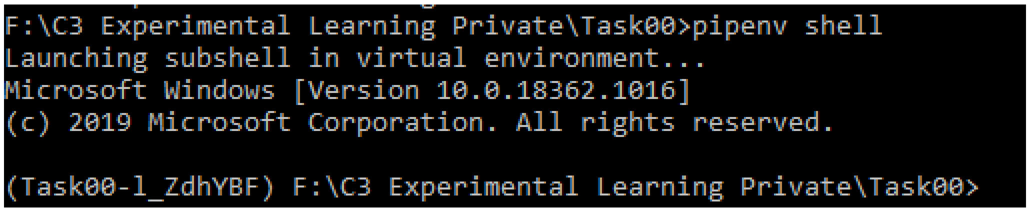
Documentation

Himanshu Yadav

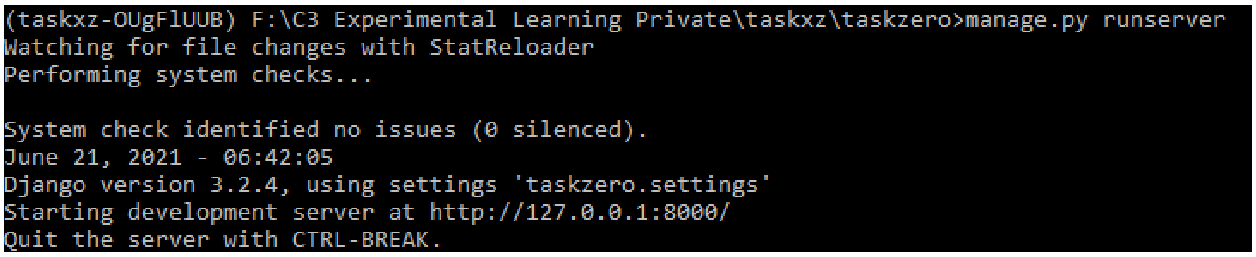
Starting the Project

1. Download the .zip or clone the repository
2. Use pipenv shell command to make virtual environment.



```
F:\C3 Experimental Learning Private\Task00>pipenv shell
Launching subshell in virtual environment...
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

(Task00-l_ZdhYBF) F:\C3 Experimental Learning Private\Task00>
```

3. Use pip install –r Requirements.txt in the directory to install the required libraries.
4. Use manage.py runserver command in the same directory as manage.py file.

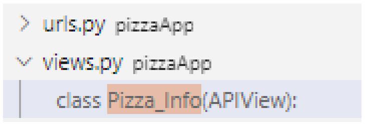


```
(taskxz-OUgFlUUB) F:\C3 Experimental Learning Private\taskxz\taskzero>manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 21, 2021 - 06:42:05
Django version 3.2.4, using settings 'taskzero.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

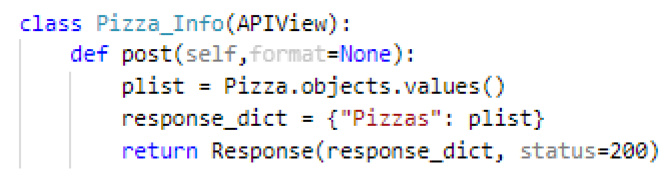
API Endpoints' documentation

```
1. admin/
2. api/pizza/ info/ [name='pInfo']
3. api/pizza/ create/ [name='pCreate']
4. api/pizza/ editdel/ [name='pEditDel']
5. api/pizza/ filter/ [name='pFilter']
```

1. View information of Pizzas:



```
> urls.py pizzaApp
∨ views.py pizzaApp
    class Pizza_Info(APIView):
```



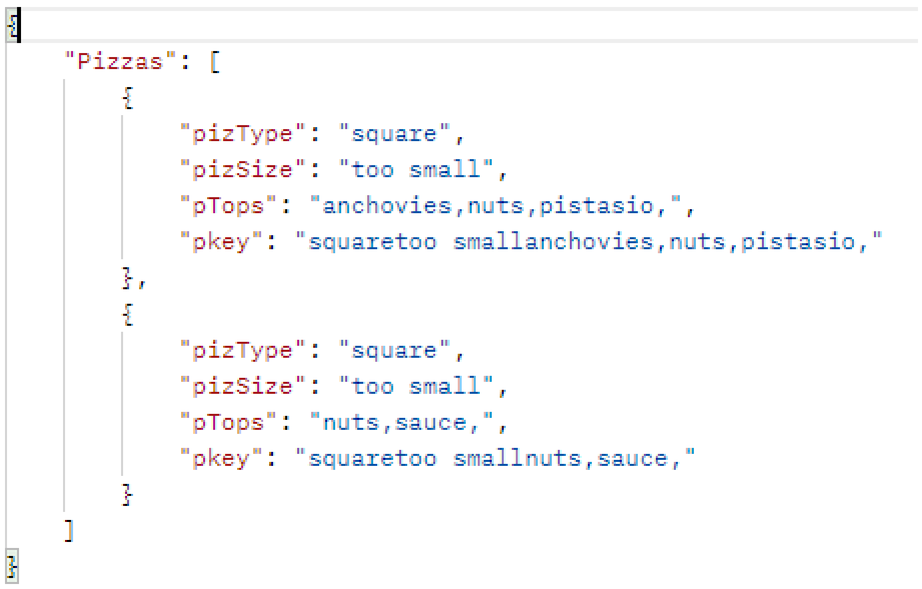
```python
class Pizza_Info(APIView):
    def post(self,format=None):
        plist = Pizza.objects.values()
        response_dict = {"Pizzas": plist}
        return Response(response_dict, status=200)
```

Description:

It takes all the information of Pizzas in the database and returns it as response_dict dictionary.

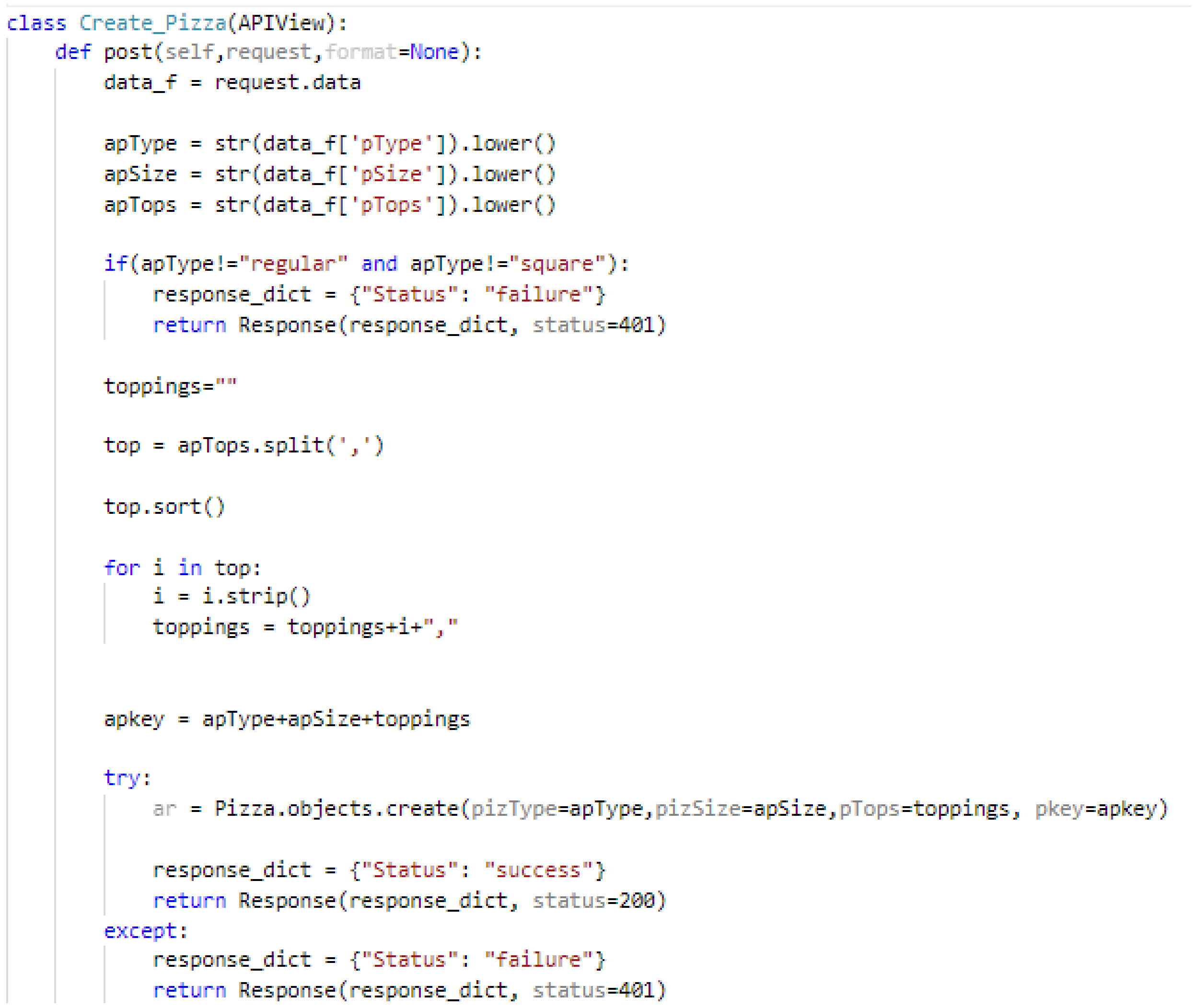
Input : takes no input

Output : List of all the Pizzas



```
{
    "Pizzas": [
        {
            "pizType": "square",
            "pizSize": "too small",
            "pTops": "anchovies,nuts,pistasio,",
            "pkey": "squaretoo smallanchovies,nuts,pistasio,"
        },
        {
            "pizType": "square",
            "pizSize": "too small",
            "pTops": "nuts,sauce,",
            "pkey": "squaretoo smallnuts,sauce,"
        }
    ]
}
```

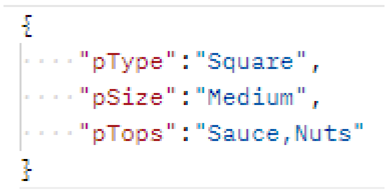
2. Create Pizza



```python
class Create_Pizza(APIView):
    def post(self,request,format=None):
        data_f = request.data

        apType = str(data_f['pType']).lower()
        apSize = str(data_f['pSize']).lower()
        apTops = str(data_f['pTops']).lower()

        if(apType!="regular" and apType!="square"):
            response_dict = {"Status": "failure"}
            return Response(response_dict, status=401)

        toppings=""

        top = apTops.split(',')

        top.sort()

        for i in top:
            i = i.strip()
            toppings = toppings+i+","


        apkey = apType+apSize+toppings

        try:
            ar = Pizza.objects.create(pizType=apType,pizSize=apSize,pTops=toppings, pkey=apkey)

            response_dict = {"Status": "success"}
            return Response(response_dict, status=200)
        except:
            response_dict = {"Status": "failure"}
            return Response(response_dict, status=401)
```

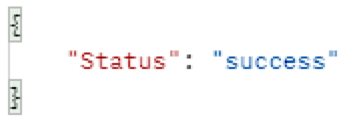
It takes value for input for a pizza and creates object for it in database, if type is not one of the two specified, it will not make the entry in DB.

Input data:



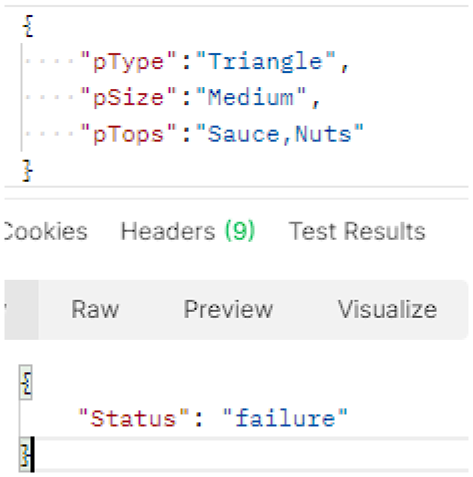
```json
{
    "pType":"Square",
    "pSize":"Medium",
    "pTops":"Sauce,Nuts"
}
```

Output and Result:



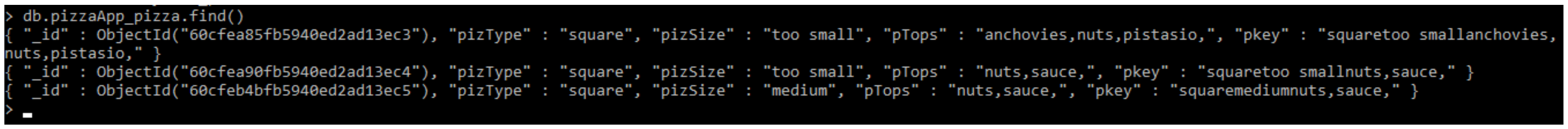
```
{
    "Status": "success"
}
```

If TYPE mentioned wrong, result is failure



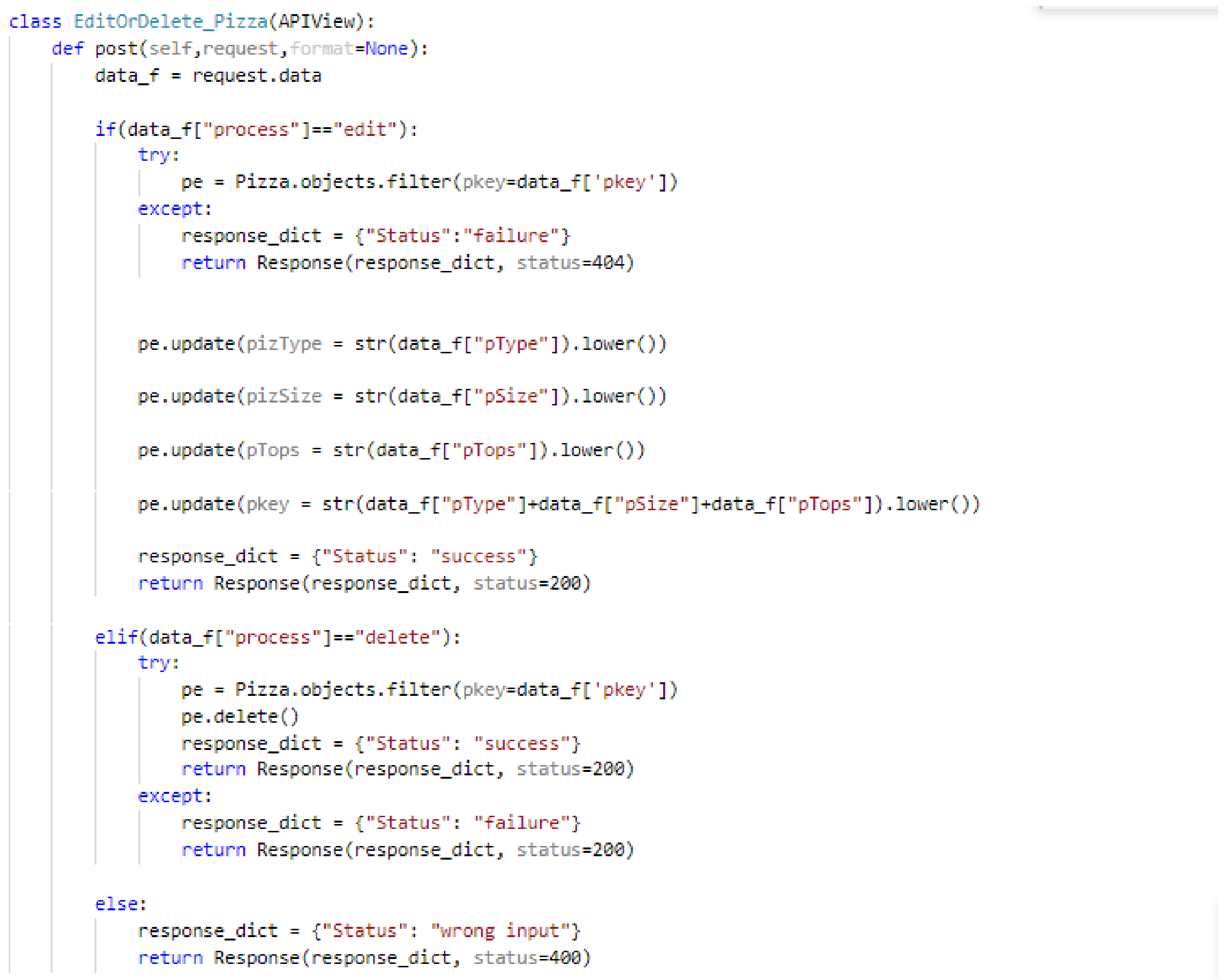
```
{
····"pType":"Triangle",
····"pSize":"Medium",
····"pTops":"Sauce,Nuts"
}
```

```
{
    "Status": "failure"
}
```



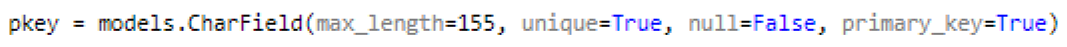
```
> db.pizzaApp_pizza.find()
{ "_id" : ObjectId("60cfea85fb5940ed2ad13ec3"), "pizType" : "square", "pizSize" : "too small", "pTops" : "anchovies,nuts,pistasio,", "pkey" : "squaretoo smallanchovies,
nuts,pistasio," }
{ "_id" : ObjectId("60cfea90fb5940ed2ad13ec4"), "pizType" : "square", "pizSize" : "too small", "pTops" : "nuts,sauce,", "pkey" : "squaretoo smallnuts,sauce," }
{ "_id" : ObjectId("60cfeb4bfb5940ed2ad13ec5"), "pizType" : "square", "pizSize" : "medium", "pTops" : "nuts,sauce,", "pkey" : "squaremediumnuts,sauce," }
> _
```

3. Edit or Delete Pizza



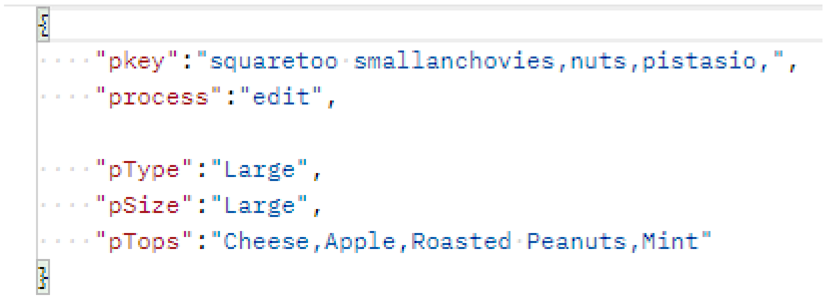
```
class EditOrDelete_Pizza(APIView):
    def post(self,request,format=None):
        data_f = request.data

        if(data_f["process"]=="edit"):
            try:
                pe = Pizza.objects.filter(pkey=data_f['pkey'])
            except:
                response_dict = {"Status":"failure"}
                return Response(response_dict, status=404)


            pe.update(pizType = str(data_f["pType"]).lower())

            pe.update(pizSize = str(data_f["pSize"]).lower())

            pe.update(pTops = str(data_f["pTops"]).lower())

            pe.update(pkey = str(data_f["pType"]+data_f["pSize"]+data_f["pTops"]).lower())

            response_dict = {"Status": "success"}
            return Response(response_dict, status=200)

        elif(data_f["process"]=="delete"):
            try:
                pe = Pizza.objects.filter(pkey=data_f['pkey'])
                pe.delete()
                response_dict = {"Status": "success"}
                return Response(response_dict, status=200)
            except:
                response_dict = {"Status": "failure"}
                return Response(response_dict, status=200)

        else:
            response_dict = {"Status": "wrong input"}
            return Response(response_dict, status=400)
```

It takes the value of primary key to identify the record to be edited or deleted, and then it updates or deletes the record as per new info provided. Process input determines the process to be performed.



```
pkey = models.CharField(max_length=155, unique=True, null=False, primary_key=True)
```

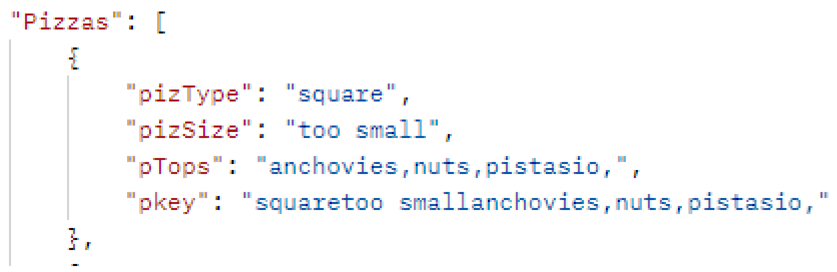
Input (edit) :



```
{
    "pkey":"squaretoo smallanchovies,nuts,pistasio,",
    "process":"edit",

    "pType":"Large",
    "pSize":"Large",
    "pTops":"Cheese,Apple,Roasted Peanuts,Mint"
}
```

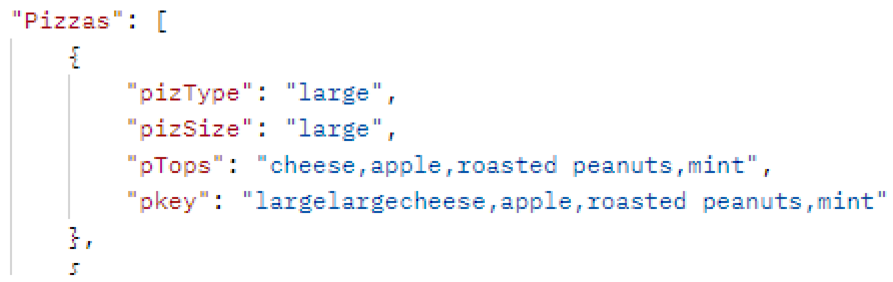
OUTPUT:

Before:



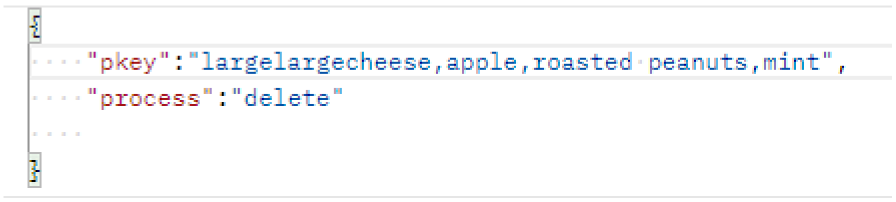
```
"Pizzas": [
    {
        "pizType": "square",
        "pizSize": "too small",
        "pTops": "anchovies,nuts,pistasio,",
        "pkey": "squaretoo smallanchovies,nuts,pistasio,"
    },
```

After:

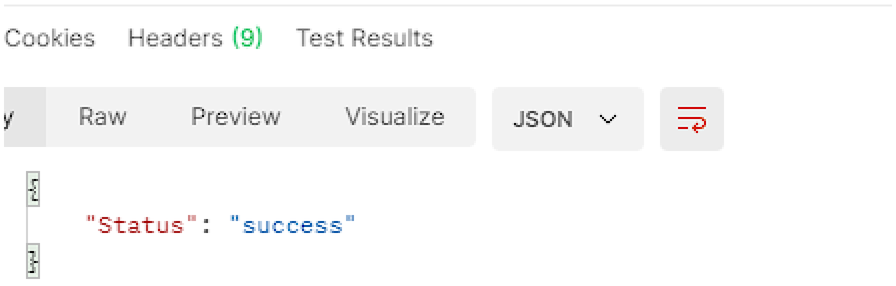


```
"Pizzas": [
    {
        "pizType": "large",
        "pizSize": "large",
        "pTops": "cheese,apple,roasted peanuts,mint",
        "pkey": "largelargecheese,apple,roasted peanuts,mint"
    },
    {
```

Input (delete):



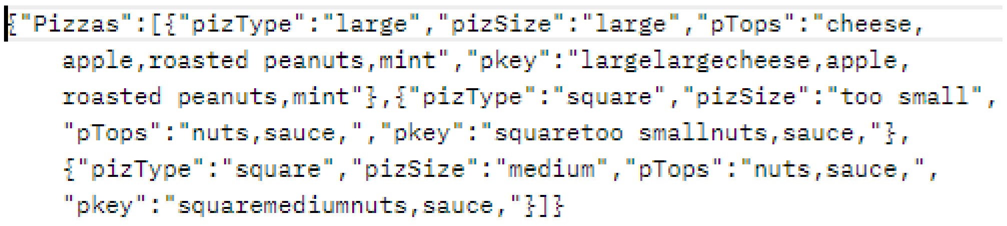
```
{
    "pkey":"largelargecheese,apple,roasted peanuts,mint",
    "process":"delete"
    
}
```



```
{
    "Status": "success"
}
```

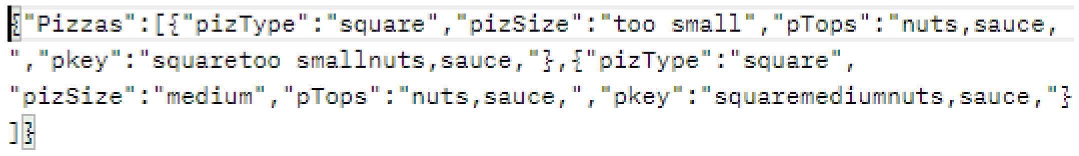
Output:

Before:



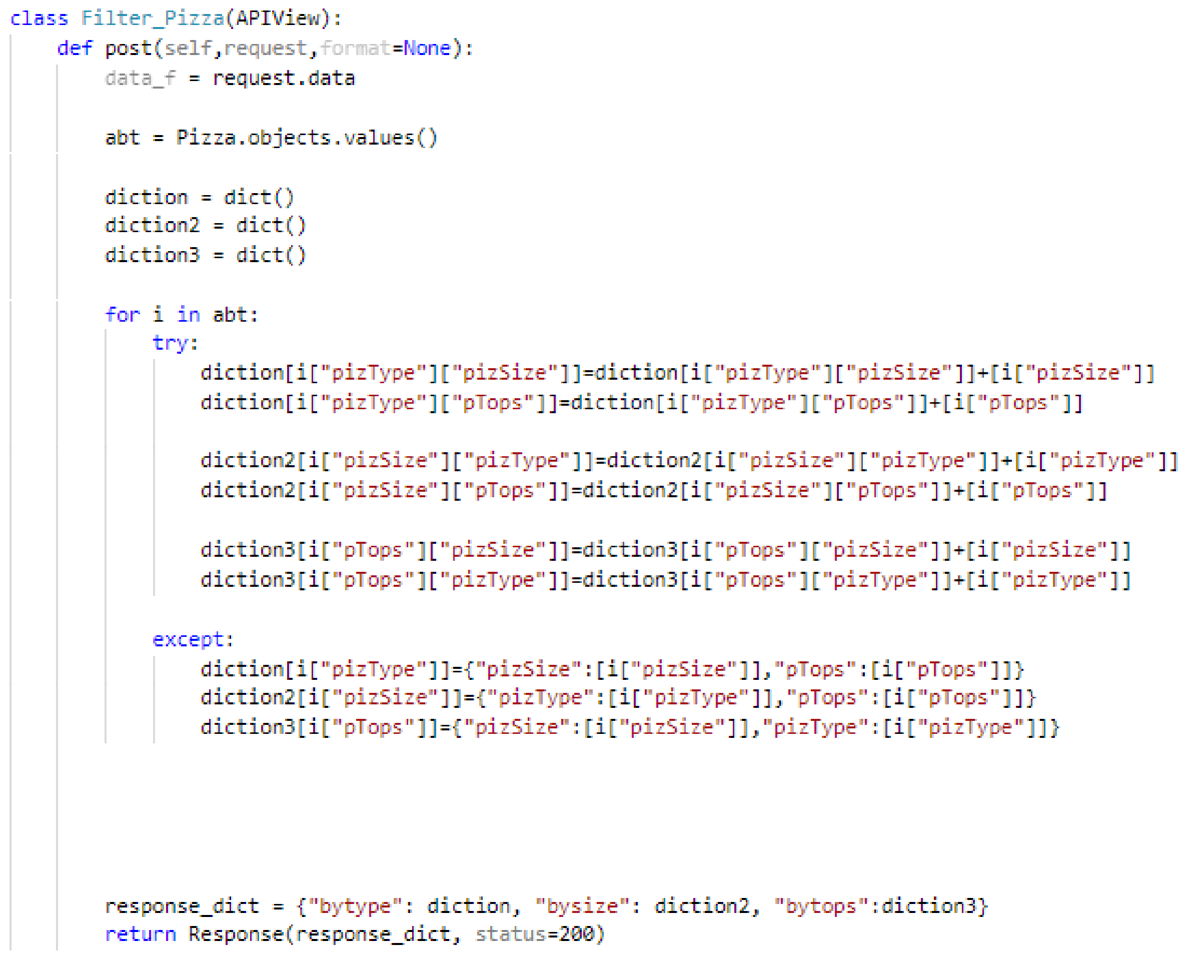
```
{"Pizzas":[{"pizType":"large","pizSize":"large","pTops":"cheese,
    apple,roasted peanuts,mint","pkey":"largelargecheese,apple,
    roasted peanuts,mint"},{"pizType":"square","pizSize":"too small",
    "pTops":"nuts,sauce,","pkey":"squaretoo smallnuts,sauce,"},
    {"pizType":"square","pizSize":"medium","pTops":"nuts,sauce,",
    "pkey":"squaremediumnuts,sauce,"}]}
```

After:



```
{"Pizzas":[{"pizType":"square","pizSize":"too small","pTops":"nuts,sauce,
","pkey":"squaretoo smallnuts,sauce,"},{"pizType":"square",
"pizSize":"medium","pTops":"nuts,sauce,","pkey":"squaremediumnuts,sauce,"}
]}
```

4. Filter:



```
class Filter_Pizza(APIView):
    def post(self,request,format=None):
        data_f = request.data

        abt = Pizza.objects.values()

        diction = dict()
        diction2 = dict()
        diction3 = dict()

        for i in abt:
            try:
                diction[i["pizType"]["pizSize"]]=diction[i["pizType"]["pizSize"]]+[i["pizSize"]]
                diction[i["pizType"]["pTops"]]=diction[i["pizType"]["pTops"]]+[i["pTops"]]

                diction2[i["pizSize"]["pizType"]]=diction2[i["pizSize"]["pizType"]]+[i["pizType"]]
                diction2[i["pizSize"]["pTops"]]=diction2[i["pizSize"]["pTops"]]+[i["pTops"]]

                diction3[i["pTops"]["pizSize"]]=diction3[i["pTops"]["pizSize"]]+[i["pizSize"]]
                diction3[i["pTops"]["pizType"]]=diction3[i["pTops"]["pizType"]]+[i["pizType"]]

            except:
                diction[i["pizType"]]={"pizSize":[i["pizSize"]],"pTops":[i["pTops"]]}
                diction2[i["pizSize"]]={"pizType":[i["pizType"]],"pTops":[i["pTops"]]}
                diction3[i["pTops"]]={"pizSize":[i["pizSize"]],"pizType":[i["pizType"]]}




        response_dict = {"bytype": diction, "bysize": diction2, "bytops":diction3}
        return Response(response_dict, status=200)
```

This takes no input and traverses through all entries and creates classification and filtering and returns a dictionary with three entities:

Filtered by Type

Filtered by Size

Filtered by Toppings

Output:

```
{
    "bytype": {
        "square": {
            "pizSize": [
                "medium"
            ],
            "pTops": [
                "nuts,sauce,"
            ]
        }
    },
    "bysize": {
        "too small": {
            "pizType": [
                "square"
            ],
            "pTops": [
                "nuts,sauce,"
            ]
        },
        "medium": {
            "pizType": [
                "square"
            ],
            "pTops": [
                "nuts,sauce,"
            ]
        }
    },
    "bytops": {
        "nuts,sauce,": {
            "pizSize": [
                "medium"
            ],
            "pizType": [
                "square"
            ]
        }
    }
}
```