■ 뷰

● 뷰를 구현하는 2가지 방법

- Class-based Views

  [Project]/views.py 또는 **[App]/views.py** 의 **클래스**로 작성

  기본 View 또는 generic 모듈의 View 상속

- Function Views

  [Project]/views.py 또는 **[App]/views.py** 의 **함수**로 작성

  함수의 인자로 request 명시

■ Class-based Views

- 공통적으로 사용하는 로직을 미리 클래스로 작성하여 제공
- 뷰를 잘 선택하면 빠른 개발이 가능하지만 실행되는 흐름이 어려워서 사용하지 않거나 단순한 목록 / 상세 조회 정도로 사용하는 경우가 많음

뷰 구분	이름	기능
Base View	View	최상위 뷰 (기본 뷰)
	TemplateView	템플릿을 지정하여 출력
	RedirectView	URL을 지정하여 이동
Generic Display View	**DetailView**	**단일 객체 출력**
	ListView	**다중 객체 출력**
Generic Edit View	FormView	폼 출력
	CreateView	객체 생성 폼 출력
	UpdateView	객체 수정 폼 출력
	DeleteView	객체 삭제 폼 출력
Generic Date View	ArchiveIndexView	날짜 필드를 기준으로 다중 객체 출력
	YearArchiveView	연도에 해당하는 다중 객체 출력
	MonthArchiveView	연/월에 해당하는 다중 객체 출력
	WeekArchiveView	연/주에 해당하는 다중 객체 출력
	DayArchiveView	연/월/일에 해당하는 다중 객체 출력
	TodayArchiveView	현재 날짜에 해당하는 다중 객체 출력
	DateDetailView	날짜에 해당하는 단일 객체 출력

■ Class-based Views

● [Project]/urls.py 사용 예

```
# views.py
class 뷰클래스(generic.ListView):
    template_name = 'app/oooo.html'
    context_object_name = 'name'

    def get_queryset(self):
        return 모델.objects.all()
```

```
# [Project]/urls.py
from django.urls import path
urlpatterns = [
    path('class/', views.뷰클래스.as_view()),
]
```

■ Class-based Views

● [App]/urls.py 사용 예

```
# views.py
class 뷰클래스(generic.ListView):
    template_name = 'app/oooo.html'
    context_object_name = 'name'

    def get_queryset(self):
        return 모델.objects.all()
```

```
# [Project]/urls.py
from django.urls import path, include
urlpatterns = [
    path('app/', include('app.urls')),
]
```

```
# [App]/urls.py
from django.urls import path
from . import views
urlpatterns = [
    path('class/', views.뷰클래스.as_view()),
]
```

■ Function Views

● [Project]/urls.py 사용 예

```python
# views.py
from django.http import HttpResponse

def 뷰함수(request):
    return HttpResponse('응답')
```

```python
# [Project]/urls.py
from django.urls import path
from app import views

urlpatterns = [
    path('function/', views.뷰함수),
]
```

■ Function Views

● [App]/urls.py 사용 예

```python
# views.py
from django.http import HttpResponse

def 뷰함수(request):
    return HttpResponse('응답')
```

```python
# [Project]/urls.py
from django.urls import path, include

urlpatterns = [
    path('app/', include('app.urls')),
]
```

```python
# [App]/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('function/', views.뷰함수)
]
```

■ Function Views

● 자주 사용되는 3가지 반환 타입

- HttpResponse : 단순 문자열(HTML)

```
from django.http import HttpResponse

def html(request):
    return HttpResponse('<h1>Plain Text</h1>')
```

- **render() : HTML Template**

```
from django.shortcuts import render

def template(request):
    data = {
        'str': 'text', 'num': 10,
        'list': [1, 2, 3],
        'dict': {'a': 'aaa', 'b': 'bbb'}
    }
    return render(
        request, 'template.html', context=data)
```

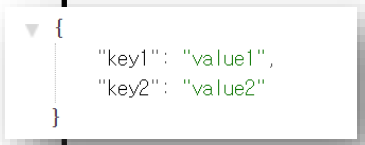
■ Function Views

● 자주 사용되는 3가지 반환 타입

- JsonResponse : JSON

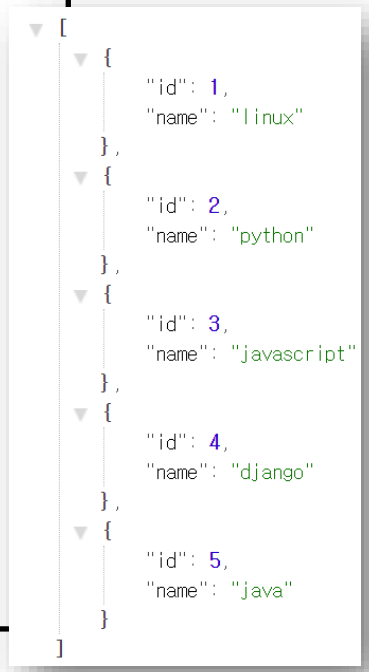
```python
from django.http import JsonResponse

# 파이썬에서 제공하는 기본 자료형태 반환
def python_data(request):
    data = { 'key1': 'value1', 'key2': 'value2' }
    # data = [ 'value1', 'value2', 'value3' ]
    return JsonResponse(data)
```




```python
from django.http import JsonResponse
from django.forms.models import model_to_dict

# 장고 모델 자료형태 반환
def model_data(request):
    curriculum = Curriculum.objects.all()
    data = []
    for c in curriculum:
        c = model_to_dict(c) # QuerySet -> Dict
        data.append(c)
    # dict가 아닌 자료는 항상 safe=False 옵션 사용
    return JsonResponse(data, safe=False)
```



■ static (static resource)

- CSS / Javascript / 이미지 / 음원 / 동영상 등 웹페이지에서 사용되는 자원
- 웹페이지를 디자인하거나 동적 기능을 넣기 위해 사용
- 정적 자원을 사용하기 위한 설정

- [Project]/settings.py

```
# 기본 주소
STATIC_URL = '/static/'

# 기본 경로인 [App]/static과 아래 경로 함께 사용
STATICFILES_DIRS = [
    BASE_DIR / 'static'
]
```

- templates/oooo.html

```
{% load static %}

<img src="/static/app/aa.jpg">

<img src="{% static 'app/aa.jpg' %}">

<img src="{% static 'bb.jpg' %}">
```

■ Templates

- 실제 View 역할
- (Django) View로부터 전달된 데이터 사용 가능
- 보통 [App]/templates/[App]/ 또는 [Project]/templates/ 위치에 HTML 파일을 생성하여 사용
- 템플릿에서 사용 가능한 태그와 문법으로 특별한 기능 제공

  - 변수, 태그, 필터, 주석, HTML 확장 등...
- 참고 문서 : https://docs.djangoproject.com/ko/3.2/ref/templates/builtins/

■ Templates

● 변수

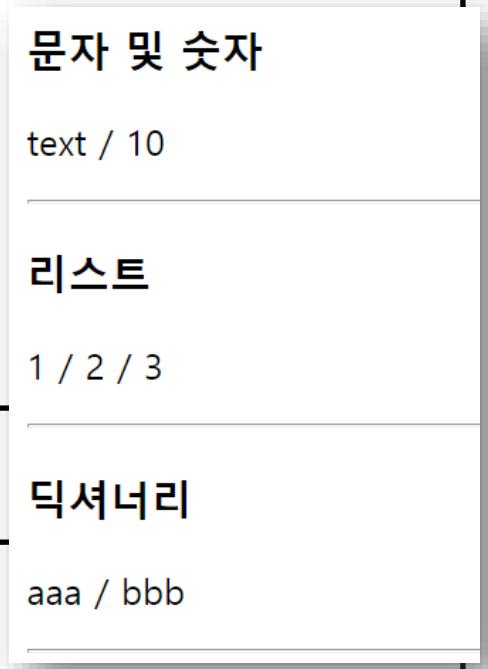
- views.py

```
from django.shortcuts import render

def template(request):
    data = {
        'str': 'text', 'num': 10,
        'list': [1, 2, 3],
        'dict': {'a': 'aaa', 'b': 'bbb'}
    }
    return render(
        request, 'firstapp/template.html', data)
```

- template.html

```
<h3>문자 및 숫자</h3>
<p>{{ str }} / {{ num }}</p><hr>
<h3>리스트</h3>
<p>{{ list.0 }} / {{ list.1 }} / {{ list.2 }}</p><hr>
<h3>딕셔너리</h3>
<p>{{ dict.a }} / {{ dict.b }}</p><hr>
```



■ Templates

● 태그

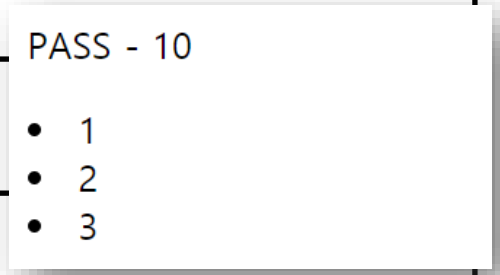
- views.py

```python
from django.shortcuts import render

def template(request):
    data = {
        'str': 'text', 'num': 10,
        'list': [1, 2, 3],
        'dict': {'a': 'aaa', 'b': 'bbb'}
    }
    return render(
        request, 'firstapp/template.html', data)
```

- template.html

```
{% if num >= 5 %} <p>PASS - {{ num }}</p>
{% else %} <p>FAIL - {{ num }}</p>
{% endif %}
<ul>
{% for i in list %}
    <li>{{ i }}</li>
{% endfor %}
</ul>
```



■ Templates

● 태그

- 일반적으로 웹 애플리케이션 제작 시 필요한 기능 3가지

① 변수 출력 {{ }}

② 반복 {% for .. in .. %}

③ 조건(분기) {% if .. %} {% elif .. %} {% else .. %}

- 반복 템플릿 안에서 사용할 수 있는 forloop

forloop 객체 속성	설명
forloop.counter	for 문의 순서로 1부터 표시
forloop.counter0	for 문의 순서로 0부터 표시
forloop.first	for 문의 첫번째 순서인 경우 True
forloop.last	for 문의 마지막 순서인 경우 True

■ Templates

● 필터

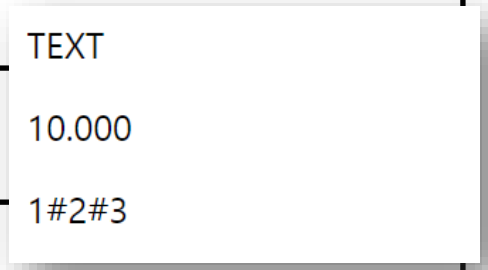
- views.py

```
from django.shortcuts import render

def template(request):
    data = {
        'str': 'text', 'num': 10,
        'list': [1, 2, 3],
        'dict': {'a': 'aaa', 'b': 'bbb'}
    }
    return render(
        request, 'firstapp/template.html', data)
```

- template.html

```
<p>{{ str|upper }}</p>
<p>{{ num|floatformat:3 }}</p>
<p>{{ list|join:'#'}}</p>
```



■ Templates

● 주석

- template.html

```
{# 한줄 주석 #}
{% comment %}
    여러줄 주석
    HTML에서 보이지 않는 부분
{% endcomment %}

<!-- HTML에서 보이는 부분 -->
```

```
1
2
3
4 <!-- HTML에서 보이는 부분 -->
```

■ Templates

● HTML 확장

- views.py

```
from django.shortcuts import render

def template(request):
    return render(
        request, 'firstapp/template.html')
```

- template.html

```
{% extends 'firstapp/layout.html' %}

{% block content %}
<form>
    아 이 디 : <input type="text"><br>
    비밀번호 : <input type="text">
</form>
{% endblock %}
```

- layout.html

```
<h1>Base Layout</h1>
<div style='background: cyan;'>
    <h1>Header</h1>
</div>
<div style='background: yellow;'>
    <h1>Content</h1>

    {% block content %}
    {% endblock %}

</div>
<div style='background: gray;'>
    <h1>Footer</h1>
</div>
```

■ Templates

● HTML 확장