**Python**

* 2\*\*3 2 power 3
* F String literals to use variable in string f pehlay phir {var name}

print(f"he says {my\_var}")

* Var name[0:9] upto 9 not add index 9
* Add variable in list “append”
* Insert same flutter wala insert(index,var)
* Remove at index = my\_list.pop(index)
* Length of list len(myList)
* Dictionaries are a [“key”:”value”] pair in python
* Print(Dictionary.key()) only the keys name
* example\_dict={"outer":{"inner":100}}
* Print(example\_dic[“outer”]) print ={“inner”:100}
* Print(example\_dic[“outer”][“inner]) print =100
* Tuples are final variables they can’t be change
* Mytopple=(1,2,4) …assign with this () brackets
* If else don’t need body just colons eg:

if employe\_data[0]['waiter']=="umer":

    print("umer is waiter")

elif employe\_data[0]['waiter']=="ahmed":

   print("ahmed is waiter")

else:

 print(f"{employe\_data[0]['waiter']} is waiter")

* For each loop eg

my\_var =[11,2,2,4,5,6,7,3]

for item1 in my\_var:

  print(item1)

* While loop

n=1

while n<10:

 print(f"n is {n} but still not 10")

 n=n + 1

* Functions in python:

def checker\_num(list):

    for it in list:

        if it==11:

         return True

    return False

ir=checker\_num(my\_var)

* Self keyword in python:
* What is SELF in Python? SELF represents the instance of class. This handy keyword allows you to access variables, attributes, and methods of a defined class in Python.
* Try catch in python

try:

    print("10"+10)

except:

    print("error")

finally:

    print("hello this is a code after error")

* class and contructors In python

class Student():

    university="iqra" #attributes of a class

    def \_\_init\_\_(self,gpa,name,id,): #contructor of a class

        self.name=name

        self.gpa=gpa

        self.id=id

x=Student(name="raza",gpa=2.8,id="42122")

x.university="ios"

print(x.university)

* methods

class Circle:

    pi=3.142

    def \_\_init\_\_(self,radius):

        self.radius=radius

    def area(self):

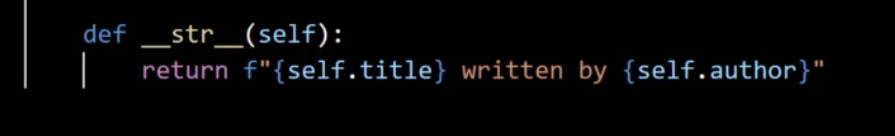
        return self.radius\*\*2\*self.pi

    def perimeter(self):

        return self.radius \*2 \*self.pi

x=Circle(radius=4)

print(x.perimeter())

* **inheritance and overriding(polymorphisim)**
* class Circle:
* pi=3.142
* def \_\_init\_\_(self,radius):
* self.radius=radius
* def area(self):
* return  f"area of circle is {self.radius\*\*2\*self.pi}"
* def area(self,pi):
* return  self.radius\*\*2\*pi
* def perimeter(self):
* return f"perimeter of circle is {self.radius \*2 \*self.pi}"
* class Shape(Circle): #inheritance
* # overiding
* def \_\_init\_\_(self,radius,border\_colors):
* Circle.\_\_init\_\_(self,radius,)
* self.radius=radius
* self.border\_colors=border\_colors
* def shapeName(self):
* if self.radius<=0:
* return f"this is square with{self.border\_colors} color"
* elif self.area(3.142) < 50:
* return f"this is big circle with {self.border\_colors} color"
* else:
* return f"this is circle with {self.border\_colors} color"
* x=Shape(radius=1,border\_colors="red")
* print (x.area(3.142))
* print (x.perimeter())
* print (x.shapeName())
* x=Circle(radius=4)
* # print(x.perimeter())
* Str return the string of an object
* ****
* Imports a class directly from same folder

from sample import Shape,Circle #this "circle" and "Shape" is a class from same folder

x=Circle(radius=3.142)

x.area(pi=3.142)

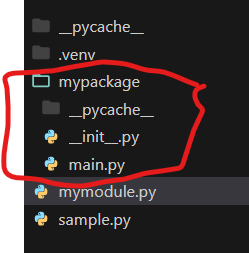
y=Shape(radius=3.142,border\_colors="red")

y.shapeName()

* And importing from other folder the folder must have “\_\_init\_\_.py” file

from mypackage.main import checker\_num

checker\_num(list=[1,2,3,4,5,6,7])



* A module is a directory which includes classes function methods A package is a directory which includes many sub module and its on \_\_inti\_\_.py file

Opening a virtual env

create virtual environment

python -m venv venv

Then activate that environment

venv\Scripts\activate

Then install Django in that environment

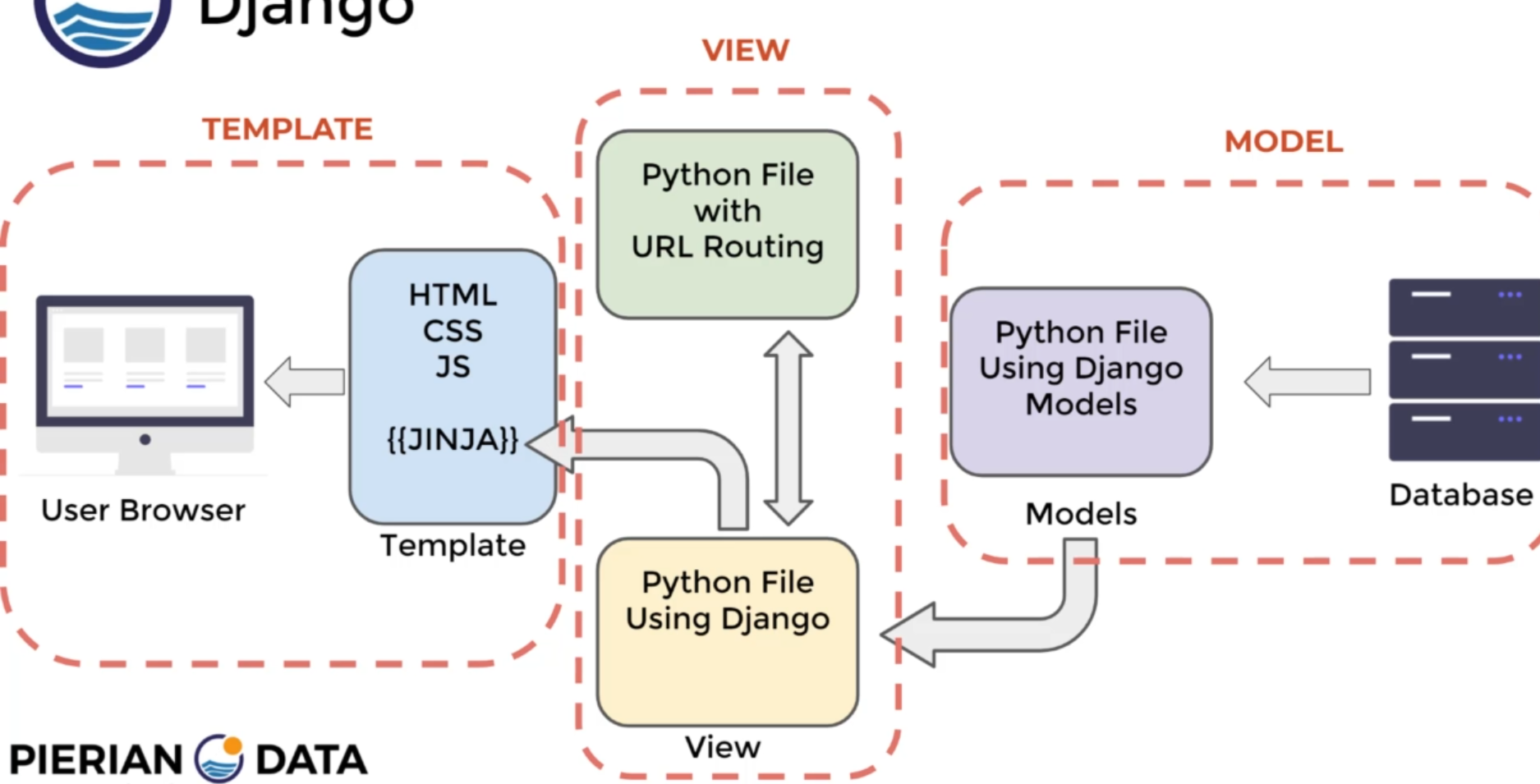
pip install django

**Django**

<https://www.djangoproject.com/>

MTV

Model Template View

****

**Create Django Project:**

* Django-admin startproject project\_name
* To run serve “python manage.py runserver” if want to change the port add “portname”
* To create another app within your app “python manage.py startapp “app\_name”
* Import httpService:

from django.http import HttpResponse

**How to start:**

* First make a view in child app in a views.py file

from django.shortcuts import render

from django.http import HttpResponse #importing https service file

# Create your views here.

def resp():

  HttpResponse("hell this is my view") #creating a view function

* then connect the view with child app url.py

from . import views

from django.urls import path #importing djnago url file

urlpatterns=[

    path('',views.resp,name="resp")

]

* then connect this child file urls .py to main project’s urls.py file

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

    path("my\_app\_/",include("my\_app\_.urls")), #connected with the child project urls.py file

    path('admin/', admin.site.urls),

]

* **Create a dynamic path**
* in views file

articles={

    "sports":"Messi Win a Ballondoor",

    "financial":"dollar got high",

    "Politics":"Trump ear got injured with a bullet shot by a person with downsyndrom"

}

def sample\_view(request,topic):

    return HttpResponse(articles[topic])

* in url file

from django.urls import path

from first\_app import views

urlpatterns=[

    path('<topic>',views.sample\_view,) #add this "<topic>" keyword in path to get a dynamic urls

]

* creating a dynamic view to add numers
* view page

#creating a dynamic view to add a numbers

def add\_num(request,num1,num2):

    add\_result=num1 + num2

    result=f"{num1} + {num2}={add\_result}"

    return HttpResponse(result)

* url page

path('<int:num1>/<int:num2>',views.add\_num,)  #add this'<int:num1>/<int:num2>' keyword in path to get a dynamic urls

* 404 error

def sample\_view(request,topic):

    try:

        result=articles[topic]

        return HttpResponse(articles[topic])

    except:

        return Http404('Generic Error 404')

* Redirect with **reverse**

**View page**

from django.http import HttpResponse,HttpResponseNotFound,Http404,HttpResponseRedirect

from django.urls import reverse

def num\_page\_redirect(request,num\_redirct):

    key\_list=list(articles.keys())

    topic=key\_list[num\_redirct]

    try:

      result=topic

        return HttpResponseRedirect(reverse("topic-page",args=[topic]))

* Urls

  path('<str:topic>',views.sample\_view,name="topic-page"), #add this "<topic>" keyword in path to get a dynamic urls add name sp they can redirect

* Importing templates
* Settings.py file

import os

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [os.path.join(BASE\_DIR,"template/")], #importing template path from template folder

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

* Views.py file

def html\_view(request):

return render(request,'first\_app/example.html') #path of html file after template/

* url.py file

  path('htmlView/',views.html\_view)