2) RESTful Web Services: REST stands for Representational State Transfer. It means that each unique URL is a represention of some object. We can get contents of this object by using HTTP GET, we can modify by using PUT/PATCH and we can delete by using DELETE. We can create by using POST. Most of the times RESTful web service will provide data in the form of JSON, parsing is not difficult. Hence this type of web services are faster when compared with SOAP based Web Services.Transfer of JSON Data over the network requires less bandwidth. Limitations: 1) It is less secured. 2) It provide support only for the protocols which can provide URI, mostly HTTP. Note: Because of ligth weight, high performance, less bandwidth requirements, easy development, human understandable message format, this type of web services are most commonly used type of web services. Differences between SOAP and REST 1) XML Based Message Protocol. 1) An Architecture Syle but not protocol. 2) Uses WSDL for communication between consumer and provider. 2) Uses xml/json to send and receive data. 3) Invokes services by using RPC Method calls. 3) Invokes services by simply URL Path. 4) Does not return human readable result. 4) Returns readable results like plain xml or JSON. 5) These are heavy weight. 5) These are light weight. 6) These require more bandwidth. 6) These require less bandwidth. 7) Can provide support for multiple protocols like HTTP,SMTP,FTP etc. 7) Can provide support only for protocols that provide URI, mostly HTTP. 8) Performance is Low. 8) Perforamnce is High. 9) More Secured. 9) Less Secured. Note: Most of the Google web services are SOAP Based. Yahoo RESTful eBay and Amazon using both SOAP and Restful. Web Service Provider vs WebService Consumer: --> The application which is providing web services is called Web Service Provider. --> The application which is consuming web data through web services, is called Web service consumer. Eg: Bookmyshow app <--> Payment Gateway app To complete our booking, bookmyshow application will communicates with payment gateway application. Hence payment gateway applications acts as webservice provider and bookmyshow application acts as web service consumer Django View Function to send HTML Response: def employee_data_view(request): employee_data= {'eno':100,'ename':'Naresh','esal':1000,'eaddr':'Hyderabad'} resp='

Employee No:{}
Employee Name:{}
Employee Salary:{}
Employee Address:{}

'.format(employee_data['eno'],employee_data['ename'],employee_data['esal'],employee_data['eaddr']) return HttpResponse(resp) Django View Function to send HTTPResponse with JSON Data: To handle json data, python provides inbuilt module: json This module contains dumps() function to convert python dictionary to json object.(serialization) def employee_data_jsonview(request): employee_data={'eno':100,'ename':'Naresh','esal':1000,'eaddr':'Hyderabad'} json_data=json.dumps(employee_data) return HttpResponse(json_data,content_type='application/json') Note: This way of sending JSON response is very old. Newer versions of Django provided a special class JsonResponse. Django View Function to send JsonResponse directly: The main advantage of JsonResponse class is it will accept python dict object directly. It is the most convinient way to provide json response. from djando.http import JsonResponse def employee_data_jsondirectview(request): employee_data= {'eno':100,'ename':'Naresh','esal':1000,'eaddr':'Hyderabad'} return JsonResponse(employee_data) Python Application to communicate with Django Application: From python if we want to send http request we should go for requests module. test.py: import requests BASE_URL='http://127.0.0.1:8000/' ENDPOINT='read/' --> this is url of view r=requests.get(BASE_URL+ENDPOINT) data=r.json() print('Employee Number:',data['eno']) print('Employee Name:',data['ename']) print('Employee Salary:',data['esal']) print('Employee Address:',data['eaddr']) Note: In the above case, python application communicates with django application to get employee data. For this common language used is: Http and common message format used is JSON. HTTPie Module: We can use this module to send http request from commond prompt. We can install as follows pip install httpie We can http request as follow C:\Users\LENOVO>http http://127.0.0.1:8000 HTTP/1.0 200 OK Content-Length: 72 Content-Type:

application/json Date: Thu, 13 Dec 2018 10:17:54 GMT Server: WSGIServer/0.2 CPython/3.6.5 X-Frame-Options: SAMEORIGIN { "eaddr": "Hyderabad", "ename": "Naresh", "eno": 100, "esal": 1000 } Class Based View(CBV) to send JSON Response: Every class based view in django should extends View class.It present in the following package. django.views.generic Within the class we have to provide http methods like get(),post() etc Whenever we are sending the request, the corresponding method will be executed. from django.views.generic import View class JsonCBV(View): def get(self,request,*args,**kwargs): employee_data= {'eno':100,'ename':'Naresh','esal':1000,'eaddr':'Hyderabad'} return JsonResponse(employee_data) urls.py url(r'^cbv1/', views.JsonCBV.as_view()) Mixin(Mixed In): -->Mixins are special type of inheritance in Python. -->It is limited version of Multiple inheritance. -->In multiple inheritance, we can create object for parent class and parent class can extend other classes. But in Mixins, for the parent class we cannot create object and it should be direct child class of object.i.e parent class cannot extend any other classes. --> In Multiple inheritance, parent class can contain instance variables.But in Mixins, parent class cannot contain instance variable but can contain class level variables. --> Hence the main purpose of parent class in Mixins is to provide functions to the child classes. Differences between Mixins and Multiple Inheritance Mixins Multiple Inheritance 1) Parent class instantiation is not possible 1) Parent class instantitation is possible. 2) It contains only instance methods but not instance variables. 2) It contains both instance methods and variables. 3) The methods are useful only for child classes. 3) The methods are useful for both Parent and child classes. 4) Parent class should be direct child class of object. 4) Parent class can extend any class need Note: 1) Mixins are reusable classes in django. 2) Mixins are available only in languages which provide support for multiple inheritance like Python,Ruby,Scala etc 3) Mixins are not applicable for Java and C#, because these languages won't support multiple inheritance