**과목명: 기초빅데이터프로그래밍**

**STS2011-02**

**<<기말프로젝트>>**

**서강대학교 [공학부 컴퓨터공학과]**

**[20171630]**

**[남주형]**

목 차

1. **프로그램 개요**
2. **프로그램 설명**
   1. main 기능
   2. 1번 기능
   3. 2번 기능
   4. 3번 기능
   5. 4번 기능
   6. 0번 기능
3. **클래스 정의**
   1. ChoiceError
   2. DateError
   3. Item
   4. have\_expiration\_date
   5. refrigerator
   6. warehouse
4. **함수 정의**
   1. checkTime
   2. direct
   3. search\_brcd
   4. select1
   5. select2
   6. select3
   7. select4
   8. select5
5. **추가 기능**
   1. file read
   2. menu & file store
6. **코드**
7. **프로젝트를 진행하며 느낀 점**

**1. 프로그램 개요**

이 프로그램은 사용자의 식품이나 생필품 등 소모품을 관리하기 위한 프로그램이다. 특히 식품에서 유통기한을 체크하는 기능에 초점을 맞추어 제작하였다. 냉장고에 유통기한이 지나 음식을 그냥 버리는 경험을 다들 해보았을 것이다. 이 프로그램을 이용하여 유통기한을 체크하여 식품의 낭비와 음식물 쓰레기를 획기적으로 줄일 수 있을 것이다. 또한 소모성 생필품을 관리하며 필요한 물건에 대한 파악도 할 수 있게 해준다. 더 나아가 이 프로그램이 더 발전된다면 편의점과 같은 상점의 재고 관리에도 사용할 수 있을 것이다.

**2. 프로그램 설명**

**2.1 main 기능**

메뉴를 보여준다.

front에서는 보이지 않지만 먼저 txt파일에 저장된 이전에 저장한 제품들의 정보를 읽어서 자료구조에 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

프로그램을 실행하면 처음 화면으로 다음과 같이 출력한다. 원하는 메뉴를 선택하면 해당하는 기능이 실행된다.

**2.2 1번 기능**

상품을 등록한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1번을 입력하면 위와 같이 직접 입력할 것인지 바코드를 입력할 것인지 선택을 할 수 있다.

먼저 직접입력을 골랐을 경우를 살펴보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같이 제품명을 입력하는 칸이 나온다.

제품명을 입력하면 카테고리를 입력하는 칸도 뒤이어 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 위와같이 유통기한을 입력할지 안할지 고른다. 여기서 1번을 선택하면 유통기한을 입력하고 최종적으로 제품을 등록하고 2번을 선택하면 유통기한을 입력하지 않고 최종적으로 제품을 등록한다.(여기선 마스크를 등록하여 아니오를 선택하겠다.)

텍스트이(가) 표시된 사진

자동 생성된 설명

그럼 다음과 같이 실행시간을 측정해주고 처음나왔던 메뉴를 다시 보여준다.

텍스트이(가) 표시된 사진

자동 생성된 설명두번째로 바코드를 입력했을 경우를 살펴보자.

여기서 예시로 입려하는 상품은 오리온 초코다이제 2000이다.

바코드 번호는 8801117221003이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

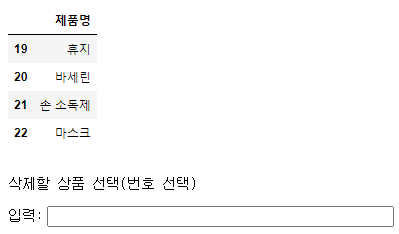
위와같이 상품등록에서 바코드입력을 선택하고 바코드를 입력하면 제품명과 카테고리가 자동으로 크롤링되어 출력된다. 그리고 유통기한 입력을 하기위해 ‘네’를 선택하고 유통기한을 입력하면 최종적으로 제품이 등록된다.

이때 유통기한을 입력하면 ‘냉장고’에 상품이 들어가고 유통기한을 등록하지 않으면 ‘창고’에 상품이 들어간다.

**2.3 2번 기능**

상품을 제거한다.





메뉴에서 2번을 선택하면 다음과 같이 먼저 가지고 있는 상품을 나열해서 보여준다. 여기서 제거하고 싶은 제품의 index를 입력하면 해당 제품을 제거해준다.

**2.4 3번 기능**

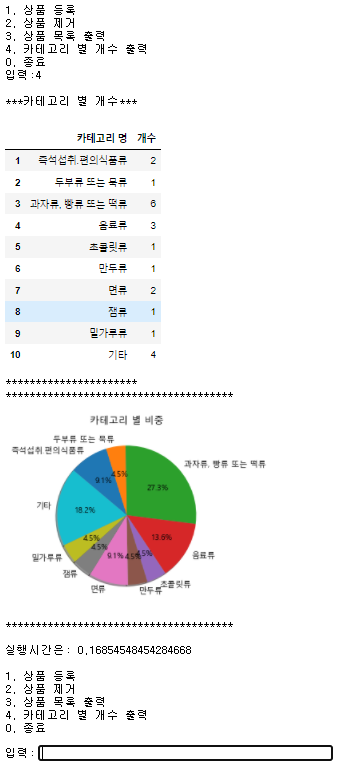
상품 목록을 냉장고에 있는 물품과 창고에 있는 물품을 구분하여 출력해준다.

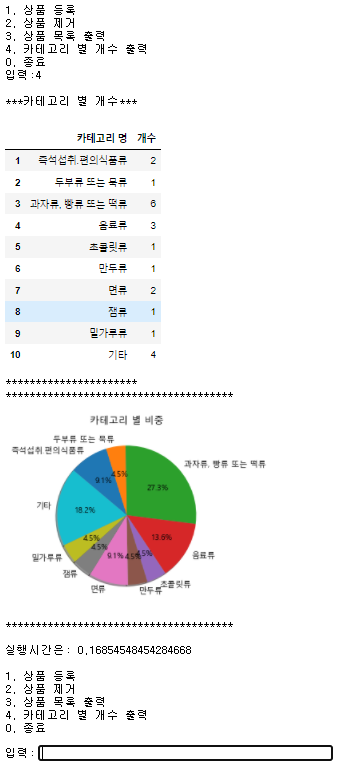
테이블이(가) 표시된 사진

자동 생성된 설명

**2.5 4번 기능**

카테고리별 개수를 출력해주고 카테고리별 차지하는 비중을 pie 차트를 통해 나타내어주었다.

해당 기능을 통해 카테고리에 해당하는 제품들이 얼마나 남아있는지 파악 할 수 있다.



**2.6 5번 기능**

남은 유통기한에 따른 재고를 파악한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 전체적인 유통기한 범위에 따라 결과를 도출할지 직접 유통기한 남은 일수를 입력하여 결과를 도출할지 선택한다.

먼저 전체적인 유통기한 범위에 따른 재고 파악에 대해서 살펴보자.

아래와 같이 유통기한이 지난 제품과 0~3일, 4~7일, 8~30일, 31일 이상의 범위에 따라 나누어 먼저 제품명과 남은 일 수를 보여준다.

텍스트이(가) 표시된 사진

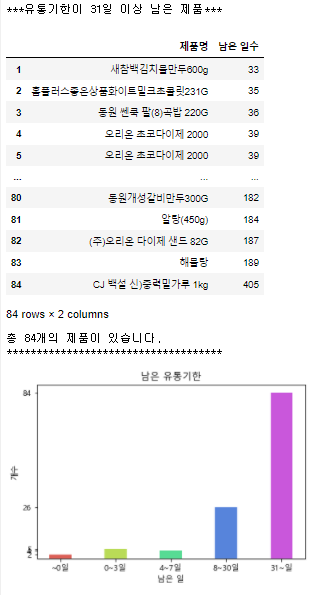
자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명



마지막으로 위와같이 범위에따른 재고량을 그래프로 표현해주었다.

두번째로 직접 선택한 유통기한에 따른 재고 파악에 대하여 살펴보자.

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

예시로 유통기한이 10일 이내인 제품명을 출력해주었다.

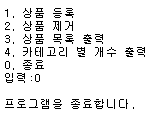
테이블이(가) 표시된 사진

자동 생성된 설명

**2.7 0번 기능**

프로그램을 종료한다.

front에서는 보이지 않지만 자료구조에 저장된 데이터들을 txt파일에 형식에 맞추어 저장한다.



다음과 같이 출력하며 종료한다.

**3. 클래스 정의**

**3.1 ChoiceError**

텍스트이(가) 표시된 사진

자동 생성된 설명

select2에서 상품을 제거할 때 존재하지 않는 상품을 제거할 때 발생하는 error이다.

**3.2 DateError**



유통기한을 입력받을 때 ‘YYYYMMDD’의 형식이 아니면 발생하는 error이다.

**3.3 Item**

텍스트이(가) 표시된 사진

자동 생성된 설명

제품의 이름과 카테고리를 저장하는 class이다.

**3.4 have\_expiration\_date**

텍스트이(가) 표시된 사진

자동 생성된 설명

유통기한이 있는 제품에 대한 정보를 저장하는 class이다. Item class를 상속받았다.

함수로는 유통기한 날짜를 현재 날짜와 비교하여 남은 유통기한 일수를 계산해주는 compute\_date 함수가 있다.

**3.5 refrigerator**

테이블이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명



텍스트이(가) 표시된 사진

자동 생성된 설명

냉장고에 해당하는 class로 유통기한이 있는 제품들을 보관하고있다.

put\_item은 냉장고에 item을 넣음과 동시에 남은 유통기한 일로 정렬해주는 함수이다.

remove\_item은 해당하는 index의 제품을 삭제하는 함수이다.

print\_item은 냉장고에 있는 item을 dataframe을 이용하여 표로 정리하여 보여주는 함수이다.

comput\_category는 제품의 카테고리 별 개수를 dictionary를 통하여 계산하여 저장하는 함수이다.

print\_day\_info는 입력받은 남은 일수내에 있는 제품을 dataframe을 이용하여 표로 정리하여 보여주는 함수이다.

print\_data\_info\_total은 기간을 나누어 해당하는 기간의 제품을 표로 정리하여 보여주고 개수를 프린트해준다. 또한 numpy와 matplotlib.pyplot을 이용하여 bar 차트로 보여준다.

**3.6 warehouse**

텍스트이(가) 표시된 사진

자동 생성된 설명

창고에 해당하는 class로 유통기한이 없는 제품들을 보관하고있다.

함수의 기능은 **refrigerator**의 bar\_chart 함수를 제외하면 **refrigerator** 함수의 기능과 같다.

**4. 함수 정의**

**4.1 checkTime**

텍스트이(가) 표시된 사진

자동 생성된 설명

원하는 함수의 실행 시간을 출력해주는 함수이다. select1, select2, select3, select4의 실행시간을 출력해준다.

**4.2 direct**

텍스트이(가) 표시된 사진

자동 생성된 설명

select1에서 제품을 직접 입력할 때 사용하는 함수이다. 먼저 제품명과 카테고리를 입력받고 유통기한을 입력길 원하는지 물어본다. 그리고 그에 대한 대답을 입력받을 때 정규식을 사용하여 에러체크를 하여 예외처리를 해준다. 또 유통기한을 입력받을 시 올바른 형식의 date를 입력했는지 체크하고 존재하는 날짜인지도 체크하여 예외처리를 해준다. 모두 올바른 데이터가 들어왔다면 유통기한 입력 여부에따라 item을 have\_expiration\_date또는 Item class로 생서해준다.

그리고 item을 flag와 함께 retrun해준다.

**4.3 search\_brcd**

텍스트이(가) 표시된 사진

자동 생성된 설명

select1에서 제품을 바코드를 통해 입력할 때 사용하는 함수이다.

먼저 바코드를 입력받는다. 이때 바코드는 13자리 숫자이므로 정규식을 이용하여 예외처리를 해준다. 올바른 data가 입력되었다면 ‘식품의약품안전처 데이터 활용서비스’에서 제공하는 유통바코드에대한 api를 이용하여 바코드에 등록되어있는 제품명과 카테고리 정보를 크롤링한다. 이때 제공하는 db에 해당 바코드 데이터가 등록되어있지않다면 직접입력할 수 있도록 한다. 바코드 정보를 찾았다면 제품명과 카테고리를 출력하고 유통기한을 등록하여 앞에서 했던 방식과 같이 예외처리를 해주고 해당 데이터를 return해준다.

**4.4 select1**

텍스트이(가) 표시된 사진

자동 생성된 설명

menu에서 1번을 선택했을 시 호출되는 함수이다.

먼저 직접입력할지 바코드를 입력할지 선택하게 하고 예외처리 역시 진행해준다. 그리고 직접입력을 선택했다면 direct함수를 호출하여 item과 flag에 대한 정보를 return받는다. 그리고 해당하는 정보를 flag에 따라 refrigerator.put\_item, warehouse.put\_itme을 통해 냉장고나 창고에 넣어준다.

**4.5 select2**

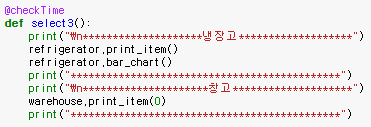
텍스트이(가) 표시된 사진

자동 생성된 설명

menu에서 2번을 선택했을 시 호출되는 함수이다.

먼저 상품목록을 출력해준다. 이때 refrigerator.print\_item을 통해 냉장고의 제품을 나열하고 그 개수를 return받아 warehouse.print\_item에 넣어주어 해당 인덱스다음 인덱스부터 창고의 제품을 나열해준다. 이때 n2가 0이면 제품이 없는 것이므로 메시지를 띄어주고 return해준다. 상품이 존재한다면 삭제할 상품의 index를 입력받는다. 이때 해당하는 범위에따라 냉장고 또는 창고에서 remove\_item함수를 통해 제품을 제거해준다. 이때 존재하지 않는 index를 입력할 시 ChoiceError를 일으켜 예외처리를 해준다.

**4.6 select3**



menu에서 2번을 선택했을 시 호출되는 함수이다.

refrigerator.print\_item을 통해 냉장고에 있는 제품을 보여주고 refrigerator.bar\_chart를 통해 냉장고에 있는 제품의 유통기한 일수에 따른 막대그래프를 보여준다. 그리고 warehouse.print\_item을 통해 창고에 있는 제품을 보여준다.

**4.7 select4**

텍스트이(가) 표시된 사진

자동 생성된 설명

menu에서 2번을 선택했을 시 호출되는 함수이다.

dictionary에 comput\_categroy함수를 이용해 category 별 개수를 구해서 넣어준다. 그리고 dictionary에서 그 정보를 cate와 num에 넣어 dataset에 zip하여 넣어준다. 이 정보를 dataframe을 이용하여 표로 만들어 display해준다.

다음으로 matplotlib.pyplot를 이용하여 카테고리별 차지하는 비중을 pie chart를 show해준다.

**4.8 select5**

텍스트이(가) 표시된 사진

자동 생성된 설명

남은 유통기한 일에 따른 재고를 파악한다.

먼저 전체 또는 직접 입력할지 고르고 직접입력한다면 refrigerator.print\_date\_info\_total함수를 통해 전체 범위에 대해 나누어서 출력해준다.

직접입력을 고르면 일수를 입력받아 refrigerator.print\_day\_info를 통해 해당 유통기한 내의 제품명을 출력해준다.

**5 추가 기능**

**5.1 file read**

텍스트이(가) 표시된 사진

자동 생성된 설명

item.txt 파일을 열어서 저장된 데이터들을 읽어 냉장고와 창고에 알맞게 넣어준다.

**5.2 menu & file store**

텍스트이(가) 표시된 사진

자동 생성된 설명

menu를 while문을 돌면서 계속 출력해준다. 해당하는 메뉴의 기능을 실행하고 0이 선택되면 item.txt파일을 열어서 자료구조에 저장된 데이터를 txt파일에 저장해준다. 그리고 break를 통해 while문을 빠져나가 프로그램을 저장해준다.

**6. 코드**

import datetime as dt

import time

import xml.etree.ElementTree as ET

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import re

import sys

import platform

if platform.system() == 'Darwin': #맥

plt.rc('font', family='AppleGothic')

elif platform.system() == 'Windows': #윈도우

plt.rc('font', family='Malgun Gothic')

elif platform.system() == 'Linux': #리눅스 (구글 콜랩)

plt.rc('font', family='Malgun Gothic')

plt.rcParams['axes.unicode\_minus'] = False

if sys.version\_info[0] == 3:

from urllib.request import urlopen

else:

from urllib import urlopen

def checkTime(func):

def new\_func(\*args, \*\*kwargs):

start = time.time();

func(\*args, \*\*kwargs)

print("\n실행시간은:",time.time()-start)

return new\_func

class ChoiceError(Exception):

def \_\_init\_\_(self, msg):

self.\_message = msg

class DateError(Exception):

def \_\_init\_\_(self, msg):

self.\_message = msg

class Item(object):

def \_\_init\_\_(self,name,category):

self.\_name=name

self.\_category=category

@property

def name(self):

return self.\_name

@property

def category(self):

return self.\_category

class have\_expiration\_date(Item):

def \_\_init\_\_(self,name,category,date):

Item.\_\_init\_\_(self,name,category)

self.\_date=date

@property

def name(self):

return self.\_name

@property

def category(self):

return self.\_category

@property

def date(self):

return self.\_date

def compute\_date(self):

now = dt.datetime.now()

remain\_date=self.date - now

return remain\_date.days

class refrigerator(object):

def \_\_init\_\_(self):

self.\_item=[]

@property

def item(self):

return self.\_item

def put\_item(self,item):

self.\_item.append(item)

self.\_item.sort(key = lambda x : x.date)

def remove\_item(self,n):

item = self.\_item[n-1]

name = item.name

self.\_item.remove(item)

print(name+"이(가) 삭제되었습니다.")

def print\_item(self):

num = 0

name=[]

date=[]

for item in self.\_item:

num+=1

name.append(item.name)

date.append(str(item.compute\_date()))

dataset=list(zip(name,date))

refr=pd.DataFrame(data=dataset, columns=['제품명','남은 일수'])

refr.index=refr.index+1

display(refr)

return num

def comput\_category(self,dic):

for item in self.\_item:

category = item.category

dic[category]=dic.get(category,0)+1

def print\_day\_info(self,n):

num=0

name=[]

date=[]

for item in self.\_item:

day=item.compute\_date()

if(day>n):

break

num = num +1

name.append(item.name)

date.append(str(day))

dataset=list(zip(name,date))

refr=pd.DataFrame(data=dataset, columns=['제품명','남은 일수'])

refr.index=refr.index+1

display(refr)

print("유통기한이 "+str(n)+"일 내인 제품은 총 "+str(num)+"개 입니다.")

def print\_data\_info\_total(self):

name0=[]

date0=[]

name1=[]

date1=[]

name2=[]

date2=[]

name3=[]

date3=[]

name4=[]

date4=[]

n0,n1,n2,n3,n4=0,0,0,0,0

for item in self.\_item:

day=item.compute\_date()

if(day<0):

name0.append(item.name)

date0.append(str(day))

n0 = n0 + 1

elif(day<=3):

name1.append(item.name)

date1.append(str(day))

n1 = n1+ 1

elif(day<=7):

name2.append(item.name)

date2.append(str(day))

n2 = n2 + 1

elif(day<=30):

name3.append(item.name)

date3.append(str(day))

n3 = n3 + 1

else:

name4.append(item.name)

date4.append(str(day))

n4 = n4 + 1

print("\n\*\*\*\*\*\*\*\*유통기한이 지난 제품\*\*\*\*\*\*\*\*")

dataset0=list(zip(name0,date0))

refr=pd.DataFrame(data=dataset0, columns=['제품명','남은 일수'])

refr.index=refr.index+1

display(refr)

print("총 "+str(n0)+"개의 제품이 있습니다.")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\n\*\*\*\*\*유통기한이 0~3일 남은 제품\*\*\*\*\*")

dataset1=list(zip(name1,date1))

refr=pd.DataFrame(data=dataset1, columns=['제품명','남은 일수'])

refr.index=refr.index+1

display(refr)

print("총 "+str(n1)+"개의 제품이 있습니다.")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\n\*\*\*\*\*유통기한이 4~7일 남은 제품\*\*\*\*\*")

dataset2=list(zip(name2,date2))

refr=pd.DataFrame(data=dataset2, columns=['제품명','남은 일수'])

refr.index=refr.index+1

display(refr)

print("총 "+str(n2)+"개의 제품이 있습니다.")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\n\*\*\*\*유통기한이 8~30일 남은 제품\*\*\*\*\*")

dataset3=list(zip(name3,date3))

refr=pd.DataFrame(data=dataset3, columns=['제품명','남은 일수'])

refr.index=refr.index+1

display(refr)

print("총 "+str(n3)+"개의 제품이 있습니다.")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\n\*\*\*유통기한이 31일 이상 남은 제품\*\*\*")

dataset4=list(zip(name4,date4))

refr=pd.DataFrame(data=dataset4, columns=['제품명','남은 일수'])

refr.index=refr.index+1

display(refr)

print("총 "+str(n4)+"개의 제품이 있습니다.")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

days=[]

num=[]

num.append(n0)

num.append(n1)

num.append(n2)

num.append(n3)

num.append(n4)

days.append("~0일")

days.append("0~3일")

days.append("4~7일")

days.append("8~30일")

days.append("31~일")

y = num

x = np.arange(len(days))

colors = sns.color\_palette('hls',len(days))

xlabel = days

plt.title("남은 유통기한")

plt.bar(x,y,color=colors,width=0.4,linewidth = 3)

plt.xticks(x,xlabel)

plt.yticks(y)

plt.xlabel("남은 일")

plt.ylabel("개수")

plt.show()

class warehouse(object):

def \_\_init\_\_(self):

self.\_item=[]

@property

def item(self):

return self.\_item

def put\_item(self,item):

self.\_item.append(item)

def remove\_item(self,n):

item = self.\_item[n-1]

name = item.name

self.\_item.remove(item)

print(name+"이(가) 삭제되었습니다.")

def print\_item(self,n):

num = n

name=[]

date=[]

for item in self.\_item:

num+=1

name.append(item.name)

dataset=list(zip(name))

refr=pd.DataFrame(data=dataset, columns=['제품명'])

refr.index=refr.index+n+1

display(refr)

return num

def comput\_category(self,dic):

for item in self.\_item:

category = item.category

dic[category]=dic.get(category,0)+1

def direct():

name = input("제품명:")

category = input("카테고리:")

print("\n유통기한을 입력하시겠습니까?\n1.네\n2.아니오")

try:

select = int(input("입력:"))

if bool(re.match(r"[1-2]",str(select)))==False :

raise ValueError

except ValueError as e:

print("선택지에 있는 숫자를 입력해주세요.")

return None, -1

if(select==1):

try:

date = input("\n유통기한 입력(YYYYMMDD):")

if len(date)!=8:

raise DateError("올비른 형식의 date를 입력하세요.")

date = dt.datetime.strptime(date,"%Y%m%d")

except DateError as e:

print(e.args[0])

return None,-1

except Exception as e:

print("존재하지 않는 날짜입니다.")

return None,-1

item = have\_expiration\_date(name,category,date)

return item, 1

else:

item = Item(name,category)

return item, 0

#인증키 = 290714c097bb4548bfa7

def search\_brcd():

try:

brcd = input("\n바코드 입력:")

if bool(re.match("\d{13}$",brcd))==False :

raise ValueError

except ValueError as e:

print("13자리 숫자를 입력해주세요.")

return None,-1

url = "https://openapi.foodsafetykorea.go.kr/api/290714c097bb4548bfa7/I2570/xml/1/5/BRCD\_NO="+brcd

response = urlopen(url).read()

xtree = ET.fromstring(response)

try:

name = xtree.find("row").find("PRDT\_NM").text

category = xtree.find("row").find("HRNK\_PRDLST\_NM").text

except Exception as e:

name = None

category = None

if(name == None and category == None):

print("제품을 찾지 못하였습니다. 직접 입력하시겠습니까?\n1.네\n2.아니오")

try:

select = int(input("입력:"))

if bool(re.match(r"[1-2]",str(select)))==False :

raise ValueError

except ValueError as e:

print("선택지에 있는 숫자를 입력해주세요.")

return None, -1

if(select==1):

name = input("제품명:")

category = input("카테고리:")

else:

return None,-1

else:

print("제품명: "+name)

print("카테고리: "+category)

print("\n유통기한을 입력하시겠습니까?\n1.네\n2.아니오")

try:

select = int(input("입력:"))

if bool(re.match(r"[1-2]",str(select)))==False :

raise ValueError

except ValueError as e:

print("선택지에 있는 숫자를 입력해주세요.")

return None, -1

if(select==1):

try:

date = input("\n유통기한 입력(YYYYMMDD):")

if len(date)!=8:

raise DateError("올비른 형식의 date를 입력하세요.")

date = dt.datetime.strptime(date,"%Y%m%d")

except DateError as e:

print(e.args[0])

return None,-1

except Exception as e:

print("존재하지 않는 날짜입니다.")

return None,-1

item = have\_expiration\_date(name,category,date)

return item, 1

else:

item = Item(name,category)

return item, 0

refrigerator = refrigerator()

warehouse = warehouse()

@checkTime

def select1():

print("1. 직접입력\n2. 바코드입력")

try:

select = int(input("입력:"))

if bool(re.match(r"[1-2]",str(select)))==False :

raise ValueError

except ValueError as e:

print("선택지에 있는 숫자를 입력해주세요.")

return

if(select == 1):

item, flag = direct()

if(flag == 1):

refrigerator.put\_item(item)

elif(flag == 0):

warehouse.put\_item(item)

elif(select == 2):

item, flag = search\_brcd()

if(flag == 1):

refrigerator.put\_item(item)

elif(flag == 0):

warehouse.put\_item(item)

@checkTime

def select2():

print("\n상품목록(0:돌아가기)")

n1 = refrigerator.print\_item()

n2 = warehouse.print\_item(n1)

if(n2==0):

print("상품이 없습니다.")

return

print("\n삭제할 상품 선택(번호 선택)")

choice = int(input("입력:"))

try:

if(choice == 0):

return

elif(choice>0 and choice<=n1):

refrigerator.remove\_item(choice)

elif(choice>n1 and choice<=n2):

warehouse.remove\_item(choice-n1)

else:

raise ChoiceError("※list에 해당하는 번호를 선택해주세요.")

except ChoiceError as e:

print(e.args[0])

@checkTime

def select3():

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*냉장고\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

refrigerator.print\_item()

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*창고\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

warehouse.print\_item(0)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

@checkTime

def select4():

dic={}

cate=[]

num=[]

refrigerator.comput\_category(dic)

warehouse.comput\_category(dic)

print("\n\*\*\*카테고리 별 개수\*\*\*")

if not dic:

print("물품이 없습니다.")

return

for c,n in dic.items():

cate.append(c)

num.append(n)

dataset=list(zip(cate,num))

category=pd.DataFrame(data=dataset, columns=['카테고리 명','개수'])

category.index=category.index+1

display(category)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

plt.pie(num,labels=cate,

shadow=True,autopct="%.1f%%",startangle=140,counterclock=False)

plt.title("카테고리 별 비중")

plt.show()

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

@checkTime

def select5():

print("\n1. 전체적인 유통기한 범위에 따른 재고 파악")

print("2. 직접 선택한 유통기한에 따른 재고 파악")

try:

select = int(input("입력:"))

if bool(re.match(r"[1-2]",str(select)))==False :

raise ValueError

except ValueError as e:

print("올바른 숫자를 입력해주세요.")

return

if(select==1):

refrigerator.print\_data\_info\_total()

elif(select==2):

print("\n일 수를 입력해주세요")

try:

select = int(input("입력(일):"))

if bool(re.match(r"\d+",str(select)))==False :

raise ValueError

except ValueError as e:

print("올바른 숫자를 입력해주세요.")

return

refrigerator.print\_day\_info(select)

fp = open("item.txt","rt",encoding='utf-8')

txt = fp.read()

line = txt.split('\n')

try:

line.remove("")

except ValueError as e:

pass

for item in line:

data = item.split('-')

if len(data)==3:

date = dt.datetime.strptime(data[2],"%Y%m%d")

item = have\_expiration\_date(data[0],data[1],date)

refrigerator.put\_item(item)

else:

item = Item(data[0],data[1])

warehouse.put\_item(item)

fp.close()

while(True):

print("\n1. 상품 등록")

print("2. 상품 제거")

print("3. 상품 목록 출력")

print("4. 카테고리 별 개수 출력")

print("5. 남은 유통기한 일에 따른 재고 파악")

print("0. 종료")

try:

select = int(input("입력:"))

if bool(re.match(r"[0-5]",str(select)))==False :

raise ValueError

except ValueError as e:

print("선택지에 있는 숫자를 입력해주세요.")

continue

if(select == 1):

select1()

elif(select == 2):

select2()

elif(select == 3):

select3()

elif(select == 4):

select4()

elif(select == 5):

select5()

elif(select == 0):

fp = open("item.txt","wt",encoding='utf-8')

for item in refrigerator.item:

date = item.date.strftime("%Y%m%d")

string = item.name+"-"+item.category+"-"+date+"\n"

fp.write(string)

for item in warehouse.item:

string = item.name+"-"+item.category+"\n"

fp.write(string)

fp.close()

print("\n프로그램을 종료합니다.")

break

**7. 프로젝트를 진행하며 느낀 점**

예전부터 생각만 하던 바코드로 유통기한을 관리하는 프로그램을 이번 기초빅데이터프로그래밍이라는 과목을 통해 직접 구현하게되어 정말 좋았다. 어려운점도 많았고 시간도 촉박하여 힘들긴 했지만 내가 생각한 아이디어를 직접 구현했다는 뿌듯함을 느낄 수 있었고 지금까지 과목에서는 뭔가 수동적으로 시키는 프로젝트만 진행했다면 이번엔 순수한 나의 작품을 만들 수 있다는 것이 앞으로의 프로그래밍에서도 자신감으로 다가올것이다.

또한 크롤링이라는 생소한 기능들을 구현하면서 새로운것들을 많이 배웠고 그 과정에서 공공데이터라는 새로운 것도 알게되었다. 앞으로 정말 많은 프로젝트들과 프로그래밍을 할텐데 이번 수업이 큰 도움이 될것같다. 또한 python을 다루는데 아직 미숙한 것을 깨닳았고 조금더 연습해 많이 익숙해져야겠다는 생각이 들었다.