **프로젝트를 처음 생성 할 때, -** (`django-admin startproject <프로젝트명> .`)
  - 마지막 `.` 는 붙일 필요가 없다 (X)
  - 프로젝트 이름에는 Python, Django에서 사용 중인 키워드는 피해햐 한다 (O)
  - 프로젝트 이름에 `-(하이픈)` 은 사용 불가능하다 (O)
  - 따라서, `Django / text / class  / django-test` 는 적절치 못한 네이밍이다 (O)

- **Application명**은 복수형으로 하는 것을 권장한다 (O)

---

4. 다음의 이유에 대해 - T / F를 판별하시오

- `python manage.py startapp '앱명'` 을 통해 앱을 만든 후에, **settings.py의 Installed_apps에 생성한 앱이름을 추가해줘야** 한다.
  - 코드를 입력했어도, Django는 앱이 만들어진 것을 인식하지 못한 상태이기 때문에, 추가 작업이 필요하다 (O)
  - 앱과 프로젝트를 위의 명령어들을 통해 만들었다면, 앱과 프로젝트 파일은 동일선상에 위치하게 된다. 그러나, 엄밀히 말하면 프로젝트는 앱의 상위관계에 있어야 하기 때문에 Django에게 이를 인식시켜주는 작업이 필요하다 (O)
  - 따라서, 프로젝트에서 앱을 사용하기 위해서는 반드시 위 작업이 필요하다 (O)

---

5. settings.py의 '추가설정'에 대해 - T / F를 판별하시오

- **LANGUAGE_CODE**
  - 모든 사용자에게 제공되는 번역을 결정한다 (O)
  - 이 설정이 적용되려면 USE_I18N이 활성화되어 있어야 한다 (O)
- **TIME_ZONE**
  - USE_TZ가 False인 상태로 이 값을 설정하게 되면 error가 발생한다 (O)
- **USE_L18N**
  - Django의 번역시스템을 활성해야 하는지 여부를 결정한다 (O)
- **USE_L10N**
  - Django 데이터의 지역화된 형식을 기본적으로 활성화할지 여부를 결정한다 (O)
  - True일 경우, Django는 현재 locale의 형식을 사용하여 숫자와 날자를 표시한다 (O)

---

6. DTL(Django Template Language)에 대해 - T / F를 판별하시오

- Django에서 사용하는, 데이터 표현을 제어하는 도구이자 표현에 관련된 로직이다 (O)
- Django template에서 사용하는 built-in template system이다 (O)
- 조건, 반복, 변수 치환, 필터 등의 기능을 제공한다 (O)
- 프로그래밍적 **논리** 로직이 아니라, 프레젠테이션을 **표현**하기 위한 로직이다 (O)
- Python처럼 프로그래밍 구조(`if, for`)를 사용할 수 있으며, 이것은 해당 Python코드로 실행되는 것이다 (X)
- views에서 정의한 변수를 template파일로 넘겨 사용할때 `{ variable }` 형식을 사용한다 (O)
- `{{ variable|truncatewords:30 }}`처럼 chained가 가능하며, 일부 필터는 인자를 받기도 한다

  (O)
- 모든 태그는 시작과 종료 태그가 필요하다 (X)

---

7. Django대해 - T / F를 판별하시오

- 각 프로젝트를 진행할 때마다, 가상환경을 설치하는 것이 프로젝트 관리에 용이하다 (O)
- 각 프로젝트를 진행할 때마다, 장고를 설치해줘야 한다 (O)
- 코드작성 순서는 데이터의 흐름 순서(URL → VIEW → TEMPLATE)대로 작성하는 것이 편하다 (O)
- Django는 전반적으로 함수를 호출할 때 `()`를 안 붙이는 경우가 많은데, 파이썬 언어로 사용하긴 하지만, 파이썬 문법이 전부 적용되는 것은 아니기 때문이다 (O)

---

8. HTML Form에 대해 - T / F를 판별하시오

- 웹에서 사용자 정보를 입력하는 여러 방식(text, button, checkbox, ...)을 제공하고, 사용자로부터 할당된 데이터를 서버로 전송하는 역할을 담당한다 (O)
- 핵심 속성(attribute)으로는 action, method가 있다
  - **action** : 입력 데이터가 전송될 URL을 지정한다 (O)
  - **method** : 입력 데이터의 전달 방식을 지정한다 (O)
    - HTTP request method에는 GET, POST, PUT, DELETE가 있지만, form태그에서는 GET/POST만 사용 가능하다 (O)
    - GET method는 데이터를 가져오기만 할 수 있다 (O)

---

9. URL을 활용하는 다양한 방법에 대해 - T / F를 판별하시오

- **Variable Routing**
  - URL의 일부를 변수를 변수로 지정하여 view함수의 인자로 넘길 수 있다 (O)
  - 하나의 path에 여러 페이지를 연결시킬 수는 없다 (X)
- **URL Path Converters -** `path('hello/<str:name>/', ...),`
  - 따로 작성하지 않을 경우, str이 기본값이다 (O)
  - int, slug, uuid, path등이 있다 (O)
- **App URL mapping**

- 하나의 프로젝트에는 보통 여러 앱이 사용되며, 각 앱의 url들을 모두 프로젝트의 urls.py에서 관리하는 것은 유지보수에 좋지 않기 때문에 사용한다 (O)
- 즉, 프로젝트의 전역변수로 관리되던 url들이 앱의 지역변수로 관리된다 (O)
- include함수를 사용해야 하며, 모듈 import 명령어는 `from django.urls import include` 이다 (O)

- **Naming URL patterns**

```
path('index/', views.index, name='index'),
```

```
<a href="{% url 'index' %}">메인 페이지</a>
```

- 링크에 url을 직접 작성하는 방식이 아니라, urls.py의 path()함수에 name 인자를 정의해서 사용한다 (O)
- url 설정에 정의된 특정한 경로들의 의존성을 제거할 수 있다 (O)

---

10. DJango코드 중에서, 잘못된 부분을 고르시오

- **문제상황 : title은 생성이 되지만, content는 None값을 반환한다**

```
# views.py
def create(request):
    title = request.GET.get('title'),
    content = request.GET.get('ssafy'),
    article = Article(title=title, content=content)
    article.save()
    return render(request, 'articles/create.html')
```

```
# templates - new.html
{% extends 'base.html' %}

{% block content %}
  <h1>NEW</h1>
  <form action="{% url 'articles:create' %}" method="GET">
    <p>TITLE: <input type="title" name="title"></p>
    <p>CONTENT: <input type="content" name="content"></p>
```

```
  </form>
{% endblock content %}
```

- 답: new.html에서 `<p>TITLE: <input type="title" name="ssafy" ></p>` 로 수정해줘야 한다

-