The background of the slide is a photograph of an open notebook with lined pages. A yellow pencil lies diagonally across the right page, with its tip pointing towards the center. A metal pencil sharpener is positioned near the pencil's tip, and several shavings of yellow wood are scattered around it. The text is overlaid on the upper half of the image.

역설계 시각장애인을 위한 웹쇼핑몰 개발 및 연구

4조
김호진, 성유정, 최하늘

목차

1 [개요]

2 [웹서버 및 DB 연구]

3 [딥러닝을 이용한
이미지분석 연구]

4 [웹 접근성을 고려한
웹 UI/UX 연구]



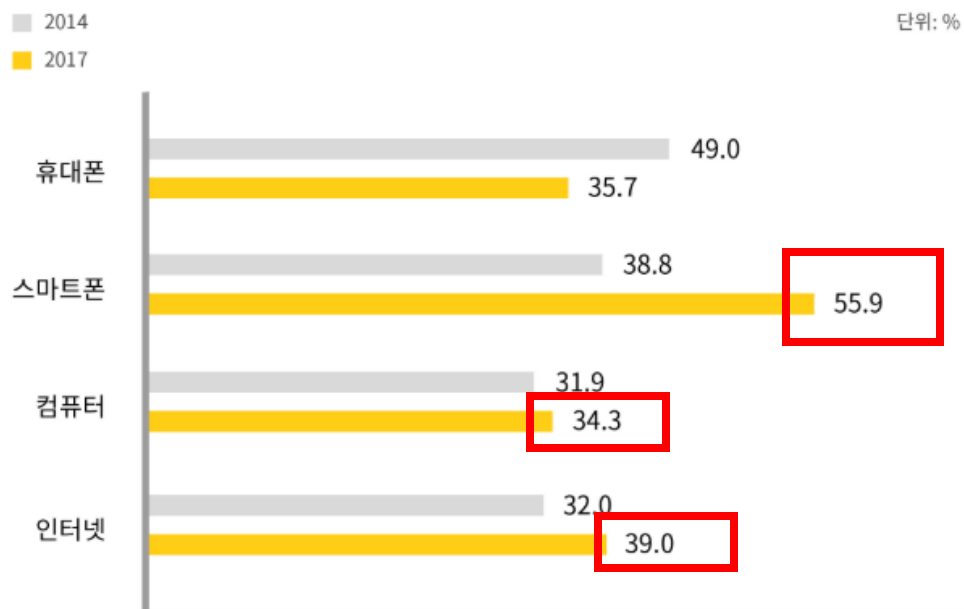
Part 1,

개요



시각장애인을 위한 웹쇼핑몰 개발 및 연구

시각장애인 정보통신기기 사용 현황



*출처_2017년 장애인실태조사

시각장애인 인터넷쇼핑 이용실태 살펴보니

장애인차별금지법에 정보 접근 차별 기준 마련해야

에이블뉴스, 기사작성일 : 2018-03-12 13:46:00

- “타인 도움 없이 혼자 온라인 쇼핑 가능하다” 시각장애인 10명 중 2명뿐

시각장애인 A씨는 “생필품을 살 때 편리하게 마트 온라인몰을 이용하고 싶는데 스크린 리더로 상세 정보를 확인할 수 없다”며 “식품을 살 때 유통기한과 영양성분 등 기본정보를 전혀 확인할 수 없다”고 호소했다.

박현규 한국웹접근성평가센터 선임연구원은 “외국의 대표적인 온라인 쇼핑몰인 아마존, 타깃은 상품 정보 대부분을 텍스트로 제공하고 있어 스크린 리더가 상품 정보를 시각장애인에게 읽어 줄 수 있다”고 설명했다. 박현규 연구원은 “그러나 국내 대형마트들은 가격 등 일부 정보를 제외하고 상세 정보는 이미지 파일에 담겨 있는 경우가 많아 시각장애인이 비장애인과 같은 정보를 받을 수 없는 게 가장 큰 문제”라고 강조했다.

문제는 여기에서 끝나지 않는다. 상품을 혼자 어렵게 찾아도 결제 과정에서 다른 사람 도움을 받을 수밖에 없다. 대형마트들의 웹 접근성 미비로 시각장애인들은 개인정보 유출 위험을 감수 하면서 남의 손을 빌려 상품을 구매하고 있었다. 지난달 한국시각장애인연합회가 전국 시각장애인 519명을 대상으로 온라인 쇼핑 접근성과 활용실태를 조사한 결과 40.8%가 홈쇼핑이나 모바일, 인터넷 쇼핑을 할 때 전적으로 타인의 도움이 필요하다고 답했다. 그만큼 온라인 쇼핑몰의 웹 접근성이 미비하다는 것이다.

Why?

시각장애인은
웹쇼핑이
어렵다.

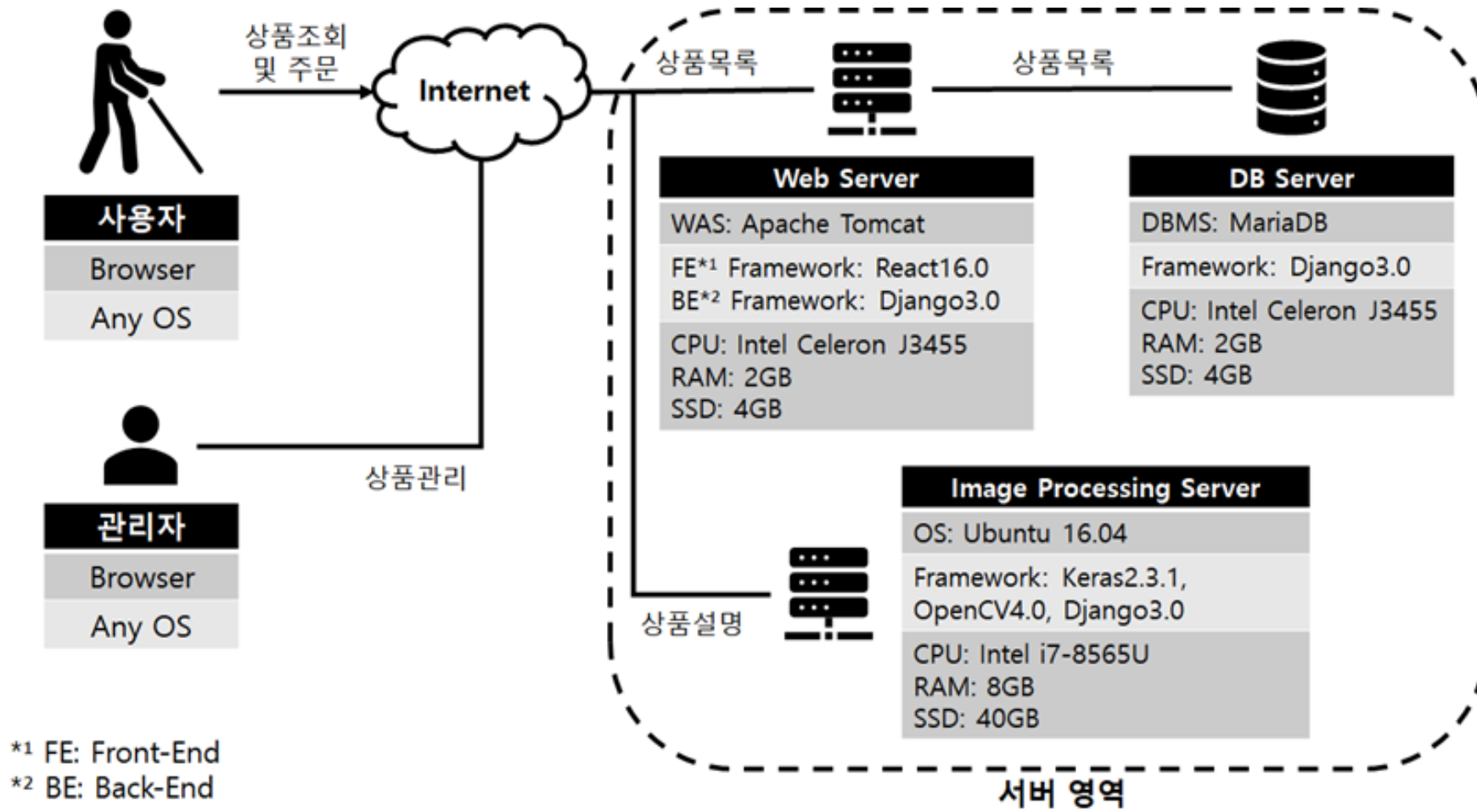
What?

시각장애인은
상품에 대한
정보를 알 수 없다.

How?

시각장애인을
웹 접근성을 고려한
쇼핑몰을 만들자.

시각장애인을 위한 웹쇼핑몰 개발 및 연구



<시스템 구조도>

시각장애인을 위한 웹쇼핑몰 개발 및 연구

1 웹 서버 및 DB 연구



Back End 개발

소프트웨어의 데이터 계층을 담당하여
서버 및 데이터베이스 서버 구축

Server / Database / API

2 딥러닝을 이용한 이미지 분석 연구



이미지 분석을 위한
딥러닝 네트워크 개발

딥러닝 네트워크/Server

3 웹 접근성을 고려한 웹 UI/UX 연구

Front End 개발

사용자가 접근하기 쉬운
웹 UI/UX 개발

UI/ UX /웹 접근성



Part 2,

웹 서버 및 **DB** 연구



웹 서버 및 DB 연구

웹 쇼핑몰 개발 순서

- ① Django 설치
- ② shop App 제작
- ③ Pillow를 활용한 image 사용환경 구축
- ④ **cart App 제작**
- ⑤ base html을 활용한 페이지 모듈화
- ⑥ Django administrator 사용
- ⑦ TTS 기능
- ⑧ 확대 축소 기능

004 >> **cart App 제작**

장바구니와 관련된 App

1. 카트에 담긴 제품수량이 변경되었을 때 확인하는 코드
2. 로그인한 유저가 담은 상품을 다른 컴퓨터에서도 유효하도록 하는 코드
3. views.py 파일 수정
4. urls.py 파일 수정
5. UI 보여주는 html 파일 생성

웹 서버 및 DB 연구

004 >> cart App 제작

1. App 만들기

>>

```
python manage.py startapp <App 이름>
```

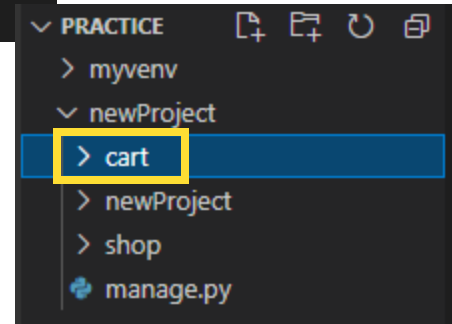
주의할 점!

manage.py가 위치한 곳에서 app을 시작해야함

```
sungy@DESKTOP-GQPQLDR MINGW64 /c/Users/sungy/Documents/Practice
$ cd newProject/

sungy@DESKTOP-GQPQLDR MINGW64 /c/Users/sungy/Documents/Practice/newProject
$ ls
manage.py  newProject  shop

sungy@DESKTOP-GQPQLDR MINGW64 /c/Users/sungy/Documents/Practice/newProject
$ python manage.py startapp cart
```

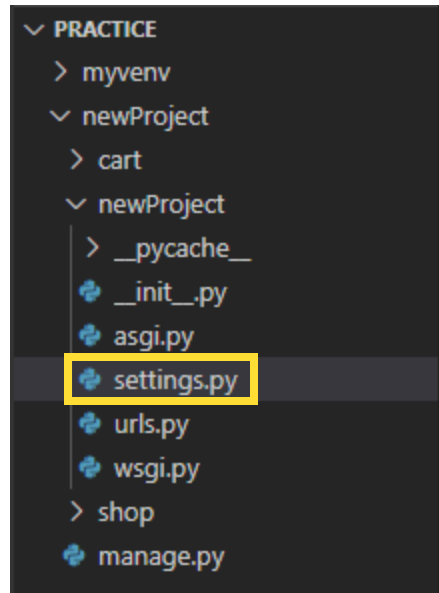


웹 서버 및 DB 연구

004 >> cart App 제작

2. App 추가하기

>> (새로 만든 프로젝트 폴더 안) → 'settings.py' 파일 → (새로 만든 App 추가)



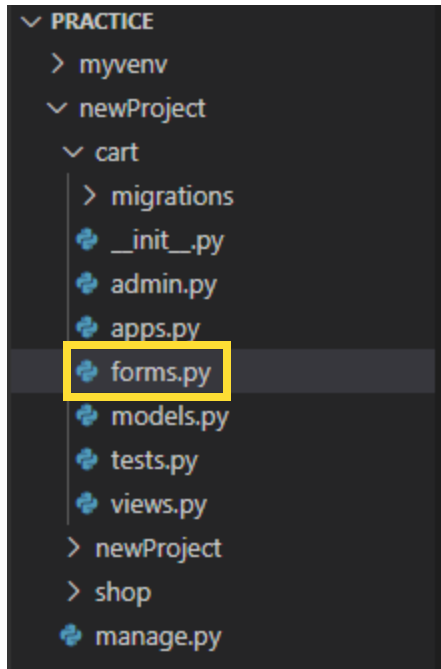
```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'shop',  
    'cart'  
]
```


웹 서버 및 DB 연구

004 >> cart App 제작

3. 'forms.py' 코드 작성 >> (새로 만든 app 폴더 안) → 'forms.py' 파일 → 코드 작성

카트에는 제품수량과 수량이 변경되었을 때를 확인하는 부분



```
newProject > cart > 📄 forms.py > ...  
1  from django import forms  
2  
3  
4  class AddProductForm(forms.Form):  
5      제품수량 = forms.IntegerField()  
6      is_update = forms.BooleanField(  
7          required=False, initial=False, widget=forms.HiddenInput  
8      )  
9
```

웹 서버 및 DB 연구

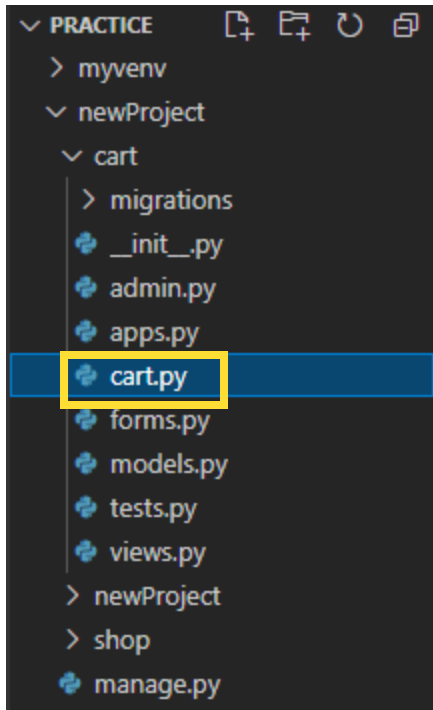
004 >> cart App 제작

4. 'cart.py' 코드 작성 >> (새로 만든 app 폴더 안) → 'cart.py' 파일 → 코드 작성

로그인 한 컴퓨터의 session을 활용하여 로그인한 유저가 담은 상품을 다른 컴퓨터에서도 유효하도록 하는 부분

```
CART_ID = "cart_in_session"

SITE_ID = 1
```



```
newProject > cart > cart.py > Cart
1  from decimal import Decimal
2  from django.conf import settings
3  from shop.models import Product
4
5  # 카트 클래스 생성
6  # 가격 총합, 정렬, 개별 수량 및 가격 체크
7  class Cart(object):
8      def __init__(self, request):
9          self.session = request.session
10         cart = self.session.get(settings.CART_ID)
11         if not cart:
12             cart = self.session[settings.CART_ID] = {}
13         self.cart = cart
14
15     def __len__(self):
16         return sum(item["quantity"] for item in self.cart.values())
17
18     def __iter__(self):
19         product_ids = self.cart.keys()
20
21         products = Product.objects.filter(id__in=product_ids)
22
23         for product in products:
24             self.cart[str(product.id)]["product"] = product
25
26         for item in self.cart.values():
27             item["price"] = Decimal(item["price"])
28             item["total_price"] = item["price"] * item["quantity"]
29
30         yield item
31
```

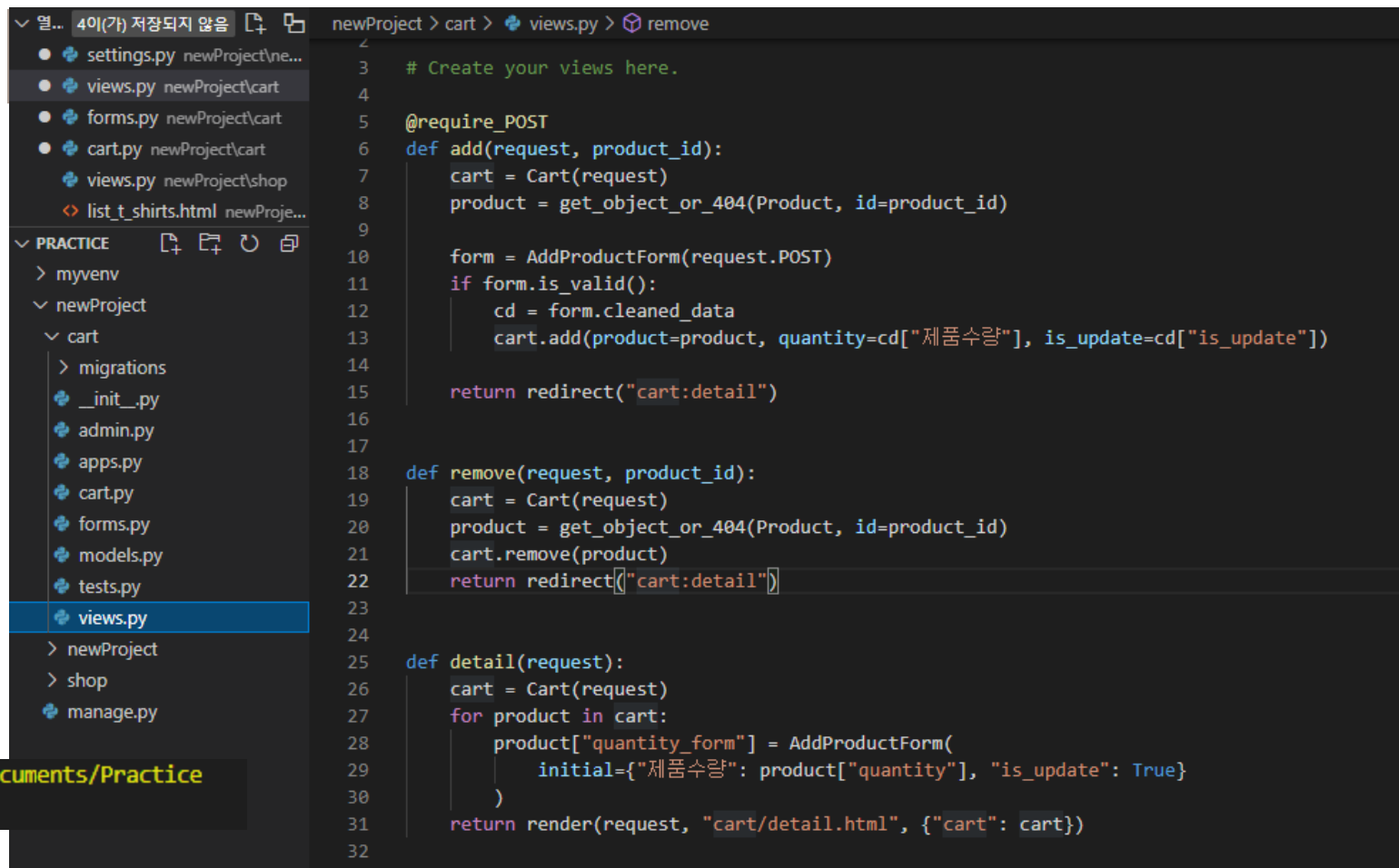
```
32 # 제품 더하기 기능
33 def add(self, product, quantity=1, is_update=False):
34     product_id = str(product.id)
35     if product_id not in self.cart:
36         self.cart[product_id] = {"quantity": 0, "price": str(product.price)}
37
38     if is_update:
39         self.cart[product_id]["quantity"] = quantity
40     else:
41         self.cart[product_id]["quantity"] += quantity
42
43     self.save()
44
45 # session을 통해 카트에 저장
46 def save(self):
47     self.session[settings.CART_ID] = self.cart
48     self.session.modified = True
49
50 # 삭제
51 def remove(self, product):
52     product_id = str(product.id)
53     if product_id in self.cart:
54         del self.cart[product_id]
55         self.save()
56
57 # session 초기화
58 def clear(self):
59     self.session[settings.CART_ID] = {}
60     self.session.modified = True
61
62 def get_product_total(self):
63     return sum(
64         Decimal(item["price"]) * item["quantity"] for item in self.cart.values()
65     )
66
```


웹 서버 및 DB 연구

004 >> cart App 제작

5. 'views.py' 코드 작성 >>

카트의 상품을 더하거나 삭제하거나
정보를 전송하는 함수를 만들어줌



```
newProject > cart > views.py > remove
1
2
3 # Create your views here.
4
5 @require_POST
6 def add(request, product_id):
7     cart = Cart(request)
8     product = get_object_or_404(Product, id=product_id)
9
10    form = AddProductForm(request.POST)
11    if form.is_valid():
12        cd = form.cleaned_data
13        cart.add(product=product, quantity=cd["제품수량"], is_update=cd["is_update"])
14
15    return redirect("cart:detail")
16
17
18 def remove(request, product_id):
19     cart = Cart(request)
20     product = get_object_or_404(Product, id=product_id)
21     cart.remove(product)
22     return redirect("cart:detail")
23
24
25 def detail(request):
26     cart = Cart(request)
27     for product in cart:
28         product["quantity_form"] = AddProductForm(
29             initial={"제품수량": product["quantity"], "is_update": True}
30         )
31     return render(request, "cart/detail.html", {"cart": cart})
32
```

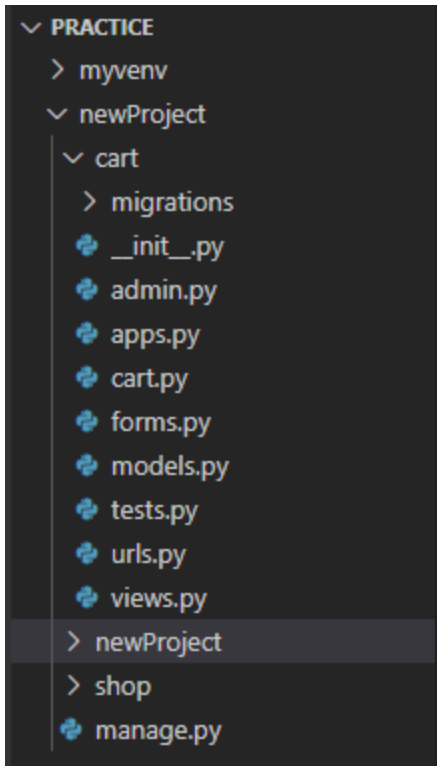
```
sungy@DESKTOP-GQPQLDR MINGW64 /c/Users/sungy/Documents/Practice
$ pip install requests
```

웹 서버 및 DB 연구

004 >> cart App 제작

6. 'urls.py' 파일 코드 작성 >>

(새로 만든 APP 폴더 안) → 'urls.py' 파일 → 코드 작성



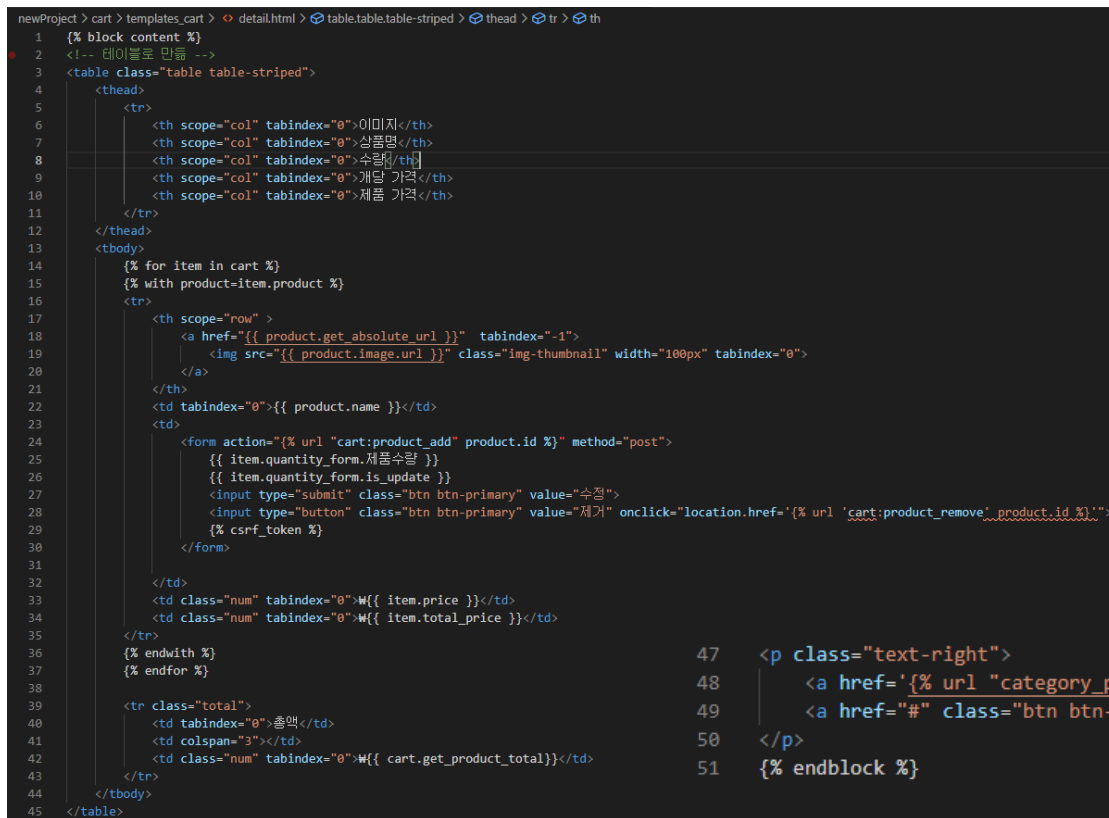
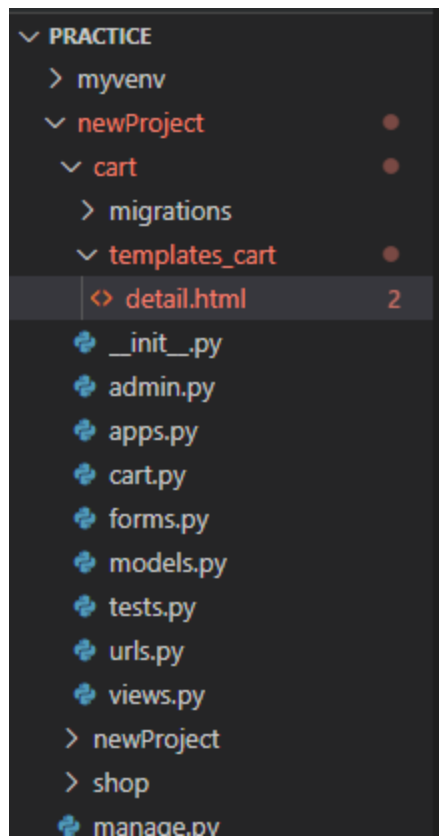
```
newProject > cart > urls.py > ...
1  from django.urls import path
2  from django.conf.urls.static import static
3  from .views import *
4
5  app_name = "cart"
6
7  urlpatterns = [
8      path("", detail, name="detail"),
9      path("add/<int:product_id>", add, name="product_add"),
10     path("remove/<product_id>", remove, name="product_remove"),
11 ]
12 |
```

웹 서버 및 DB 연구

004 >> cart App 제작

7. 담긴 카트를 보여주는 파일 만들기 >> (새로 만든 app 폴더 안) → 'templates' 폴더 생성 → html 파일 생성

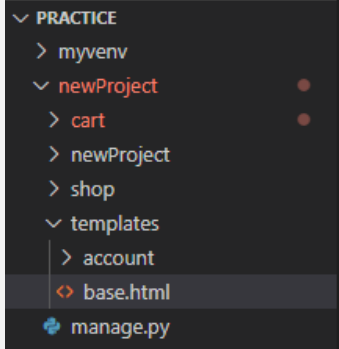
테이블의 형태로 만들 것이고 column은 이미지, 상품명, 수량, 개당 가격, 제품 가격



```
47 <p class="text-right">
48   <a href="{% url 'category_page:categories' %}" class="btn btn-secondary">계속 쇼핑하기</a>
49   <a href="#" class="btn btn-secondary">주문하기</a>
50 </p>
51 {% endblock %}
```


웹 서버 및 DB 연구

005 >> base.html을 활용한 페이지 모듈화



```
newProject > templates > base.html > html > body
1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="en">
4
5  <head>
6    <meta charset="UTF-8">
7    <title>{% block title %}{% endblock %}</title>
8    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
9      integrity="sha384-MCw98/SFmGE8fJT3GxwE0ngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdKnLPMO" crossorigin="anonymous">
10
11    <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorigin="anonymous"></script>
12    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
13      integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/8WvCWPpIm49" crossorigin="anonymous">
14    </script>
15    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
16      integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous">
17    </script>
18    <script src="http://code.jquery.com/jquery-latest.min.js"></script>
19
20    {% block script %}
21    {% endblock %}
22
23    {% block style %}
24    {% endblock %}
25
26    {% block home_style %}
27    {% endblock %}
28
29    {% block category_style %}
30    {% endblock %}
31
32    {% block detail_style %}
33    {% endblock %}
34
35  </head>
```

```
39 <body>
40 <nav class="navbar navbar-expand-lg navbar-light bg-light">
41   <a class="navbar-brand" href="/">시각장애인 쇼핑물</a>
42   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
43     aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
44     <span class="navbar-toggler-icon"></span>
45   </button>
46
47   <div class="collapse navbar-collapse justify-content-end" id="navbarSupportedContent">
48     <ul class="navbar-nav justify-content-end">
49       <li class="nav-item">
50         <!-- <a class="nav-link" href="{% url 'shop:product_all' %}">카테고리</a> -->
51         <a class="nav-link" href="{% url 'category_page:categories' %}">카테고리</a>
52       </li>
53       <li class="nav-item">
54         {% if user.is_authenticated %}
55         <a class="nav-link" href="{% url 'account_logout' %}">로그아웃</a>
56         {% else %}
57         <a class="nav-link" href="{% url 'account_login' %}">로그인</a>
58         {% endif %}
59       </li>
60       <li class="nav-item">
61         <a class="nav-link btn btn-outline-success" href="{% url 'cart:detail' %}">
62           {% if cart|length > 0 %}
63             총 {{cart|length}} 개의 물품
64           {% else %}
65             카트가 비었음
66           {% endif %}
67         </a>
68       </li>
69     </ul>
70   </div>
71 </nav>
72 <!-- <div align="center"> -->
73 <!-- <div style="width: 90%; height: 100%;" -->
74 <div class="container">
75   {% block content %}
76   {% endblock %}
77 </div>
78 </div>
79
80 {% block category_content %}
81 {% endblock %}
```

웹 서버 및 DB 연구

006

>>

django Administrator 사용

쇼핑몰주소/admin

Django administration

Site administration

ACCOUNTS

Email addresses

+ Add

✎ Change

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

✎ Change

Users

+ Add

✎ Change

SHOP

Categories

+ Add

✎ Change

Products

+ Add

✎ Change

SITES

Sites

+ Add

✎ Change

SOCIAL ACCOUNTS

Social accounts

+ Add

✎ Change

Social application tokens

+ Add

✎ Change

Django administration

WELCOME, ADMIN | [VIEW SITE](#) | [CHANGE PASSWORD](#) | [LOG OUT](#)

[Home](#) | [Shop](#) | [Categories](#) | [Add category](#)

Add category

Name:

Meta description:

Slug:

Save and add another

Save and continue editing

SAVE

웹 서버 및 DB 연구

006

>>

django Administrator 사용

쇼핑몰주소/admin

Django administration

Site administration

ACCOUNTS

Email addresses

+ Add

✎ Change

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

✎ Change

Users

+ Add

✎ Change

SHOP

Categories

+ Add

✎ Change

Products

+ Add

✎ Change

SITES

Sites

+ Add

✎ Change

SOCIAL ACCOUNTS

Social accounts

+ Add

✎ Change

Social application tokens

+ Add

✎ Change

Add product

Category:

✎ + ✖

Name:

Slug:

Image:

파일 선택 선택된 파일 없음

Description:

Meta description:

Meta description:

Price:

Stock:

☒ Display

☒ Order

Save and add another

Save and continue editing

SAVE

웹 서버 및 DB 연구

006

>>

django Administrator 사용

쇼핑몰주소/admin

Select product to change

ADD PRODUCT +

Action: Go 0 of 20 selected

<input type="checkbox"/>	NAME	SLUG	CATEGORY	PRICE	STOCK	DISPLAY	ORDER	IMAGE	CREATED	U
<input type="checkbox"/>	초록 반팔 티	green_short	의류	31000	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	products/2020/06/04/11.jpg	June 4, 2020, 5:03 a.m.	Jt
<input type="checkbox"/>	보라 색반팔 티	purple_short	의류	30000	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	products/2020/06/04/8.jpg	June 4, 2020, 5:02 a.m.	Jt
<input type="checkbox"/>	노란 반팔 티	yellow_short	의류	30000	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	products/2020/06/04/7.jpg	June 4, 2020, 5:02 a.m.	Jt
<input type="checkbox"/>	회색 반팔 티	gray_short	의류	28000	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	products/2020/06/04/4.jpg	June 4, 2020, 5 a.m.	Jt
<input type="checkbox"/>	하얀 반팔 티	white_short	의류	21000	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	products/2020/06/04/3.jpg	June 4, 2020, 4:59 a.m.	Jt
<input type="checkbox"/>	파란 반팔	blue_short	의류	20000	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	products/2020/06/04/2_IM9CT5g.jpg	June 4, 2020, 4:58 a.m.	Jt

FILTER

By Display

All
Yes
No

By created

Any date
Today
Past 7 days
This month
This year

By updated

Any date
Today
Past 7 days
This month
This year

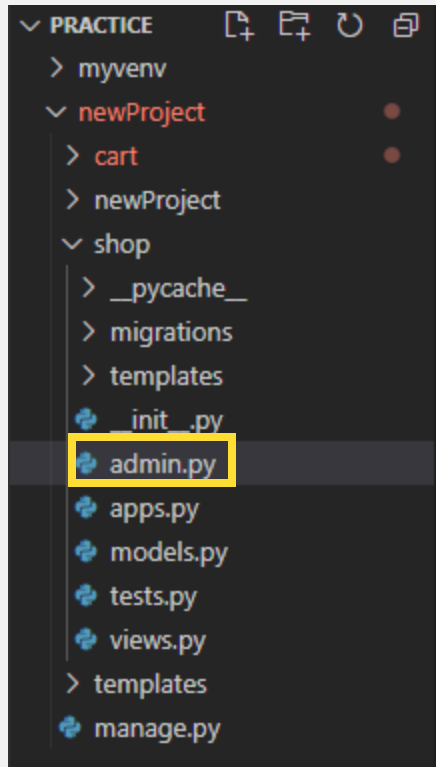
웹 서버 및 DB 연구

006

>>

django Administrator 사용

쇼핑몰주소/admin



```
newProject > shop > admin.py > ...
1  from django.contrib import admin
2
3  from .models import *
4
5
6  class CategoryAdmin(admin.ModelAdmin):
7      list_display = ["name", "slug"]
8      prepopulated_fields = {"slug": ("name",)}
9
10
11  class ProductAdmin(admin.ModelAdmin):
12      list_display = [
13          "name",
14          "slug",
15          "category",
16          "price",
17          "stock",
18          "available_display",
19          "available_order",
20          "image",
21          "created",
22          "updated",
23      ]
24      list_filter = ["available_display", "created", "updated", "category"]
25      list_editable = ["price", "stock", "available_display", "available_order"]
26      prepopulated_fields = {"slug": ("name",)}
27
28
29  admin.site.register(Category, CategoryAdmin)
30  admin.site.register(Product, ProductAdmin)
31
```

웹 서버 및 DB 연구

007 >> TTS 기능

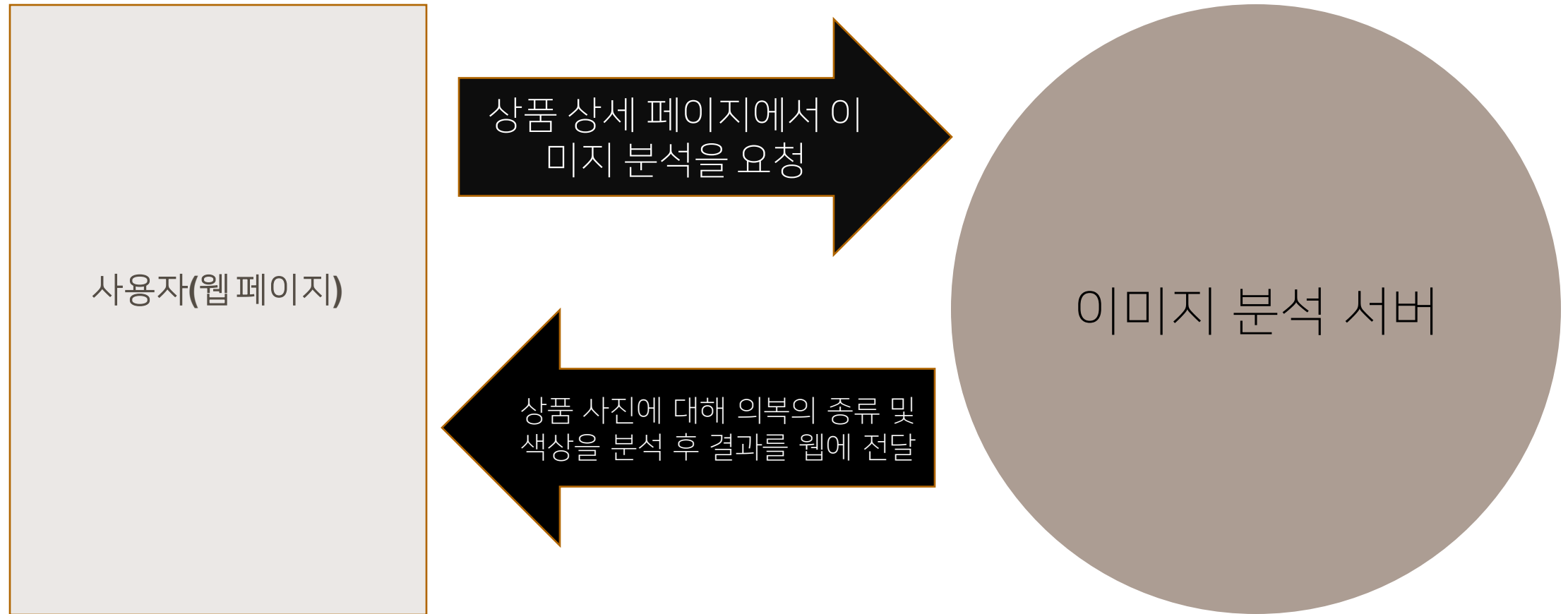
008 >> 확대, 축소 기능

Part 3,

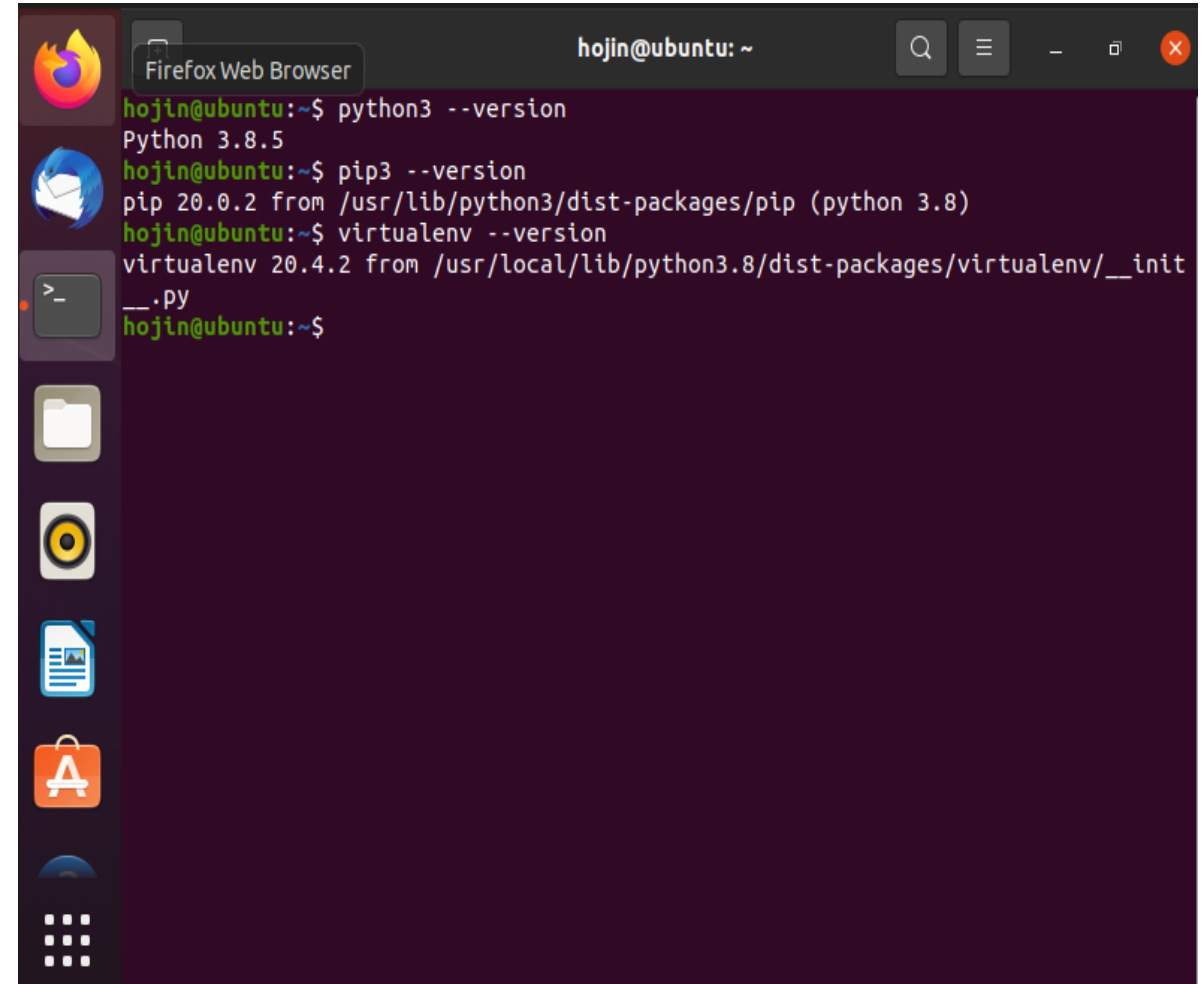
딥러닝을 이용한
이미지 분석 연구



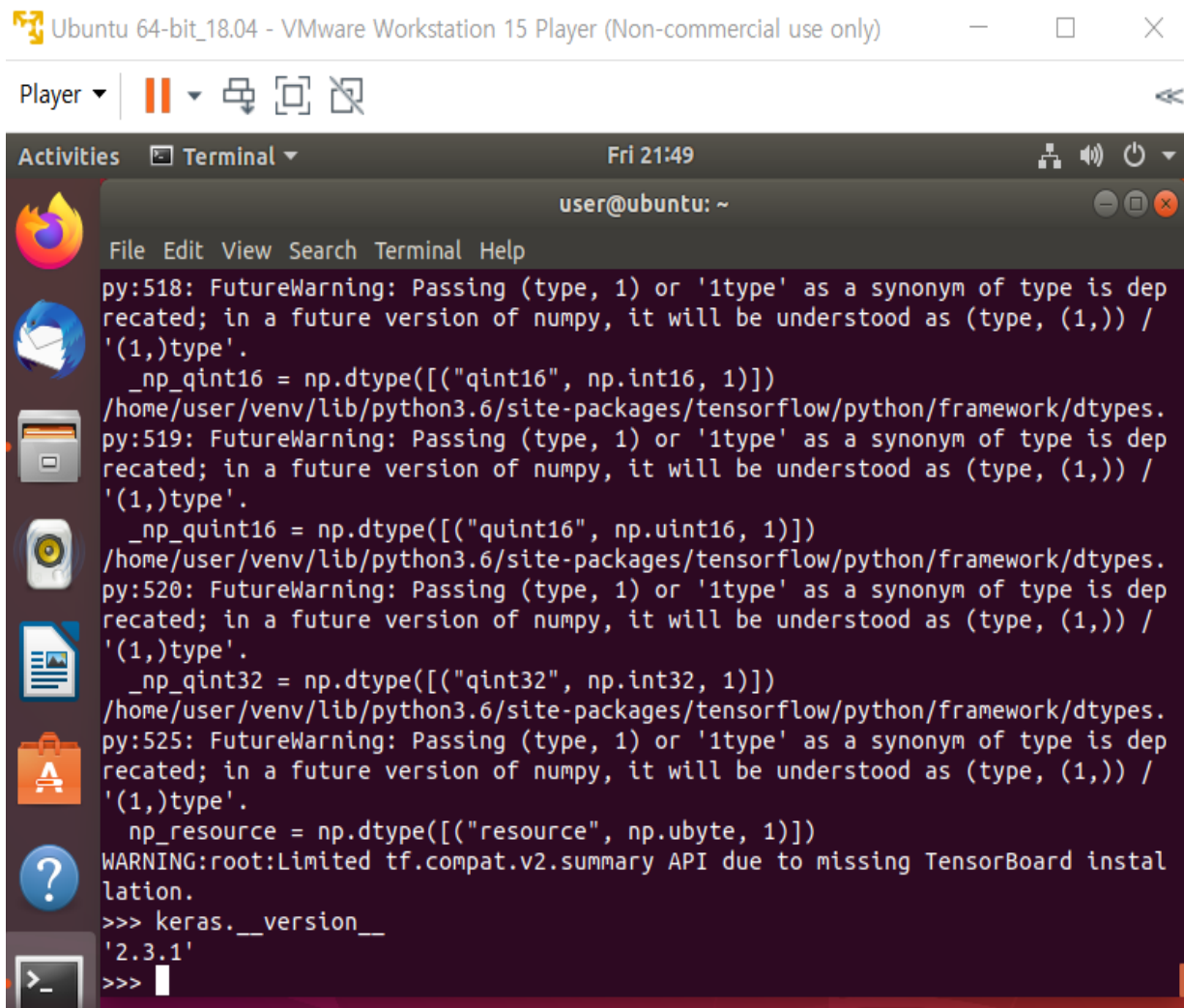
딥러닝을 이용한 이미지 분석 연구



이미지 분석 서버



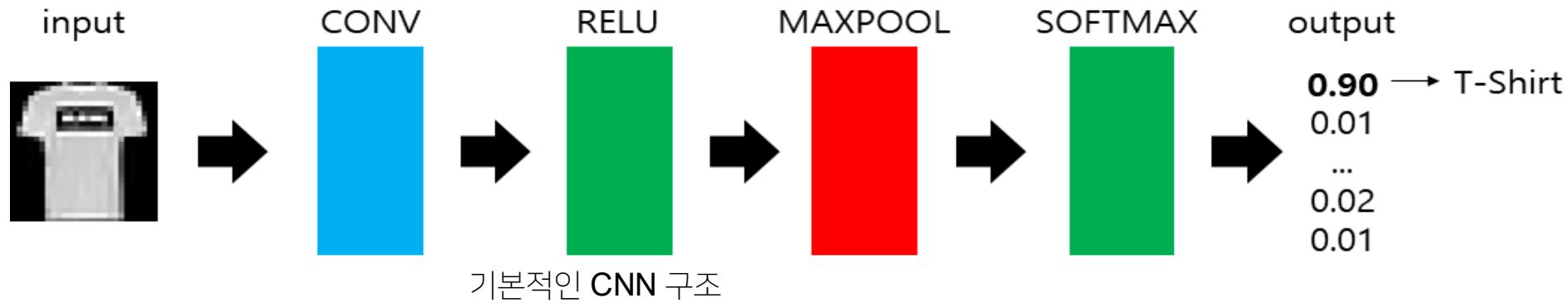
이미지 분석 서버



```
user@ubuntu: ~  
File Edit View Search Terminal Help  
py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
_np_qint16 = np.dtype(["qint16", np.int16, 1])  
/home/user/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.  
py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
_np_quint16 = np.dtype(["quint16", np.uint16, 1])  
/home/user/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.  
py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
_np_qint32 = np.dtype(["qint32", np.int32, 1])  
/home/user/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.  
py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.  
np_resource = np.dtype(["resource", np.ubyte, 1])  
WARNING:root:Limited tf.compat.v2.summary API due to missing TensorBoard installation.  
>>> keras.__version__  
'2.3.1'  
>>>
```

Tensor Flow 설치
↓
Keras 프레임워크 설치

딥러닝 네트워크 구현 - 의복 분류(CNN구조)



CNN
: Convolutional
Neural
Networks의
약자

Input Image
5×5

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter (Kernel)
3×3

1	0	1
0	1	0
1	0	1

• $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$.

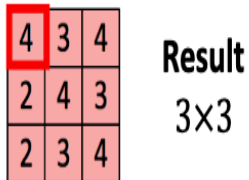
• Inner product:

$$\langle \mathbf{A}, \mathbf{B} \rangle = \sum_i \sum_j a_{ij} b_{ij} = 70.$$

딥러닝 네트워크 구현 - 의복 분류(CNN구조)



Convolution:



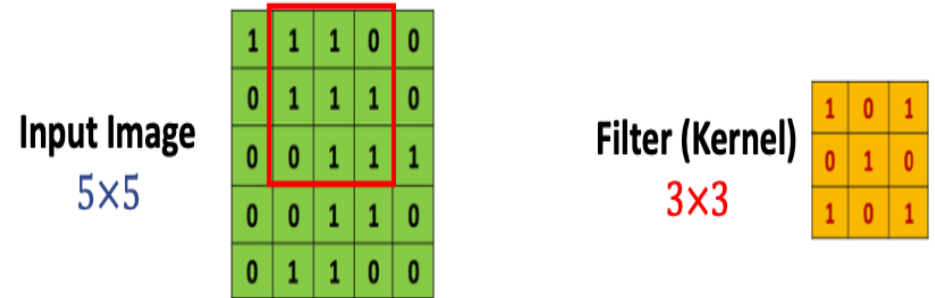
The value **4** is the inner product of the patch

1	1	1
0	1	1
0	0	1

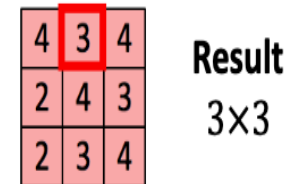
 and the filter

1	0	1
0	1	0
1	0	1

.



Convolution:



The value **3** is the inner product of the patch

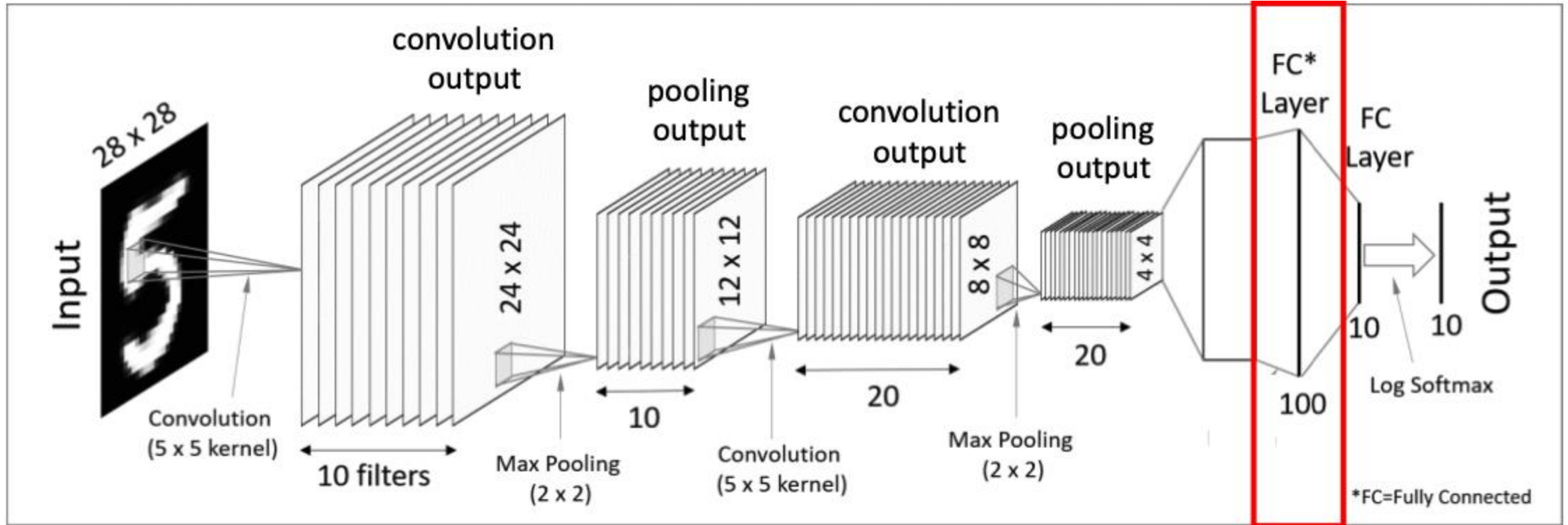
1	1	0
1	1	1
0	1	1

 and the filter

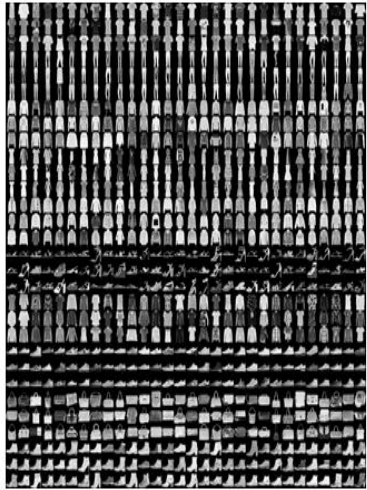
1	0	1
0	1	0
1	0	1

.

딥러닝 네트워크 구현 - 의복 분류(CNN구조)



딥러닝 네트워크 구현 - 의복 분류(코드)



Labels	
Each training and test example is assigned to one of the following labels:	
Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Fashion-MNIST

: 10가지 종류와 70,000개의 흑백 이미지로 구성된 의류에 대한 데이터셋

먼저 기본적인 패키지를 import(임포트)한다.

```
from __future__ import absolute_import, division, print_function, unicode_literals, unicode_literals

import tensorflow as tf
from tensorflow import keras

import numpy as np
import matplotlib.pyplot as plt
```

그 후 'Fashion-MNIST' 데이터셋을 가져오는 부분을 함수로 작성한다.

```
# load train and test dataset
def load_dataset():
    # load dataset
    (trainX, trainY), (testX, testY) = fashion_mnist.load_data()

    # reshape dataset to have a single channel
    trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
    testX = testX.reshape((testX.shape[0], 28, 28, 1))

    # one hot encode target values
    trainY = to_categorical(trainY)
    testY = to_categorical(testY)
    return trainX, trainY, testX, testY
```

딥러닝 네트워크 학습을 위해 데이터셋 이미지의 픽셀 값이 0에서 1사이가 되도록 정규화하는 부분을 함수로 작성한다.

```
# scale pixels
def prep_pixels(train, test):
    # convert from integers to floats
    train_norm = train.astype('float32')
    test_norm = test.astype('float32')

    # normalize to range 0-1
    train_norm = train_norm / 255.0
    test_norm = test_norm / 255.0

    # return normalized images
    return train_norm, test_norm
```

딥러닝 네트워크 구현 - 의복 분류(코드)

다음은 딥러닝 네트워크를 모델링한다. 앞서 그림으로 나타냈던대로 층을 만들어 모델에 추가한다.
가장 먼저 CONV 계층을 만들고 활성화함수는 RELU로 하여 추가한다. 모든 계층은 RELU와 He초기화를 적용한다.
그 다음 MAXPOOL 계층을 추가한다. 그 다음 SOFTMAX 계층을 추가한다.

```
# define cnn model
def define_model():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))

    # compile model
    opt = SGD(lr=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```

```
# run the test harness for evaluating a model
def run_test_harness():
    # load dataset
    trainX, trainY, testX, testY = load_dataset()

    # prepare pixel data
    trainX, testX = prep_pixels(trainX, testX)

    # define model
    model = define_model()

    # fit model
    model.fit(trainX, trainY, epochs=10, batch_size=32, verbose=0)

    # save model
    model.save('models/final_model.h5')
    # evaluate model on test dataset
    _, acc = model.evaluate(testX, testY, verbose=0)
    print("> %.3f %% (acc * 100.0))
```

다음은 지금까지 작성했던 함수들을 호출하여
딥러닝 학습을 수행하는 부분이다.

딥러닝 네트워크 구현 - 의복 분류(코드)

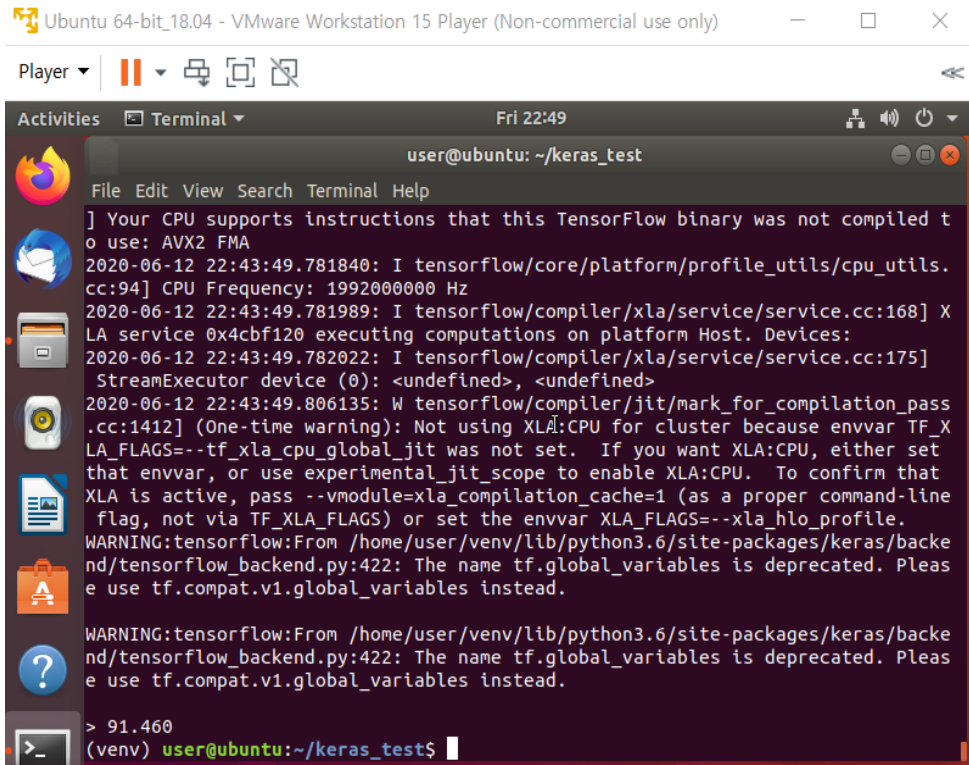
마지막으로 다음 코드를 추가함으로써 딥러닝 학습을 위한 `train.py` 작성을 완료한다.

```
# entry point, run the test harness
run_test_harness()
```

그리고 `models` 폴더를 생성하여 학습된 모델이 저장될 수 있도록 한다

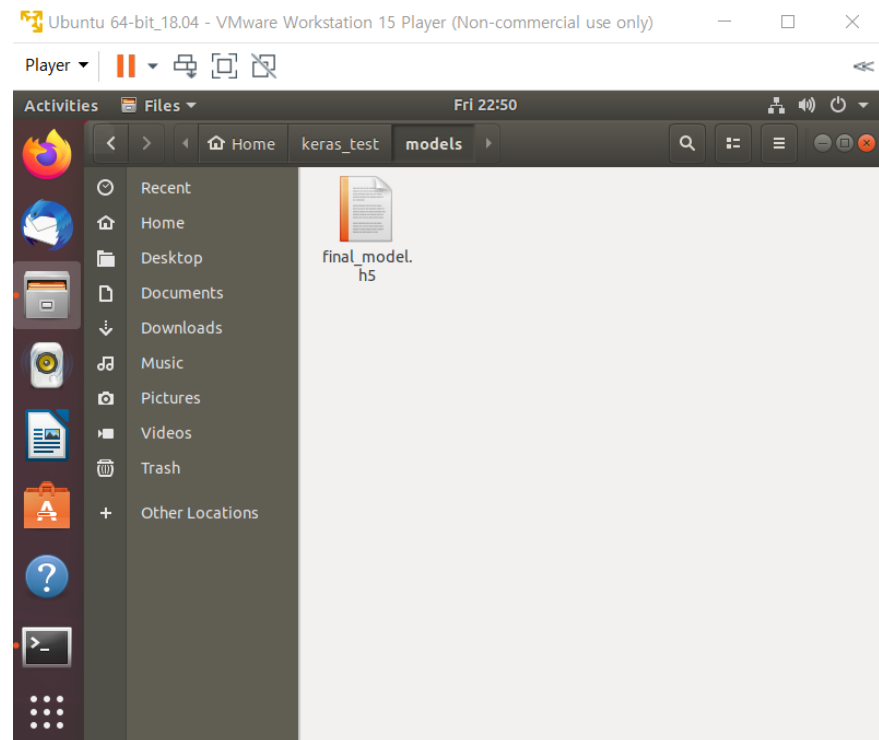
다음 명령어를 입력하여 학습을 시작한다.

```
python train.py
```



Terminal window showing the execution of `python train.py`. The output includes TensorFlow logs indicating that the binary was not compiled for AVX2 FMA, CPU frequency is 1992000000 Hz, and a warning about XLA compilation. The final output shows the accuracy: `> 91.460`.

학습이 완료 된 모습



딥러닝 모델 생성

딥러닝 네트워크 구현 - 의복 분류(테스트)

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
from matplotlib import pyplot
import PIL.ImageOps

# Image Path
IMAGE_PATH = 'images/0.jpg'
# class_names = ['T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat',
#                 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
class_names = ['티셔츠', '바지', '긴팔티', '드레스', '코트',
               '샌들', '셔츠', '스니커즈', '가방', '구두']

# load an image and predict the class
def run_example(self):

    anl = Analyze()
    # load the image
    img, img2 = anl.load_image(IMAGE_PATH)
    # load model
    model = load_model('models/final_model.h5')
    # predict the class
    result = model.predict_classes(img2)
    print(class_names[result[0]])

# load and prepare the image
def load_image(self, filename):
    # load the image
    img = load_img(filename, grayscale=True, target_size=(28, 28))

    # invert img
    img2 = PIL.ImageOps.invert(img)
    # convert to array
    img2 = img_to_array(img2)
    # reshape into a single sample with 1 channel
    img2 = img2.reshape(1, 28, 28, 1)
    # prepare pixel data
    img2 = img2.astype('float32')
    img2 = img2 / 255.0
    return img, img2

python main.py
```

먼저 필요한 패키지를 임포트한다.

다음은 테스트할 이미지 파일 경로와 클래스를 지정한
본 문서에서는 클래스를 한글로 지정하였다. 인덱스
순서만 동일하면 상관없다.

다음은 이미지를 가져와서 전처리한다. 이를
load_image 함수로 작성한다.

다음은 의복 분류를 진행하는 함수를 작성한다.

다음 명령어로 딥러닝 모델 테스트를 시작한다.

딥러닝 네트워크 구현 - 의복 분류(테스트)

Ubuntu 64-bit_18.04 - VMware Workstation 15 Player (Non-commercial use only)

Player ▾ | [Icons] [Icons] [Icons]

Activities [Terminal] ▾ Fri 23:01 [Icons]

user@ubuntu: ~/keras_test

File Edit View Search Terminal Help

```
] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2020-06-12 22:58:55.837568: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 1992000000 Hz
2020-06-12 22:58:55.837728: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x2e56b90 executing computations on platform Host. Devices:
2020-06-12 22:58:55.837747: I tensorflow/compiler/xla/service/service.cc:175] StreamExecutor device (0): <undefined>, <undefined>
2020-06-12 22:58:55.847045: W tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412] (One-time warning): Not using XLA:CPU for cluster because envvar TF_XLA_FLAGS=--tf_xla_cpu_global_jit was not set. If you want XLA:CPU, either set that envvar, or use experimental_jit_scope to enable XLA:CPU. To confirm that XLA is active, pass --vmodule=xla_compilation_cache=1 (as a proper command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=--xla_hlo_profile.
WARNING:tensorflow:From /home/user/venv/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
WARNING:tensorflow:From /home/user/venv/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
```

티셔츠
(venv) user@ubuntu:~/keras_test\$

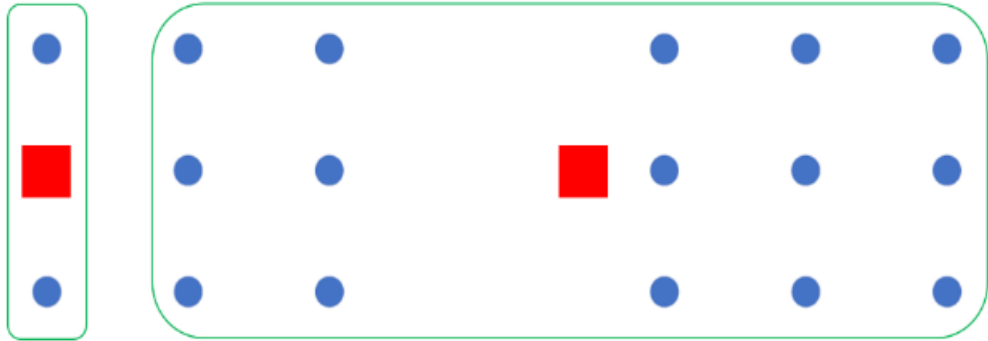
티셔츠 사진을 넣고 확인 한 결과

파이썬을 이용한 색상 판단 - 색 분포 파악



군집의 중심 초기화

군집 수(K)를 2로 놓고, 군집의 중심을 랜덤 초기화



Maximization 스텝

중심을 군집 경계에 맞게 갱신



Expectation 스텝

모든 개체들을 가장 가까운 중심의 군집으로 할당

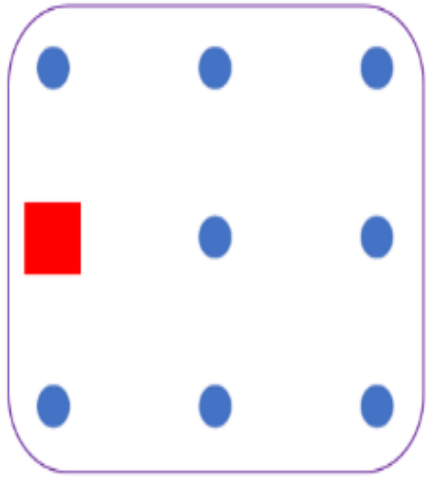
$$X = C_1 \cup C_2 \dots \cup C_K, \quad C_i \cap C_j = \phi$$

$$\operatorname{argmin}_C \sum_{i=1}^K \sum_{x_j \in C_i} \|x_j - c_i\|^2$$

K-평균 군집화 수식

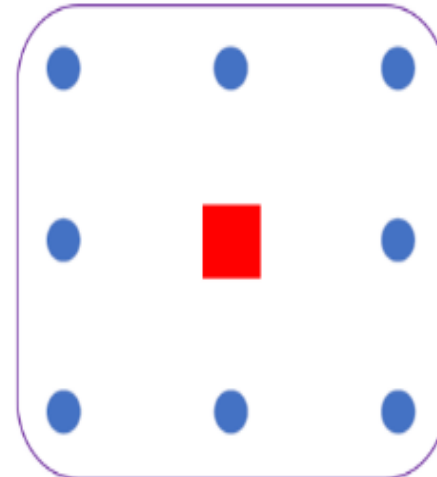
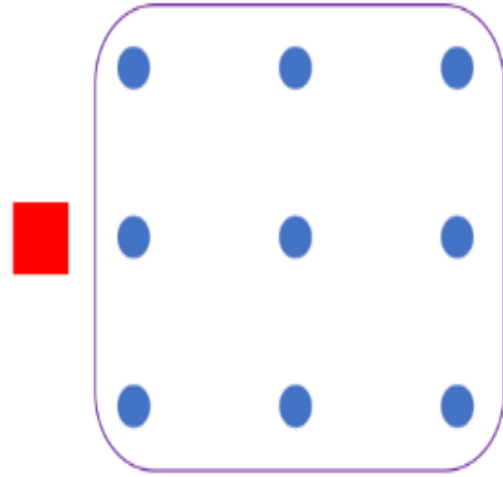
K-Means Clustering(K-평균 군집화)
를 사용하여 색 분포 파악
EM 알고리즘을 기반으로 작동
Expectation 스텝
Maximization 스텝

파이썬을 이용한 색상 판단 - 색 분포 파악



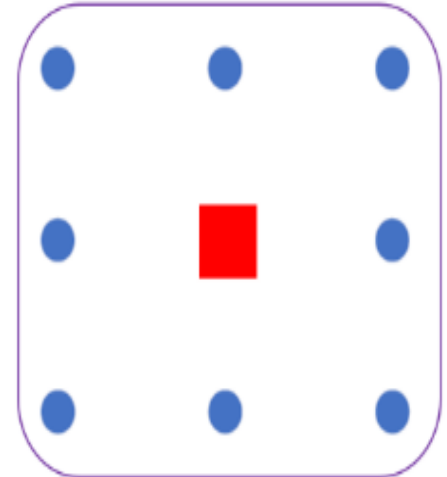
Expectation 스텝 재적용

다시 개체들을 가장 가까운 중심에
군집으로 할당해주는 작업



알고리즘 적용 결과

다시 **Maximization** 스텝을 적용하여
중심을 갱신



파이썬을 이용한 색상 판단 - 색 분포 파악(코드)

지금까지 설명한 k-평균 군집화를 파이썬으로 구현한다. 'color_recog-clustering.py'로 작성한다. 먼저 필요한 패키지를 import(임포트)한다.

```
import numpy as np
import cv2
import matplotlib.image as mpimg
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from webcolors import rgb_to_name
```

파이썬 가상환경에 각종 모듈을 설치했다면 가상환경을 활성화시켜야 한다.

```
source ./env/bin/activate
```

색 분포를 알아낼 이미지 파일 경로를 지정한다.

```
image_path = "images/0.jpg"
```

다음은 이미지 내 색상들을 히스토그램으로 변환하는 'centroid_histogram' 함수를 작성한다.

```
def plot_colors(self, hist, centroids):
    # initialize the bar chart representing the relative frequency
    # of each of the colors
    bar = np.zeros((50, 300, 3), dtype = "uint8")
    startX = 0

    # loop over the percentage of each cluster and the color of
    # each cluster
    for (percent, color) in zip(hist, centroids):
        endX = startX + (percent * 300)
        cv2.rectangle(bar, (int(startX), 0), (int(endX), 50), color.astype("uint8").to
list(), -1)
        startX = endX

    # return the bar chart
    return bar
```


파이썬을 이용한 색상 판단 - 색 분포 파악(코드)

다음은 위의 함수들을 호출하여 샘플이미지의 색 분포를 확인한다. 아래 명령어를 추가로 작성한다.

```
clrRecog = ColorRecog()

image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = image.reshape((image.shape[0] * image.shape[1], 3))

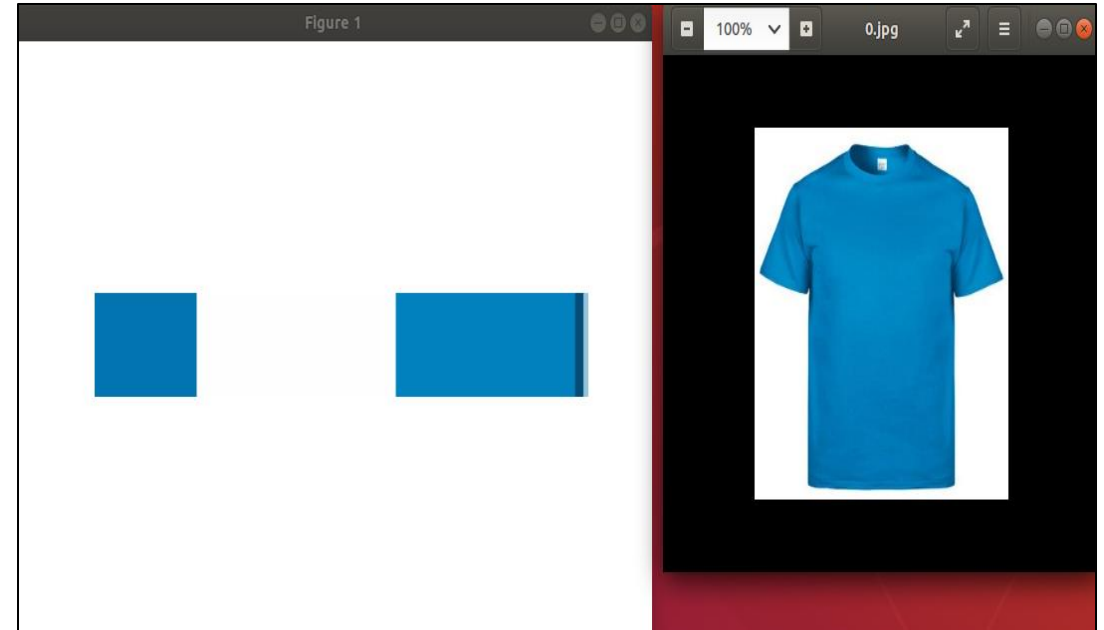
clt = KMeans(n_clusters = k)
clt.fit(image)

hist = clrRecog.centroid_histogram(clt)
bar = clrRecog.plot_colors(hist, clt.cluster_centers_)

plt.figure()
plt.axis("off")
plt.imshow(bar)
plt.show()
```

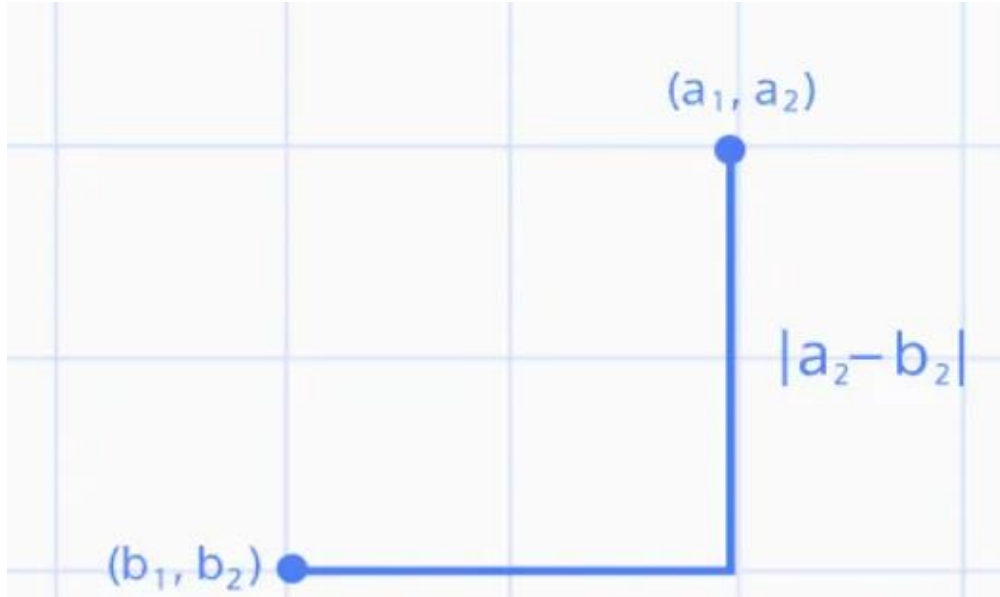
다음의 명령어로 프로그램을 실행한다.

```
python color_recog-clustering.py
```



실행 결과

파이썬을 이용한 색상 판단 - 색상 값 비교



Manhattan Distance (맨하탄 거리)
를 사용하여 색 분포 중 가장 비율이
높은 색상을 알아내고, 이 색상 값을
RGB 색상표와 비교하여 어떤 색상
인지 판단

$$d = |a_1 - b_1| + |a_2 - b_2|$$

Manhattan Distance(맨하탄 거리)

: 두 점 사이의 거리를 구하는 방법
거리가 가까울수록 유사도가 높다

파이썬을 이용한 색상 판단 - 색상 값 비교(코드)

먼저 구분할 색상들을 지정한다.

```
color_class = ['하얀', '검정', '빨간', '초록', '파란', '노란', '분홍색', '민트색', "갈색", "보라색", "주황색", "연두색", "진한회색", "연한회색"]
```

```
def plot_colors(self, hist, centroids):
    clrRecog = ColorRecog()

    # initialize the bar chart representing the relative frequency
    # of each of the colors
    bar = np.zeros((50, 300, 3), dtype="uint8")
    startX = 0
    first_color = -1
    second_color = -1
    color_name = ""
    # loop over the percentage of each cluster and the color of
    # each cluster
    for (percent, color) in zip(hist, centroids):

        ## Most Color
        if first_color < percent:

            first_color = percent
            first_color_list = color

            red = round(first_color_list[0])
            green = round(first_color_list[1])
            blue = round(first_color_list[2])

            first_color_list_int = (red, green, blue)

        ## Second-most Color
        elif second_color < first_color and second_color < percent:
            second_color = percent
            second_color_list = color

            red = round(second_color_list[0])
            green = round(second_color_list[1])
            blue = round(second_color_list[2])
            second_color_list_int = (red, green, blue)

        # plot the relative percentage of each cluster
        endX = startX + (percent * 300)
        cv2.rectangle(bar, (int(startX), 0), (int(endX), 50),
                      color.astype("uint8").tolist(), -1)
        startX = endX

    ## classify color
    for i in range(len(color_class)):
        if i == 0:
            for i in range(len(color_class)):
                if clrRecog.classify_color(second_color_list_int) == color_class[i]:
                    color_name = color_class[i]

            elif clrRecog.classify_color(first_color_list_int) == color_class[i]:
                color_name = color_class[i]

    # return the bar chart
    return bar, color_name
```

다음으로 앞서 작성했던 `plot_color` 함수를 수정한다. 가장 많은 색과 그 다음으로 많은 색을 구분하도록 한다. 이는 바탕색이 흰색인 경우와 이미지 내 사물의 색상이 동일하게 흰색인 경우를 고려하였다.

파이썬을 이용한 색상 판단 - 색상 값 비교(코드)

다음은 기준 색상을 지정하고, 이미지 내의 사물 색상과 비교하는 `classify_color` 함수를 작성한다.

```
def classify_color(self, rgb_tuple):
    colors = { "하얀": (255,255,255),
               "검정": (0,0,0),
               "빨간": (255,0,0),
               "초록": (0,255,0),
               "파란": (0,0,255),
               "노란": (255,255,0),
               "분홍색": (255,0,255),
               "민트색": (0,255,255),
               "갈색": (128,64,0),
               "보라색": (128,0,128),
               "보라색": (114,77,188),
               "주황색": (255,128,0),
               "연두색": (0,255,64),
               "진한회색": (128,128,128),
               "연한회색": (192,192,192),
            }
    manhattan = lambda x,y : abs(x[0] - y[0]) + abs(x[1] - y[1]) + abs(x[2] - y[2])
    distances = {k: manhattan(v, rgb_tuple) for k, v in colors.items()}
    color = min(distances, key=distances.get)

    return color
```

앞서 맨하탄 거리의 개념을 설명할 때 언급했던 수식을 구현한 것이다. 기준 색상의 RGB값을 지정하고 k-평균 군집화를 통해 알아낸 색상과 비교함으로써 사물의 색상을 판단할 수 있다. 다음은 선언한 함수들을 호출하여 사용하는 부분을 작성한다.

```
def image_color_cluster(self, image_path, k = 5):
    clrRecog = ColorRecog()

    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = image.reshape((image.shape[0] * image.shape[1], 3))

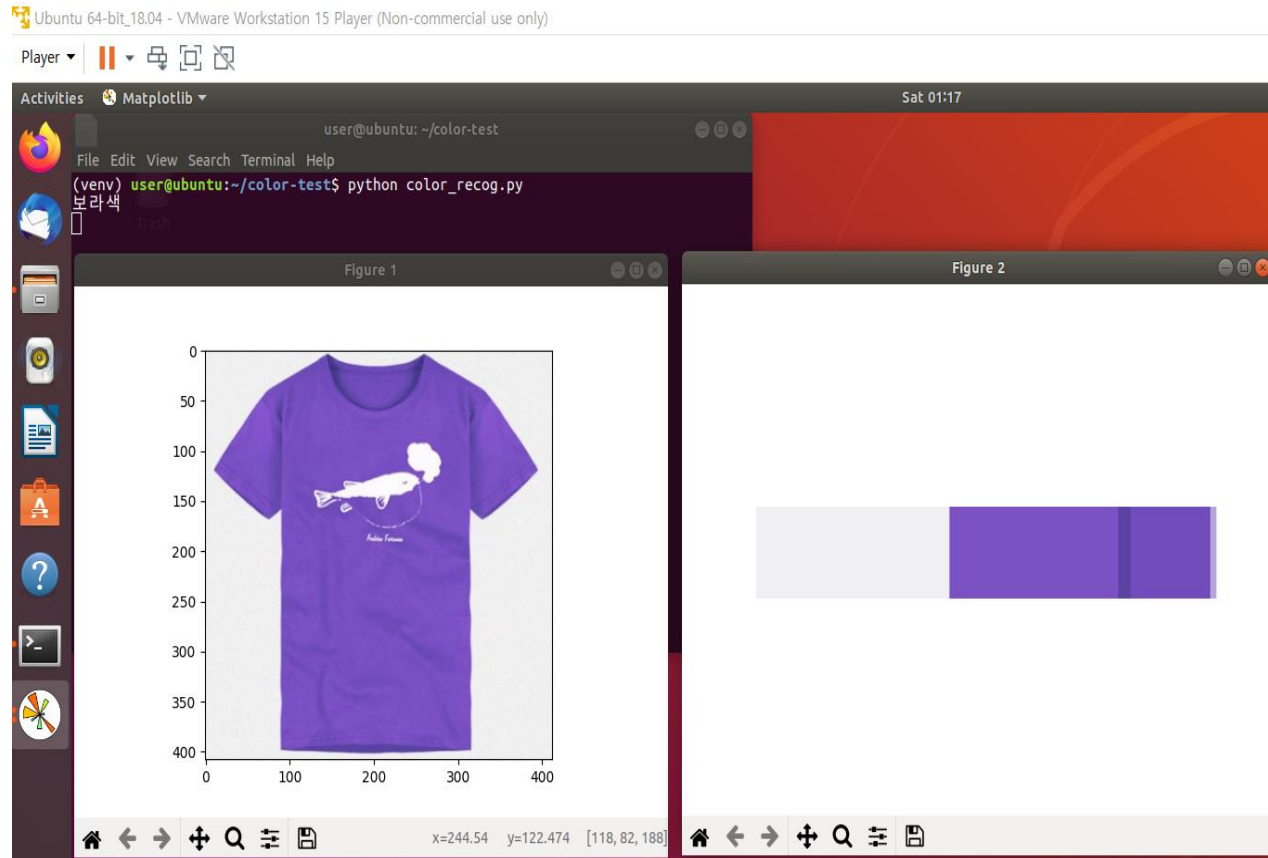
    clt = KMeans(n_clusters = k)
    clt.fit(image)
    hist = clrRecog.centroid_histogram(clt)
    bar, color_name = clrRecog.plot_colors(hist, clt.cluster_centers_)
    print(color_name)

    plt.figure()
    plt.axis("off")
    plt.imshow(bar)
    plt.show()
    return color_name
```

파이썬을 이용한 색상 판단 - 색상 값 비교(코드)

다음 명령어를 입력하여 실행한다.

```
python color_recog.py
```



실행 결과

Part 4,

웹 접근성을 고려한
웹 UI/UX 연구



웹 쇼핑몰 UI 구성

시각장애인 쇼핑몰

카테고리

로그아웃

장바구니가 비었음

```
<nav>
```

```
  <a tabindex="0"> 시각장애인 쇼핑몰 </a>
```

```
  <div>
```

```
    <ul>
```

```
      <li>
```

```
        <a tabindex="0"> 카테고리 </a>
```

```
        <a tabindex="0"> 로그아웃 </a>
```

```
        <a tabindex="0"> 장바구니가 비었음 </a>
```

```
      </li>
```

```
    </ul>
```

```
  </div>
```

```
</nav>
```

웹 쇼핑몰 UI 구성

```
<div>
  <div>
    <div>
      <font tabindex="0"> 당신을 위한 추천 상품 </font>
      <div>
        <div>
          <div>
            <font tabindex="0"> 순위 </font>
            <div>
              <a tabindex="-1">
                <img tabindex="0">
              </a>
            </div>
          </div>
          <div>
            <span tabindex="0">상품</span>
            <span tabindex="0">가격</span>
          </div>
        </div>
      </div>
      <!-- 나머지 두 이미지 부분 생략 -->
    </div>
  </div>
</div>
```

웹 쇼핑몰 UI 구성

```
<div>
  <div>
    <button tabindex="0"> 대분류 </button>
    <button tabindex="0"> 남성의류 </button>
    <button tabindex="0"> 여성의류 </button>
    <button tabindex="0"> 생필품 </button>
    <button tabindex="0"> 전자기기 </button>
    <button tabindex="0"> 식품 </button>
    <button tabindex="0"> 시각장애인 용품 </button>
  </div>
  <!-- 중분류, 소분류도 대분류와 구조가 비슷하여 생략 -->
</div>
```

대분류

남성의류

중분류

모자

소분류

바탕 티셔츠

웹 쇼핑몰 UI 구성

```
<div>
  <div>
    <div>
      <font tabindex="0"> 남성의류 / 상의 / 반팔티셔츠 </font>
    </div>
    <div>
      <div>
        <a tabindex="-1">
          <img tabindex="0">
        </a>
      </div>
      <div>
        <span tabindex="0"> 상품명 </span>
        <span tabindex="0"> 가격 </span>
      </div>
    </div>
    <!-- 나머지 상품은 생략 -->
  </div>
</div>
```


웹 쇼핑몰 UI 구성

```
<div>
  <div>
    <div>
      <form>
        <label tabindex="0"> 제품수량 </label>
        <input tabindex="0"> ... </input>
        <input tabindex="0"> 장바구니 넣기 </input>
      </form>
    </div>
  </div>
</div>
```

초록반팔티

이름

상세설명

웹 쇼핑몰 UI 구성

```
<div>  
    <table>  
        <tbody>  
            <tr>  
                <tb>  
                    <a tabindex="-1">  
                        <img tabindex="0">  
                    </a>  
                </tb>  
                <tb tabindex="0"> 초록 반팔티 </tb>  
                <tb>  
                    <form>  
                        <input tabindex="0"> 수량입력 </input>  
                        <input tabindex="0"> 수정 </input>  
                        <input tabindex="0"> 제거 </input>  
                    </form>  
                </tb>  
                <tb tabindex="0"> 개당가격 </tb>  
                <tb tabindex="0"> 제품가격 </tb>  
            </tr>  
            <tr>  
                <tb tabindex="0"> 총액 </tb>  
                <tb tabindex="0"> 계산된 총액 </tb>  
            </tr>  
        </tbody>  
    </table>  
    <p>  
        <a tabindex="0"> 계속 쇼핑하기 </a>  
        <a tabindex="0"> 주문하기 </a>  
    </p>  
</div>
```

A row of yellow pencils with a central yellow bar. The pencils are arranged in a row, with their tips pointing upwards. The central pencil is replaced by a solid yellow rectangular bar. The entire scene is set against a solid yellow background.

감사합니다