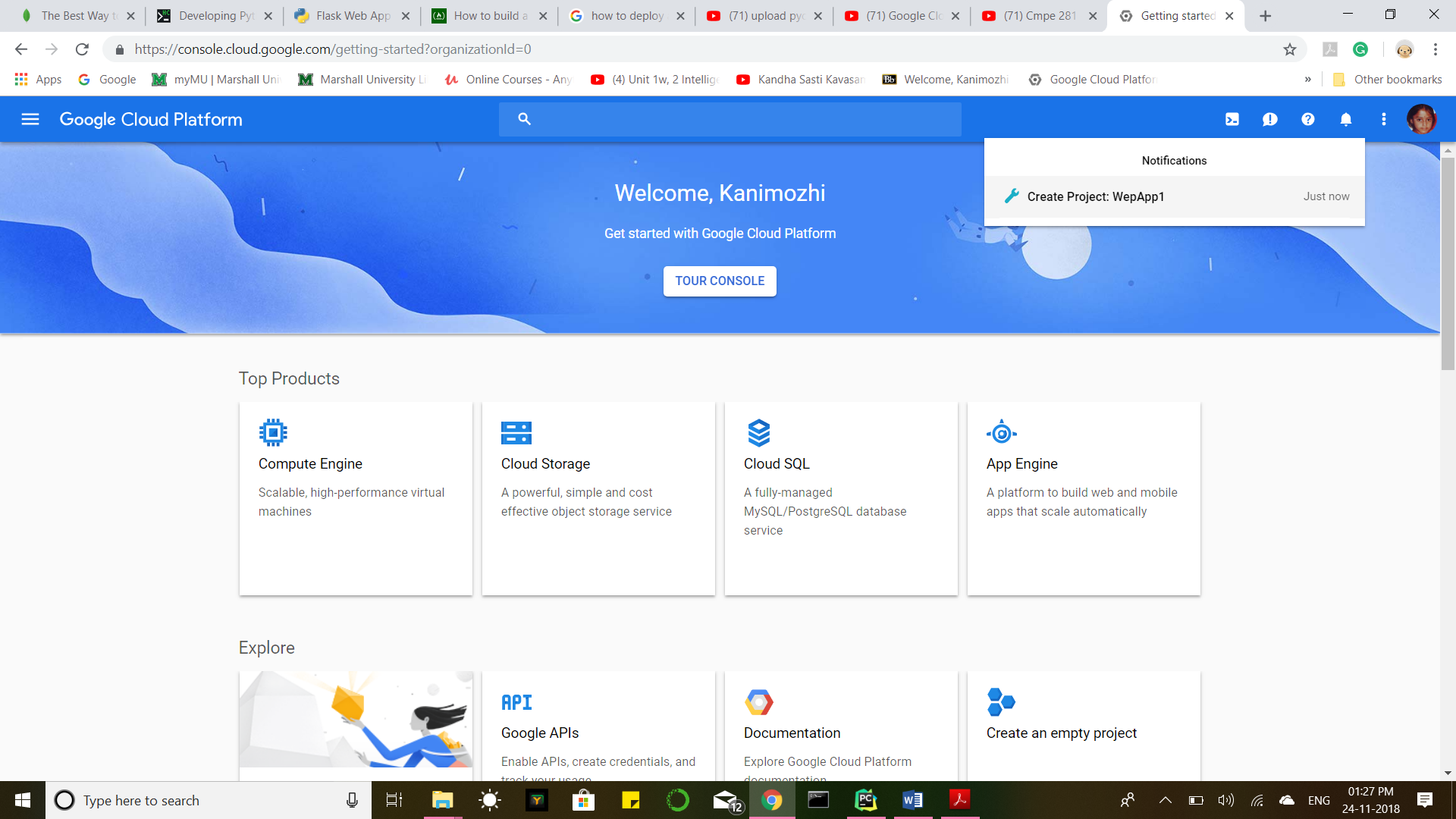
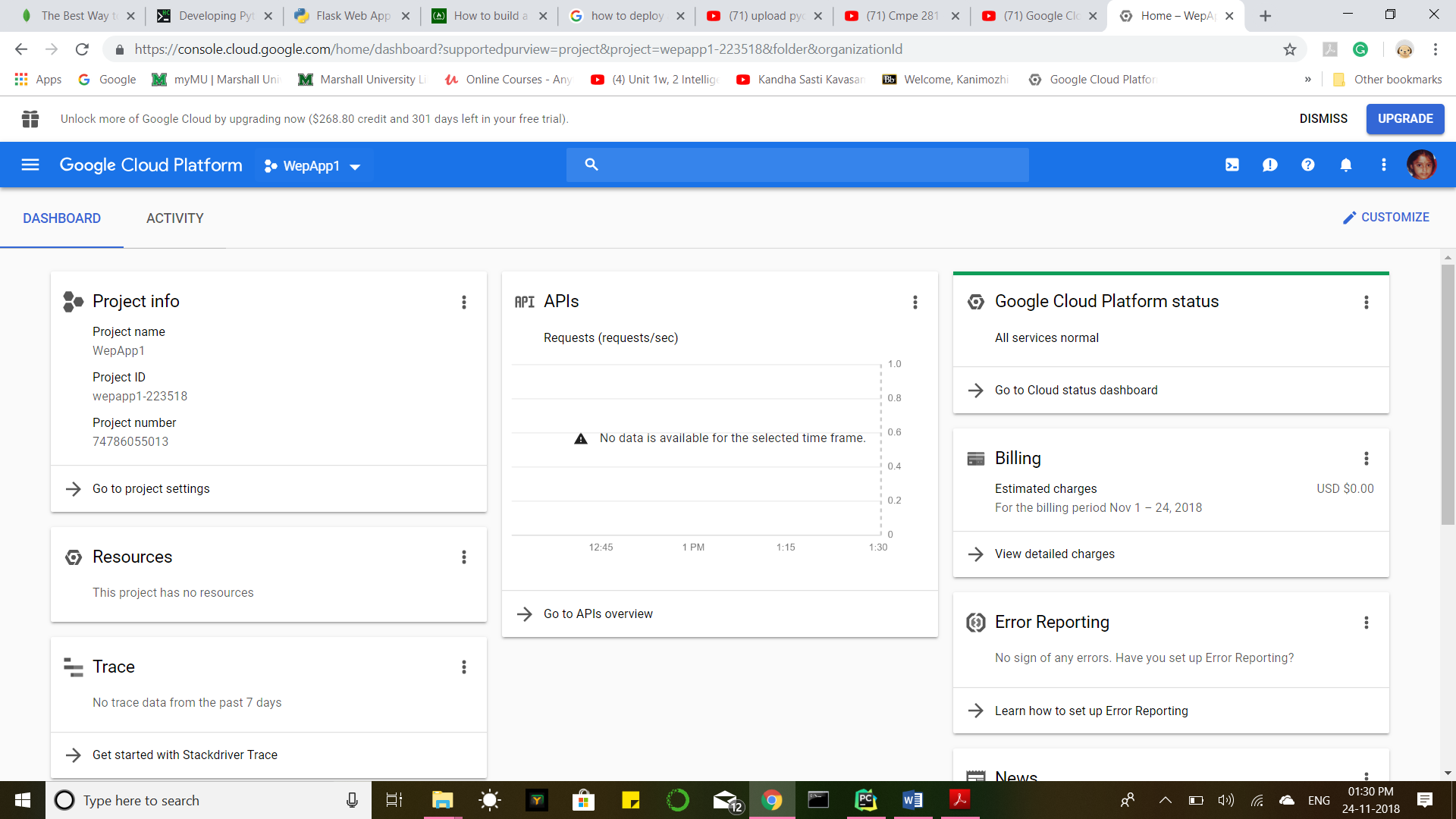
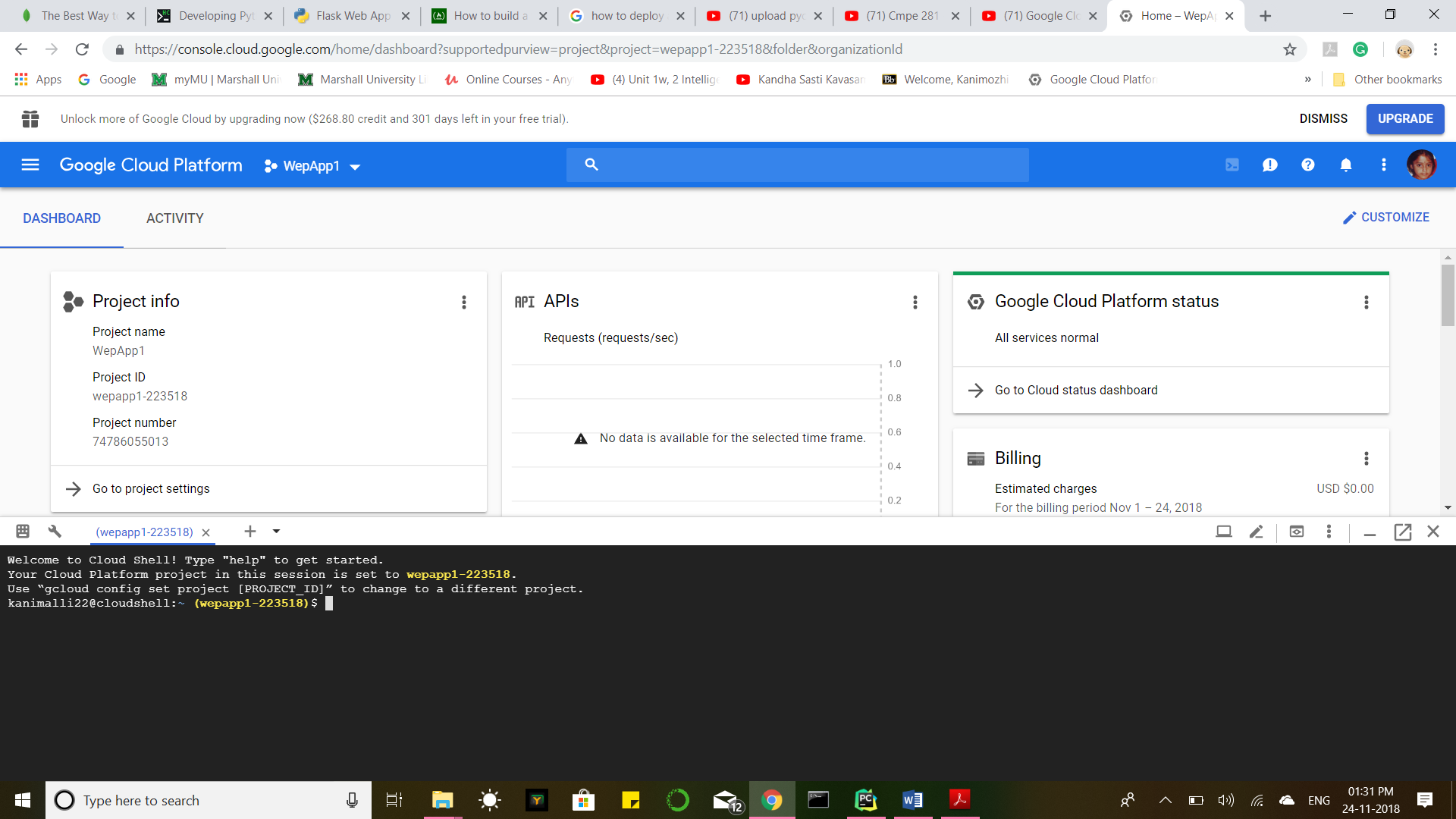
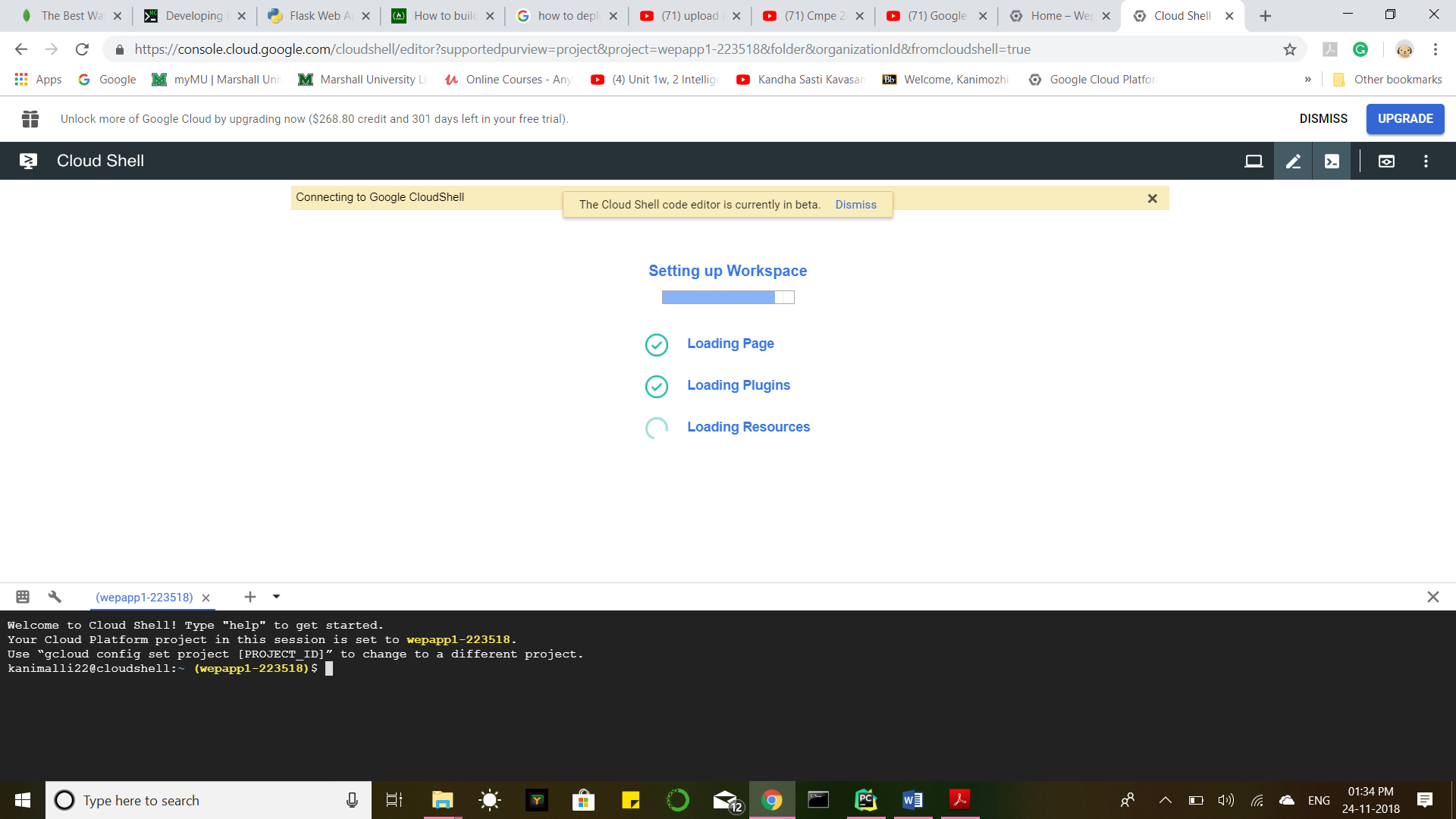


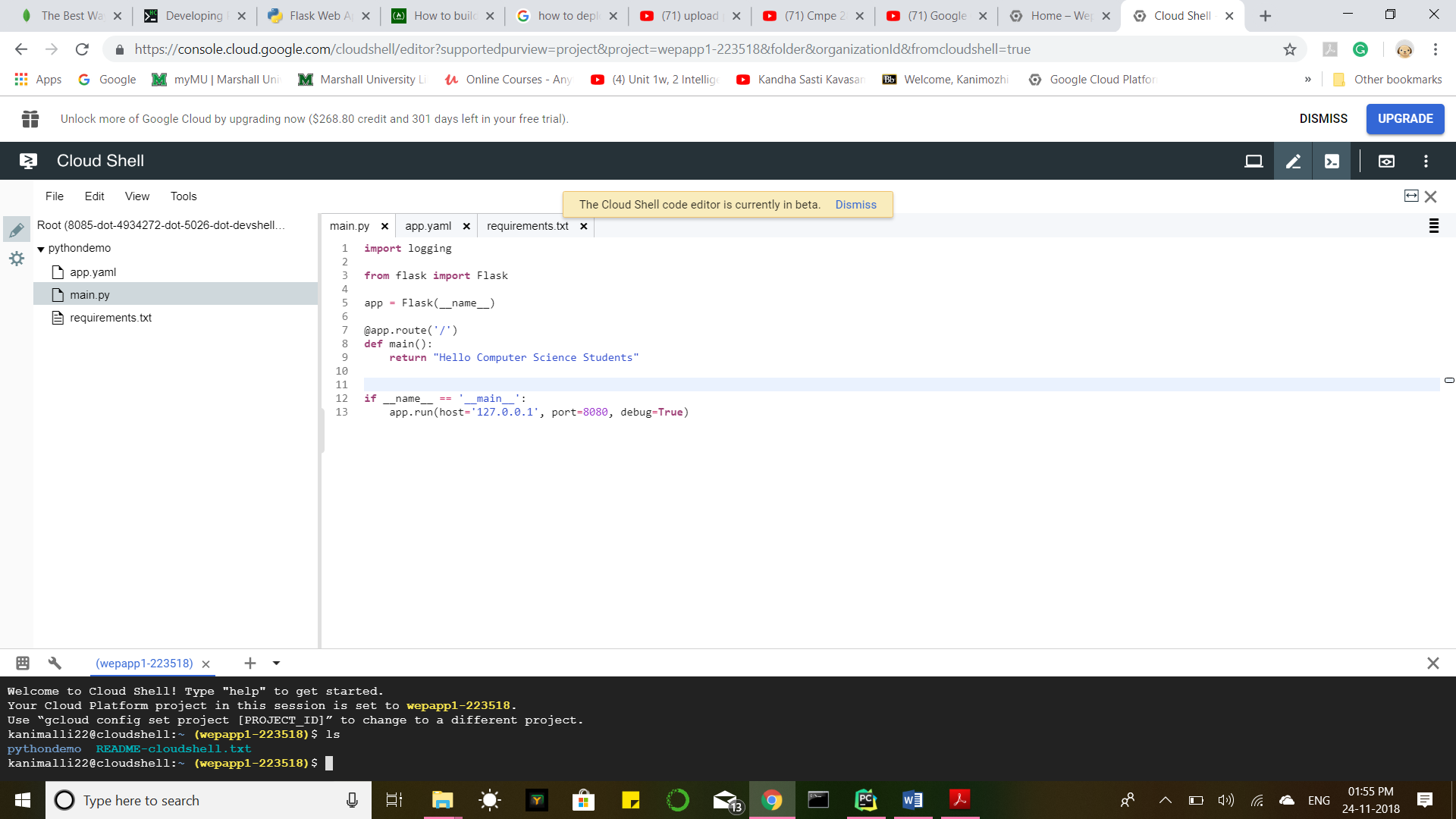
https://www.youtube.com/watch?v=o8XxAWZwnOg











Files :

main.py

import logging

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def main():

return "Hello Computer Science Students"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='127.0.0.1', port=8080, debug=True)

app.yaml

runtime : python

env : flex

entrypoint : gunicorn -b :$PORT main:app

runtime\_config:

python\_version: 3

requirements.txt

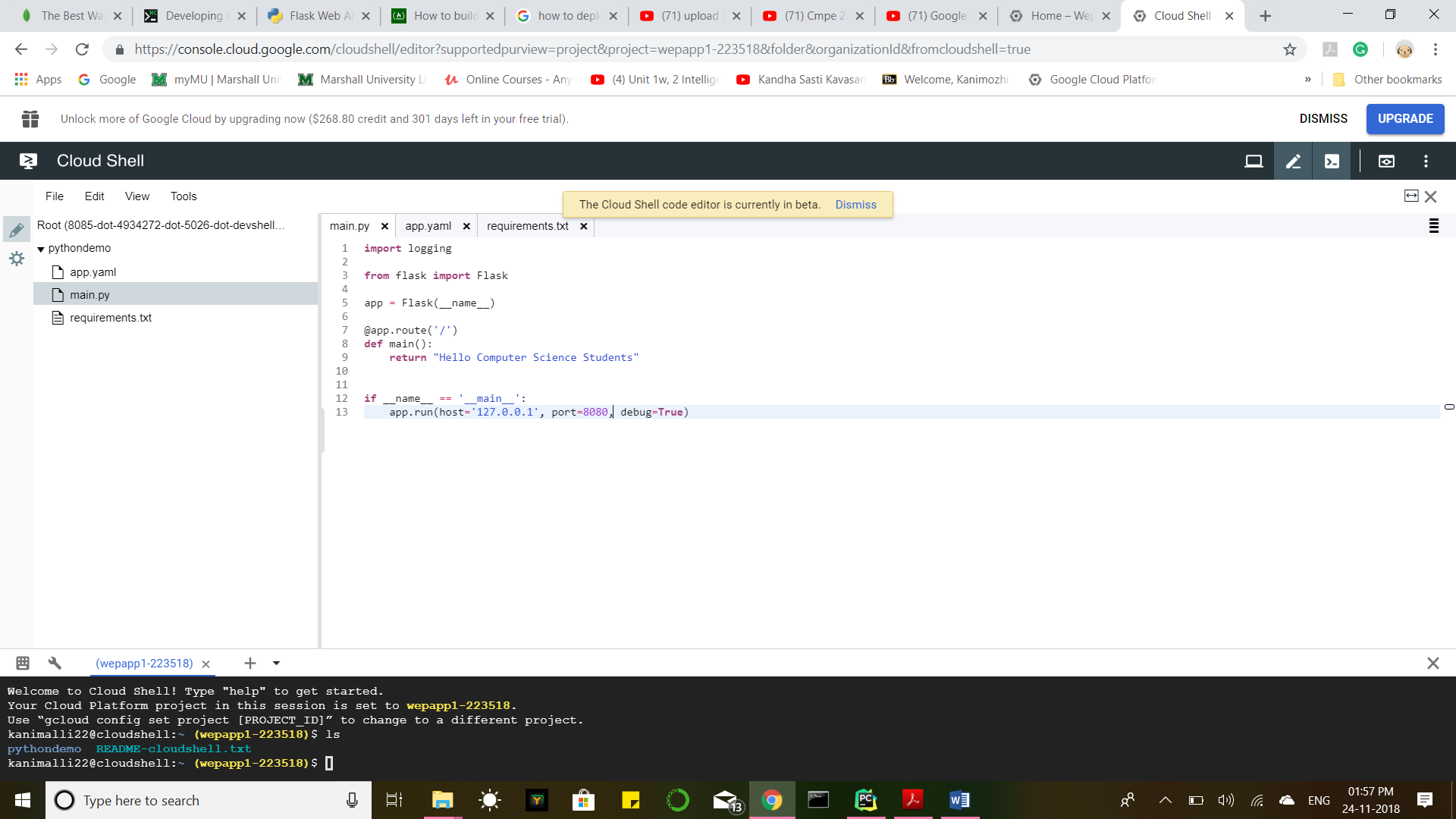
runtime : python

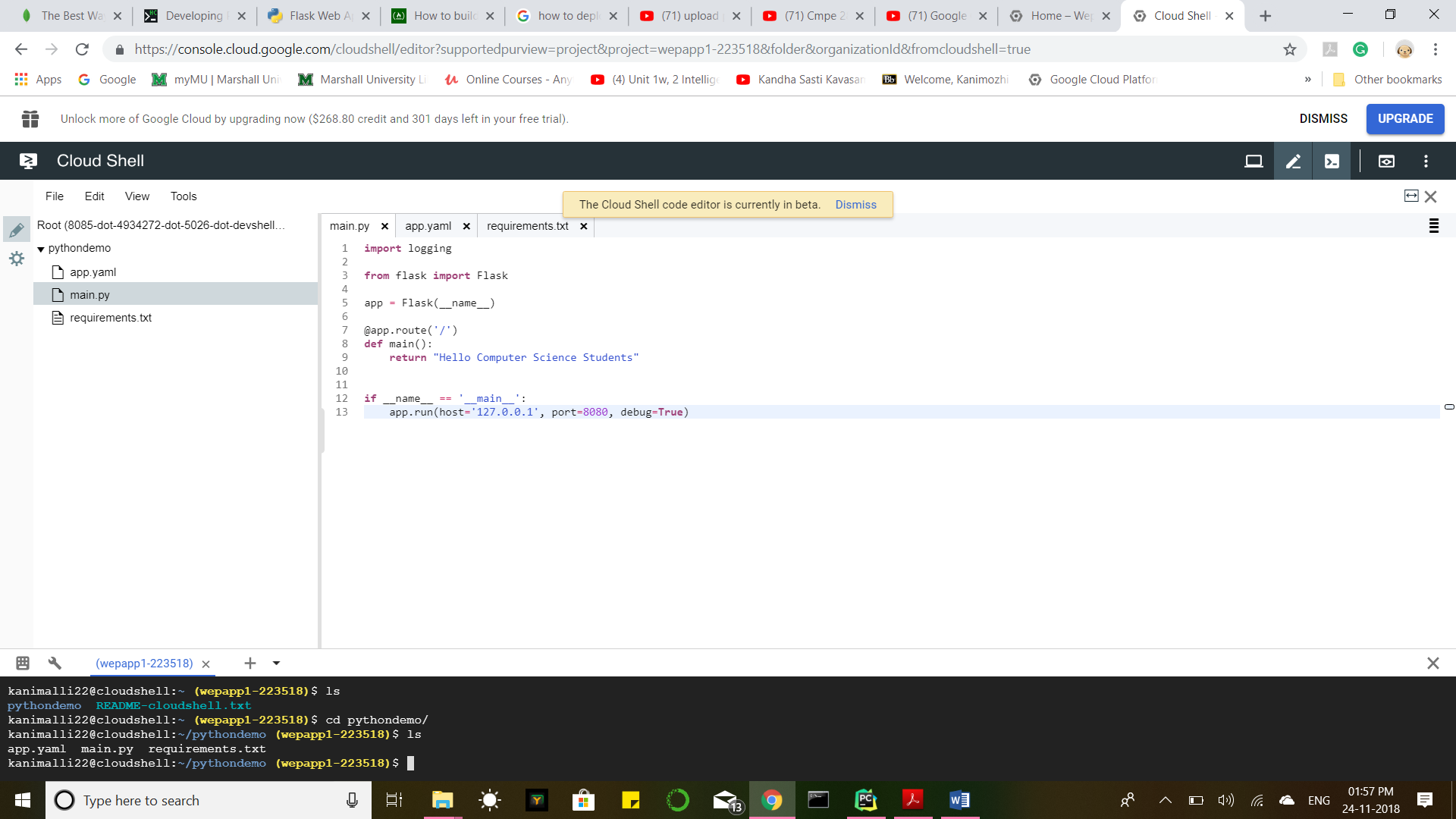
env : flex

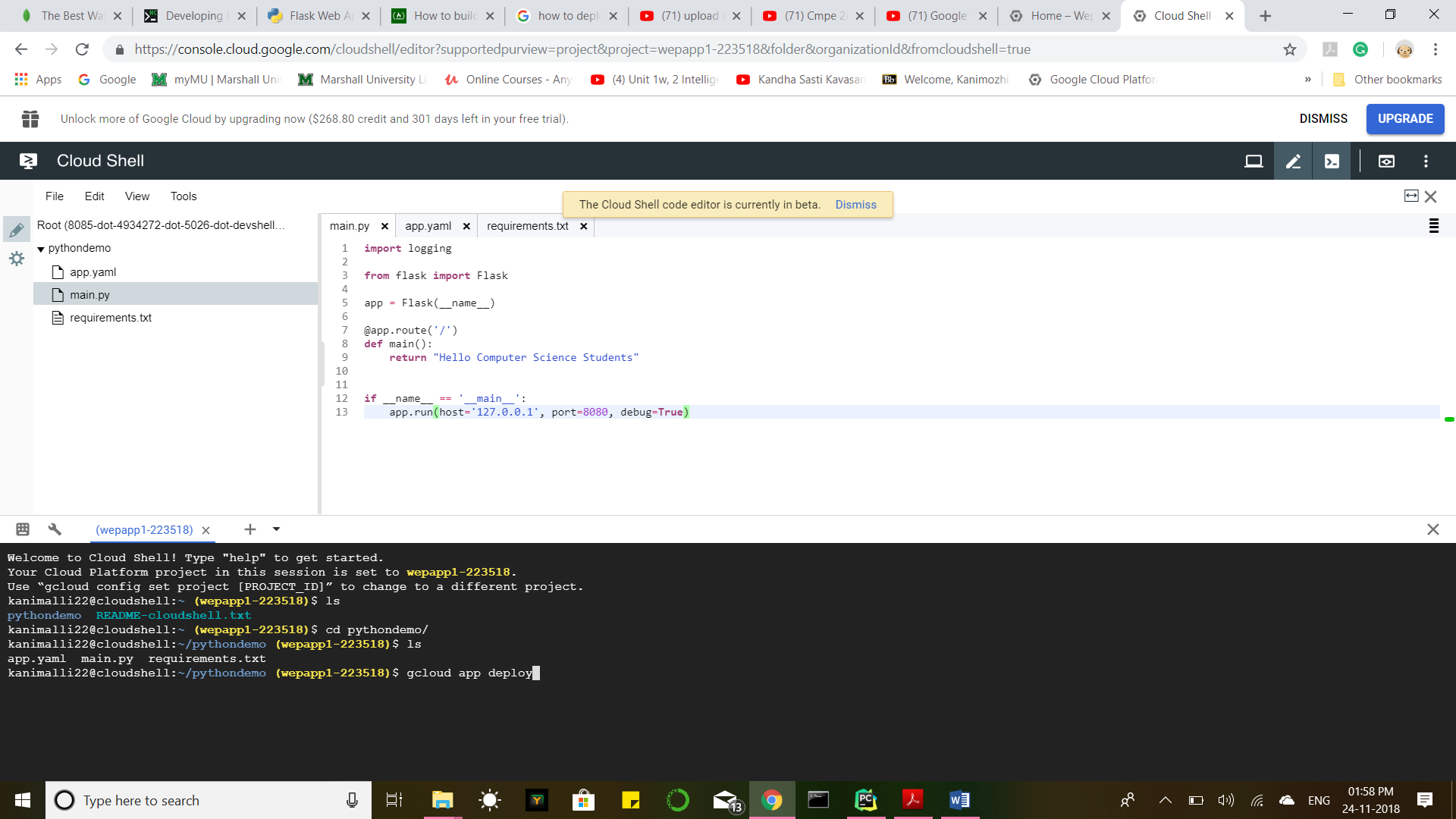
entrypoint : gunicorn -b :$PORT main:app

runtime\_config:

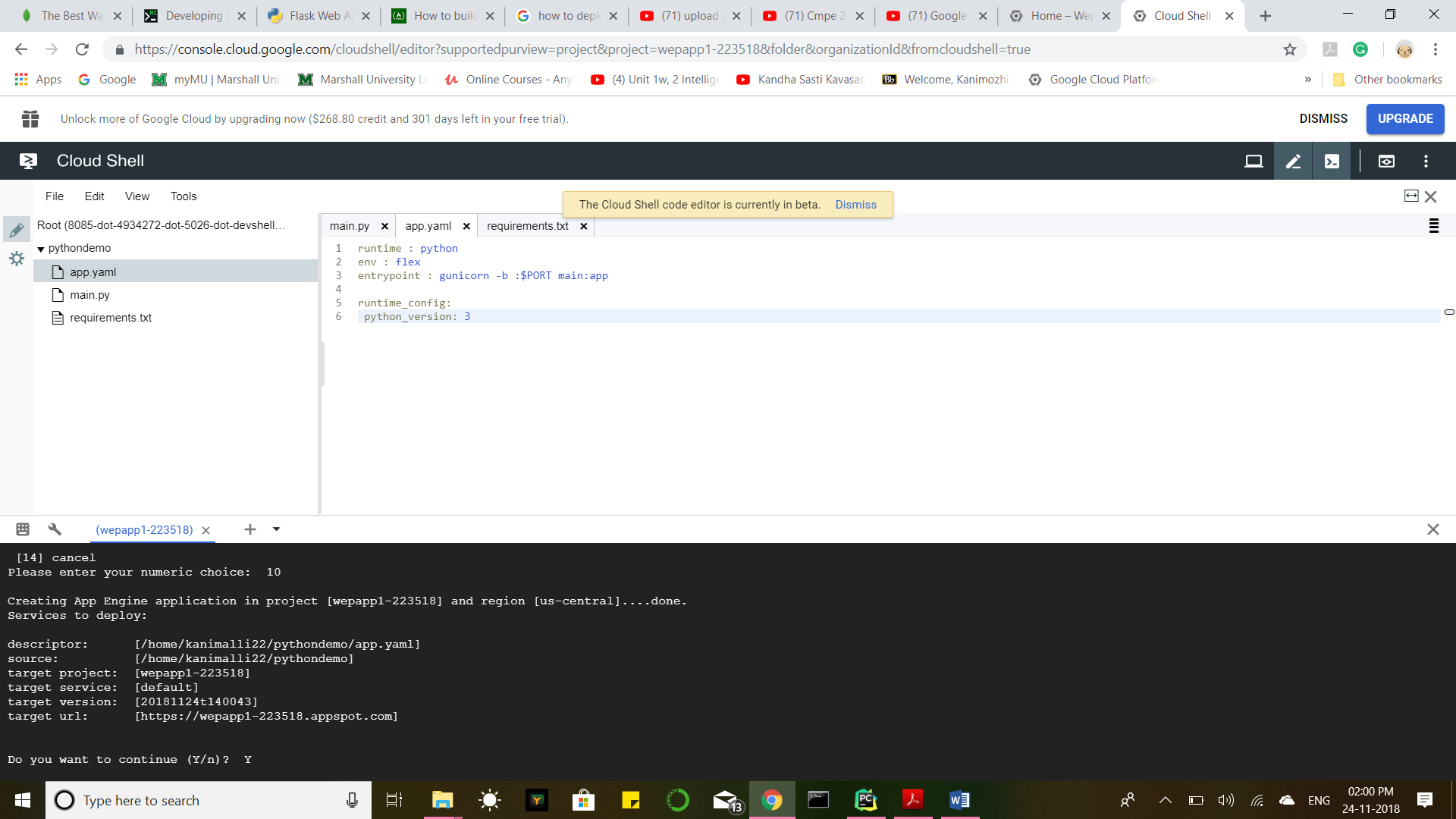
python\_version: 3

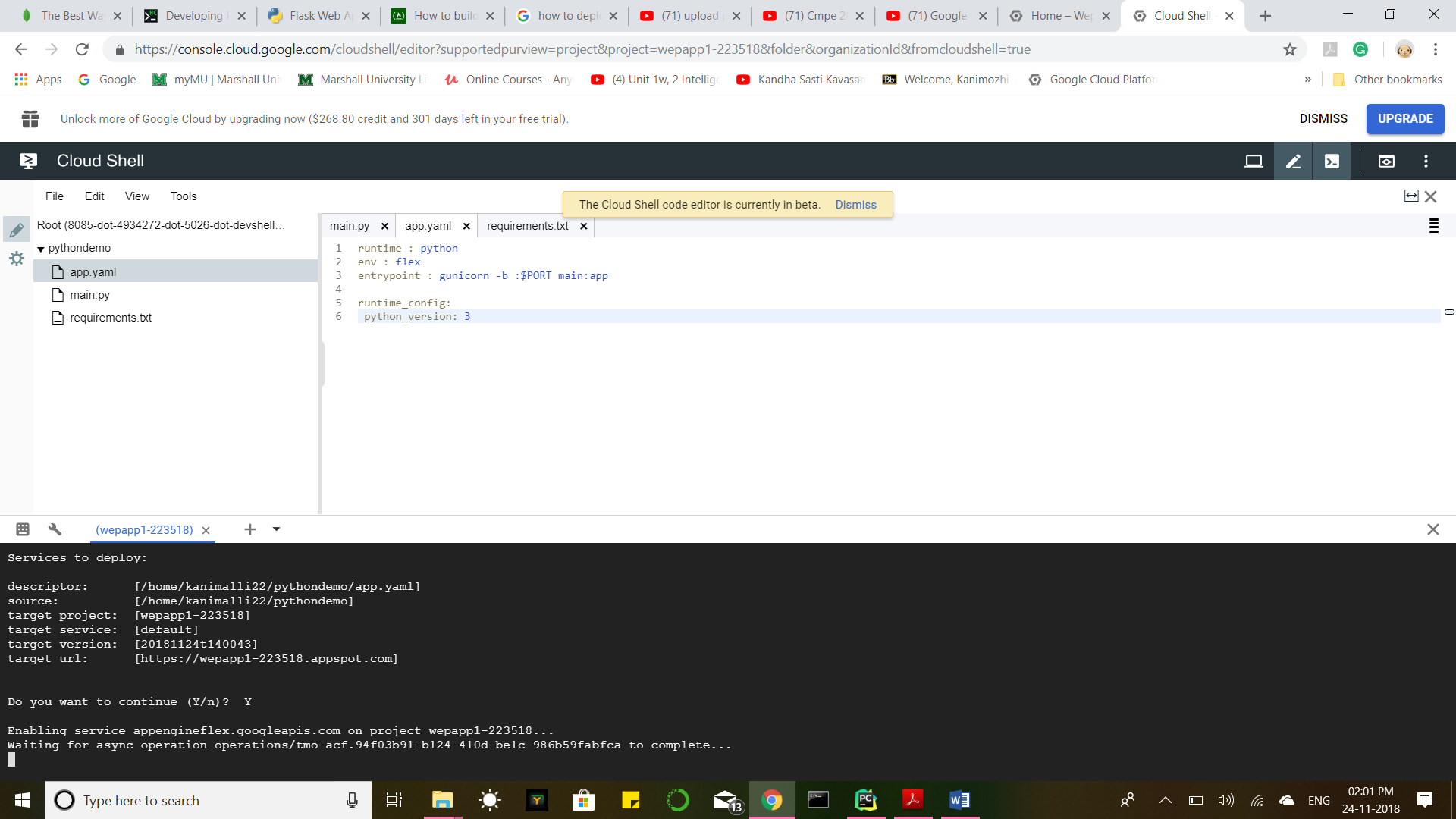




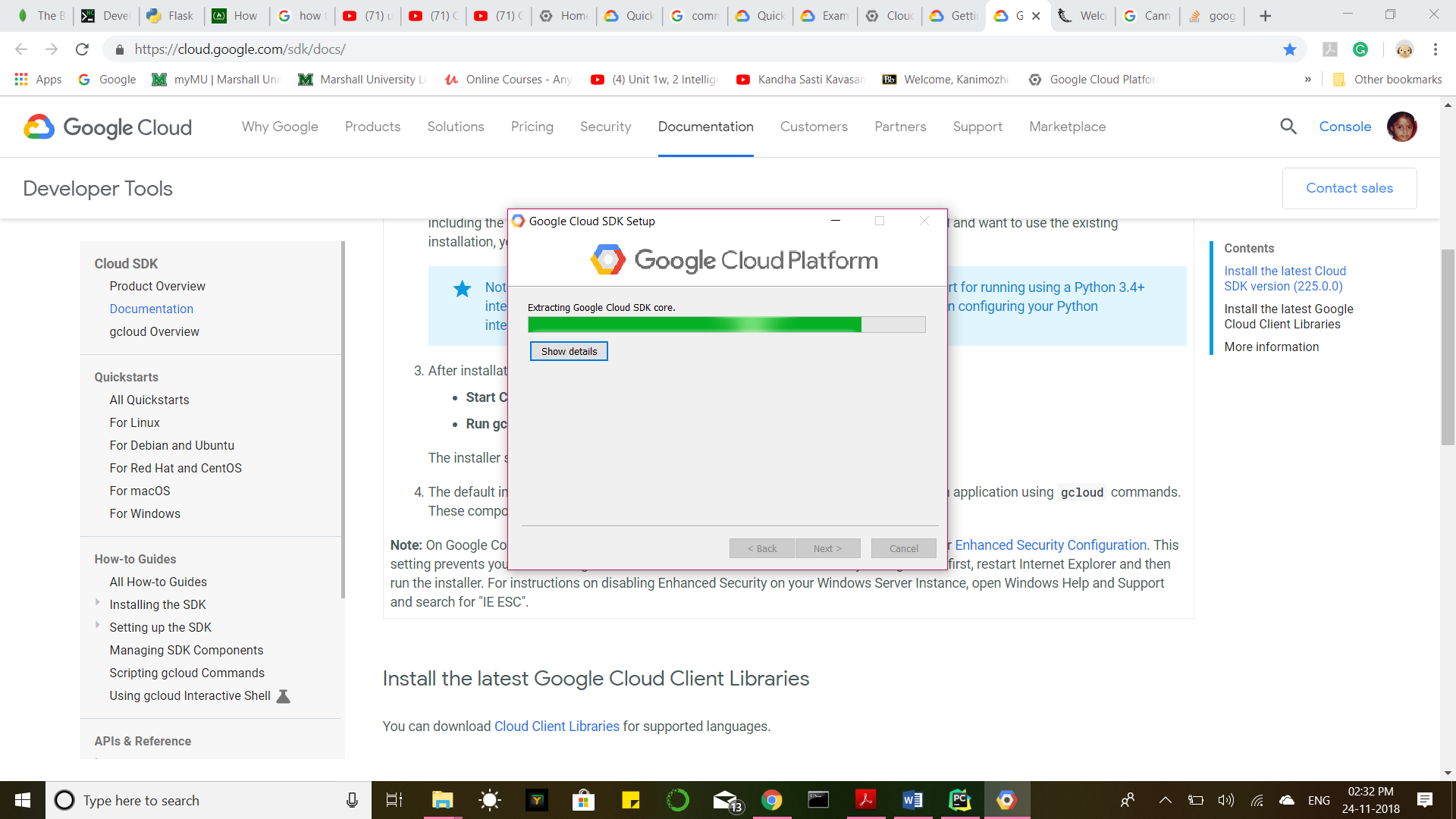
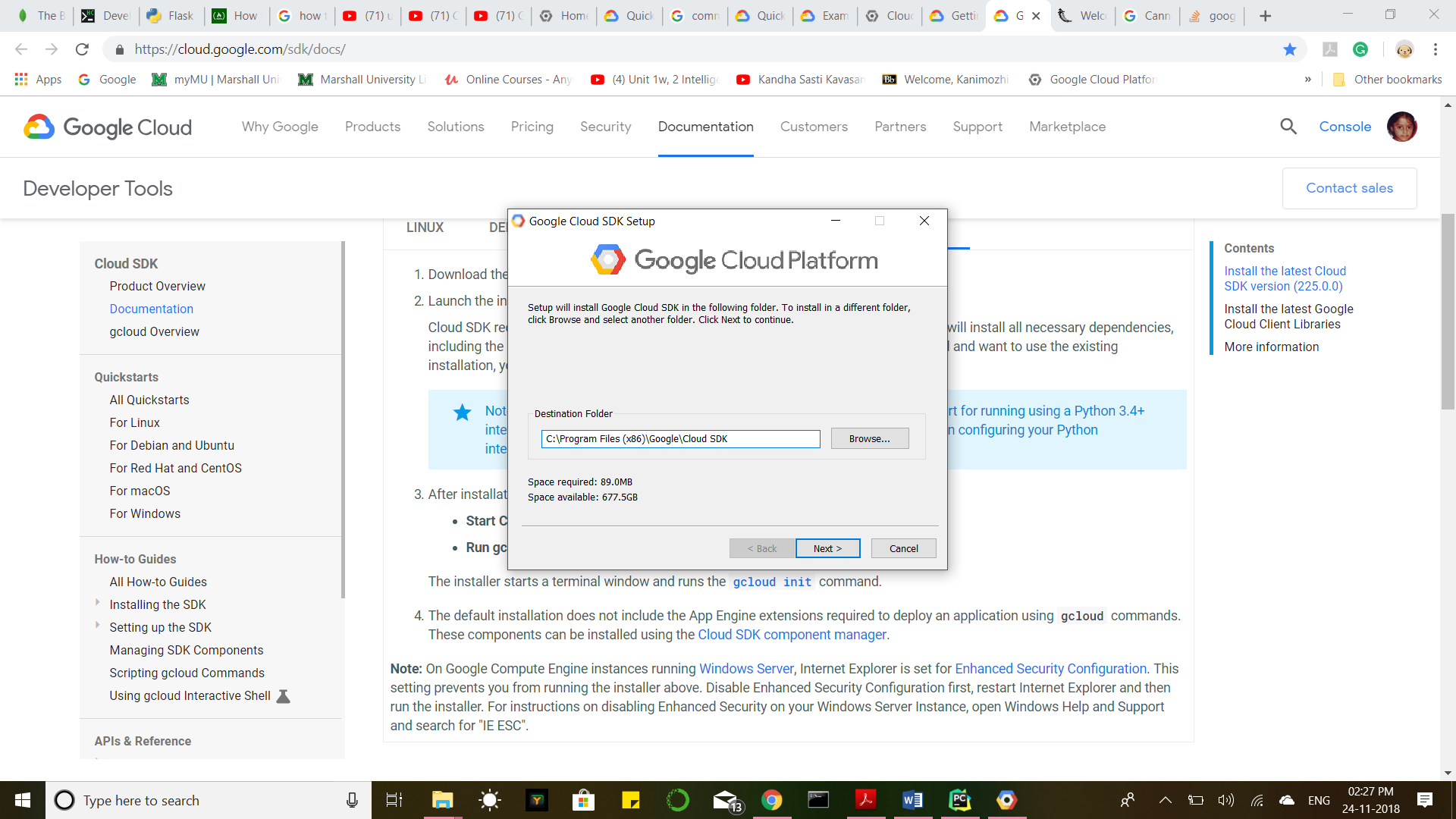


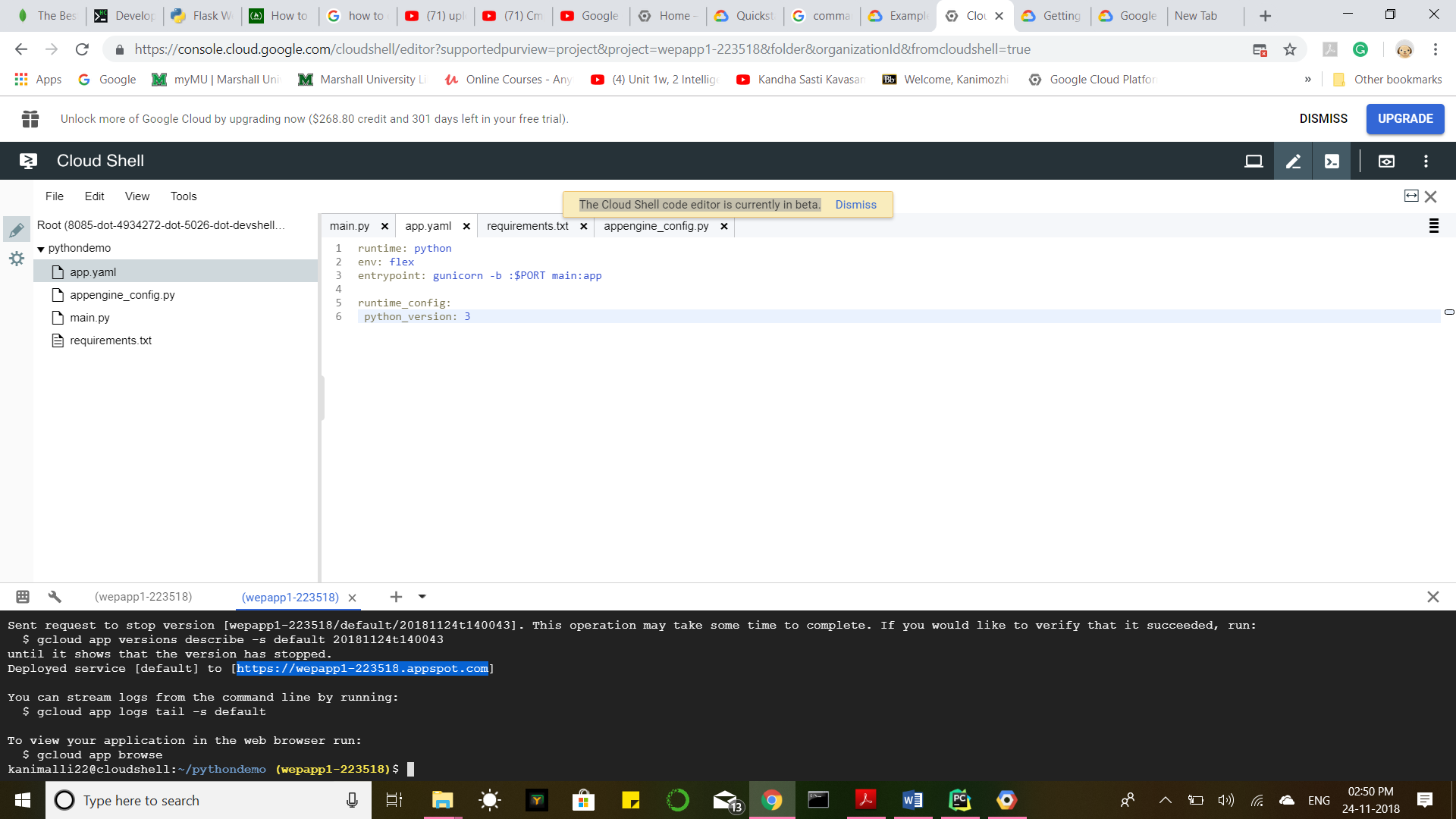






Google Cloud SDK Installation:

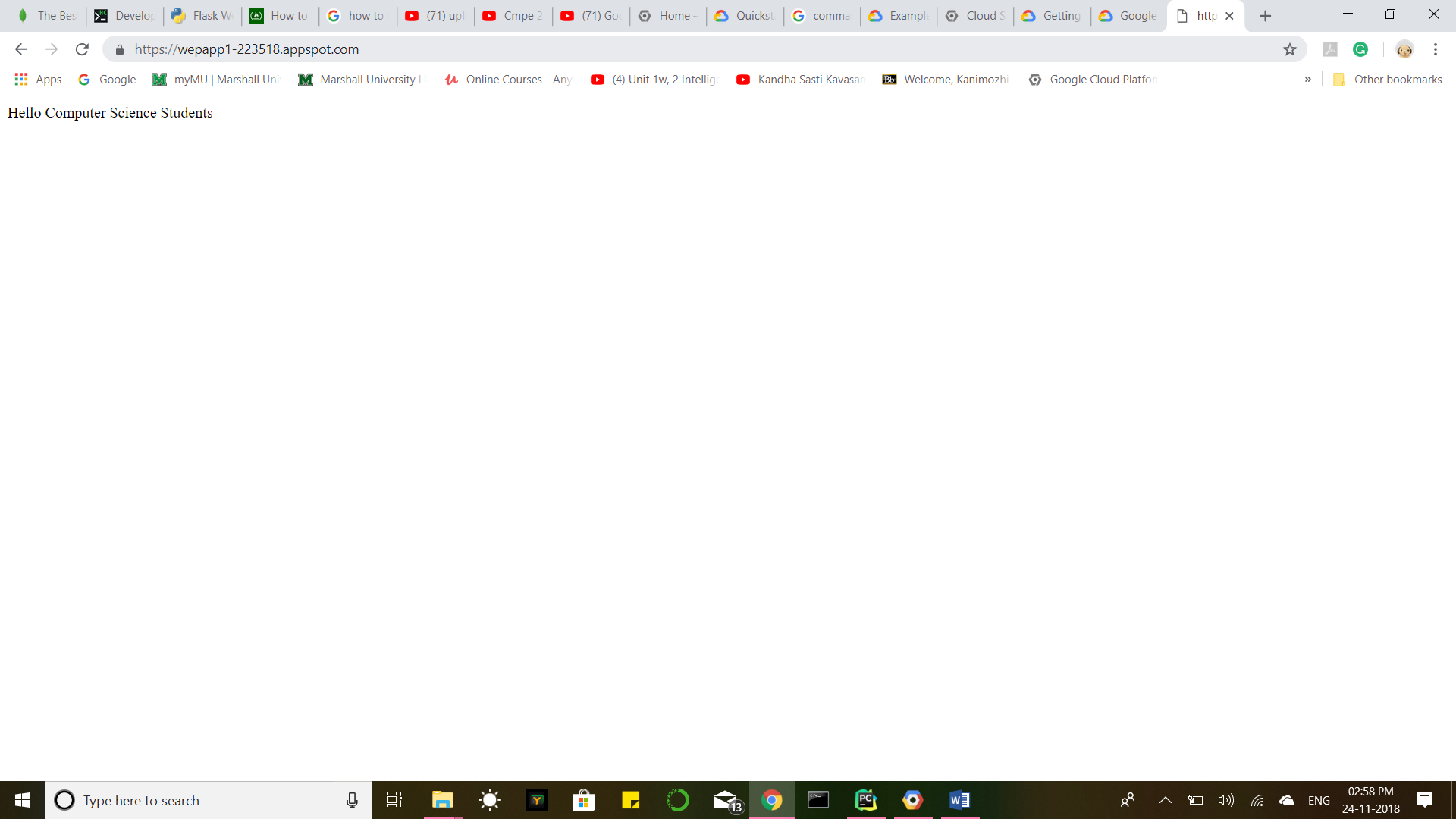




https://wepapp1-223518.appspot.com

https://wepapp1-223518.appspot.com

https://wepapp1-223518.appspot.com/



from flask import Flask, render\_template, request  
from pymongo import MongoClient  
from pprint import pprint  
import pprintjson  
  
  
app = Flask(\_\_name\_\_)  
  
  
@app.route('/')  
def main():  
 return render\_template('homepage.html')  
  
  
@app.route('/signin', methods=['POST'])  
def signin():  
 # Retrieve the HTTP POST request parameter value from 'request.form' dictionary  
 \_username = request.form.get('username') # get(attr) returns None if attr is not present  
 \_password = request.form.get('password')  
  
 # Validate and send response  
 if \_username == 'kani' and \_password == 'Terence04':  
 return render\_template('signin.html', username=\_username)  
 else:  
 return render\_template('notauthorized.html') # 400 Bad Request  
  
  
@app.route('/search', methods=['POST'])  
def search():  
 if request.form['search\_button'] == 'Search':  
 return render\_template('search.html')  
 else:  
 pass  
  
  
@app.route('/insert', methods=['POST'])  
def insert():  
 if request.form['insert\_button'] == 'Insert':  
 return render\_template('insert.html')  
 else:  
 pass  
  
  
@app.route('/update', methods=['POST'])  
def update():  
 if request.form['update\_button'] == 'Update':  
 return render\_template('update.html')  
 else:  
 pass  
  
  
@app.route('/delete', methods=['POST'])  
def delete():  
 if request.form['delete\_button'] == 'Delete':  
 return render\_template('delete.html')  
 else:  
 pass  
  
  
@app.route('/searching', methods=['POST'])  
def searching():  
 # Retrieve the HTTP POST request parameter value from 'request.form' dictionary  
 mykey1 = request.form.get('searchkey') # get(attr) returns None if attr is not present  
 myvalue1 = request.form.get('searchvalue')  
  
 client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")  
 print(client)  
 db = client.restaurant  
 d = {}  
 dlist = []  
 mydocs = db.docs.find({mykey1:myvalue1})  
 for x in mydocs:  
 print(x)  
 d['data'] = x  
 dlist.append(d.copy())  
 print(dlist)  
  
 if request.method == 'POST':  
 return render\_template("searchoutput.html",mydocs=dlist)  
  
  
@app.route('/inserting', methods=['POST'])  
def inserting():  
 # Retrieve the HTTP POST request parameter value from 'request.form' dictionary  
 mykey1 = request.form.get('insertkey') # get(attr) returns None if attr is not present  
 myvalue1 = request.form.get('insertvalue')  
  
 client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")  
 print(client)  
 db = client.restaurant  
 mydict = {mykey1: myvalue1}  
 x = db.docs.insert\_one(mydict)  
  
 d = {}  
 dlist = []  
 mydocs = db.docs.find({mykey1:myvalue1})  
 for x in mydocs:  
 print(x)  
 d['data'] = x  
 dlist.append(d.copy())  
 print(dlist)  
  
 if request.method == 'POST':  
 return render\_template("insertoutput.html",mydocs=dlist)  
  
@app.route('/updating', methods=['POST'])  
def updating():  
 # Retrieve the HTTP POST request parameter value from 'request.form' dictionary  
 mykey1 = request.form.get('updatekey') # get(attr) returns None if attr is not present  
 myvalue1 = request.form.get('updatevalue')  
 mychangekey = request.form.get('newkey')  
 mychangevalue = request.form.get('newvalue')  
  
 client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")  
 print(client)  
 db = client.restaurant  
 myquery = {mykey1: myvalue1}  
 newvalues = {"$set": {mychangekey: mychangevalue}}  
  
 db.docs.update\_many(myquery, newvalues)  
  
 d = {}  
 dlist = []  
 mydocs = db.docs.find({mychangekey: mychangevalue})  
 for x in mydocs:  
 print(x)  
 d['data'] = x  
 dlist.append(d.copy())  
 print(dlist)  
  
 if request.method == 'POST':  
 return render\_template("updateoutput.html", mydocs=dlist)  
  
@app.route('/deleting', methods=['POST'])  
def deleting():  
 # Retrieve the HTTP POST request parameter value from 'request.form' dictionary  
 mykey1 = request.form.get('deletekey') # get(attr) returns None if attr is not present  
 myvalue1 = request.form.get('deletevalue')  
  
 client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")  
 print(client)  
 db = client.restaurant  
  
 myquery = {mykey1: myvalue1}  
 db.docs.delete\_one(myquery)  
  
 d = {}  
 dlist = []  
 mydocs = db.docs.find({mykey1: myvalue1})  
 for x in mydocs:  
 print(x)  
 d['data'] = x  
 dlist.append(d.copy())  
 print(dlist)  
  
 if request.method == 'POST':  
 return render\_template("deleteoutput.html", mydocs=dlist)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

from flask import Flask, render\_template, request

from pymongo import MongoClient

from pprint import pprint

import pprintjson

app = Flask(\_\_name\_\_)

@app.route('/')

def main():

return render\_template('homepage.html')

@app.route('/signin', methods=['POST'])

def signin():

# Retrieve the HTTP POST request parameter value from 'request.form' dictionary

\_username = request.form.get('username') # get(attr) returns None if attr is not present

\_password = request.form.get('password')

# Validate and send response

if \_username == 'kani' and \_password == 'Terence04':

return render\_template('signin.html', username=\_username)

else:

return render\_template('notauthorized.html') # 400 Bad Request

@app.route('/search', methods=['POST'])

def search():

if request.form['search\_button'] == 'Search':

return render\_template('search.html')

else:

pass

@app.route('/insert', methods=['POST'])

def insert():

if request.form['insert\_button'] == 'Insert':

return render\_template('insert.html')

else:

pass

@app.route('/update', methods=['POST'])

def update():

if request.form['update\_button'] == 'Update':

return render\_template('update.html')

else:

pass

@app.route('/delete', methods=['POST'])

def delete():

if request.form['delete\_button'] == 'Delete':

return render\_template('delete.html')

else:

pass

@app.route('/searching', methods=['POST'])

def searching():

# Retrieve the HTTP POST request parameter value from 'request.form' dictionary

mykey1 = request.form.get('searchkey') # get(attr) returns None if attr is not present

myvalue1 = request.form.get('searchvalue')

client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")

print(client)

db = client.restaurant

d = {}

dlist = []

mydocs = db.docs.find({mykey1:myvalue1})

for x in mydocs:

print(x)

d['data'] = x

dlist.append(d.copy())

print(dlist)

if request.method == 'POST':

return render\_template("searchoutput.html",mydocs=dlist)

@app.route('/inserting', methods=['POST'])

def inserting():

# Retrieve the HTTP POST request parameter value from 'request.form' dictionary

mykey1 = request.form.get('insertkey') # get(attr) returns None if attr is not present

myvalue1 = request.form.get('insertvalue')

client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")

print(client)

db = client.restaurant

mydict = {mykey1: myvalue1}

x = db.docs.insert\_one(mydict)

d = {}

dlist = []

mydocs = db.docs.find({mykey1:myvalue1})

for x in mydocs:

print(x)

d['data'] = x

dlist.append(d.copy())

print(dlist)

if request.method == 'POST':

return render\_template("insertoutput.html",mydocs=dlist)

@app.route('/updating', methods=['POST'])

def updating():

# Retrieve the HTTP POST request parameter value from 'request.form' dictionary

mykey1 = request.form.get('updatekey') # get(attr) returns None if attr is not present

myvalue1 = request.form.get('updatevalue')

mychangekey = request.form.get('newkey')

mychangevalue = request.form.get('newvalue')

client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")

print(client)

db = client.restaurant

myquery = {mykey1: myvalue1}

newvalues = {"$set": {mychangekey: mychangevalue}}

db.docs.update\_many(myquery, newvalues)

d = {}

dlist = []

mydocs = db.docs.find({mychangekey: mychangevalue})

for x in mydocs:

print(x)

d['data'] = x

dlist.append(d.copy())

print(dlist)

if request.method == 'POST':

return render\_template("updateoutput.html", mydocs=dlist)

@app.route('/deleting', methods=['POST'])

def deleting():

# Retrieve the HTTP POST request parameter value from 'request.form' dictionary

mykey1 = request.form.get('deletekey') # get(attr) returns None if attr is not present

myvalue1 = request.form.get('deletevalue')

client = MongoClient("mongodb+srv://kani:Terence04@clustermongodb-xwcjz.gcp.mongodb.net/test?retryWrites=true")

print(client)

db = client.restaurant

myquery = {mykey1: myvalue1}

db.docs.delete\_one(myquery)

d = {}

dlist = []

mydocs = db.docs.find({mykey1: myvalue1})

for x in mydocs:

print(x)

d['data'] = x

dlist.append(d.copy())

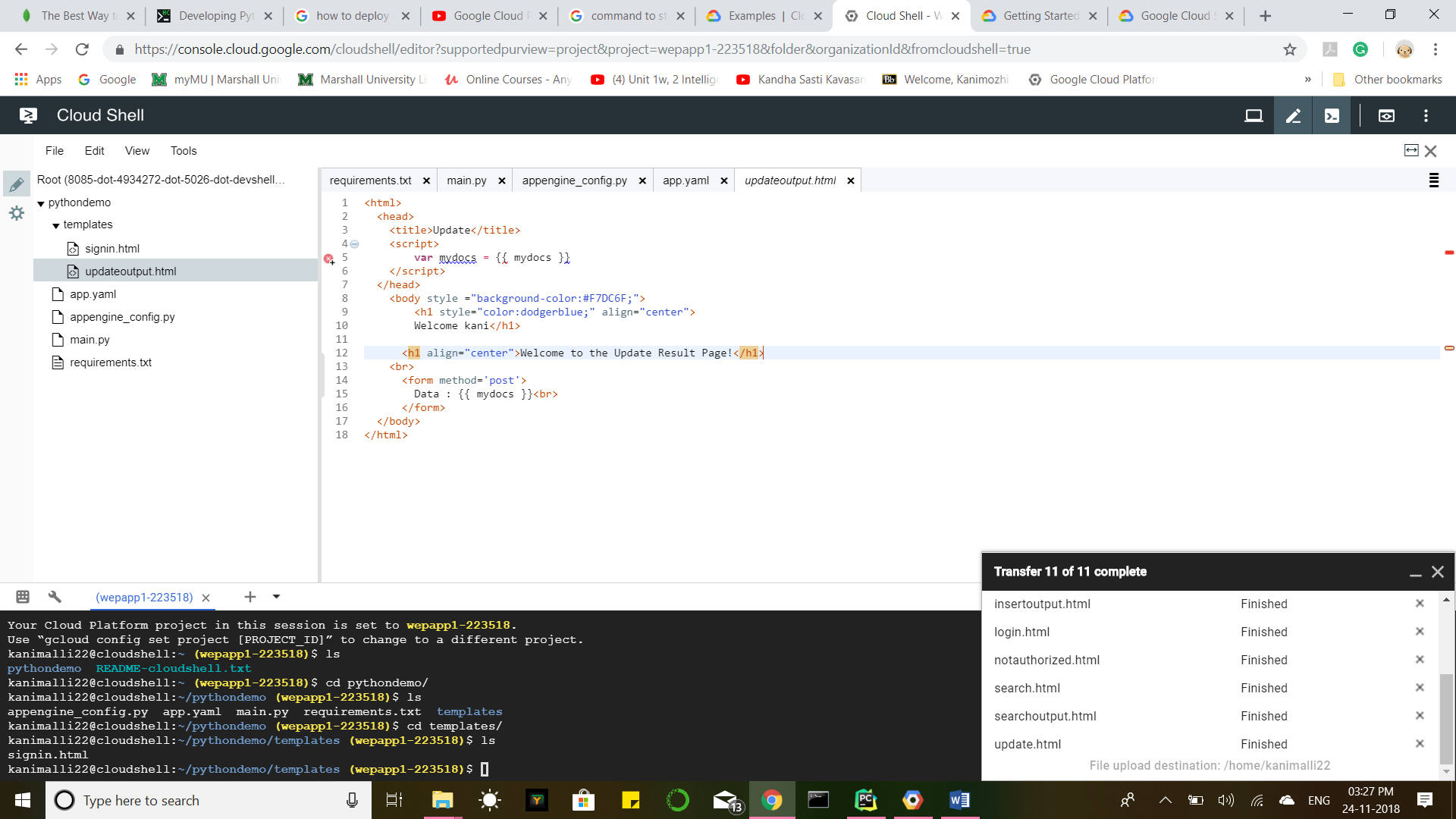
print(dlist)

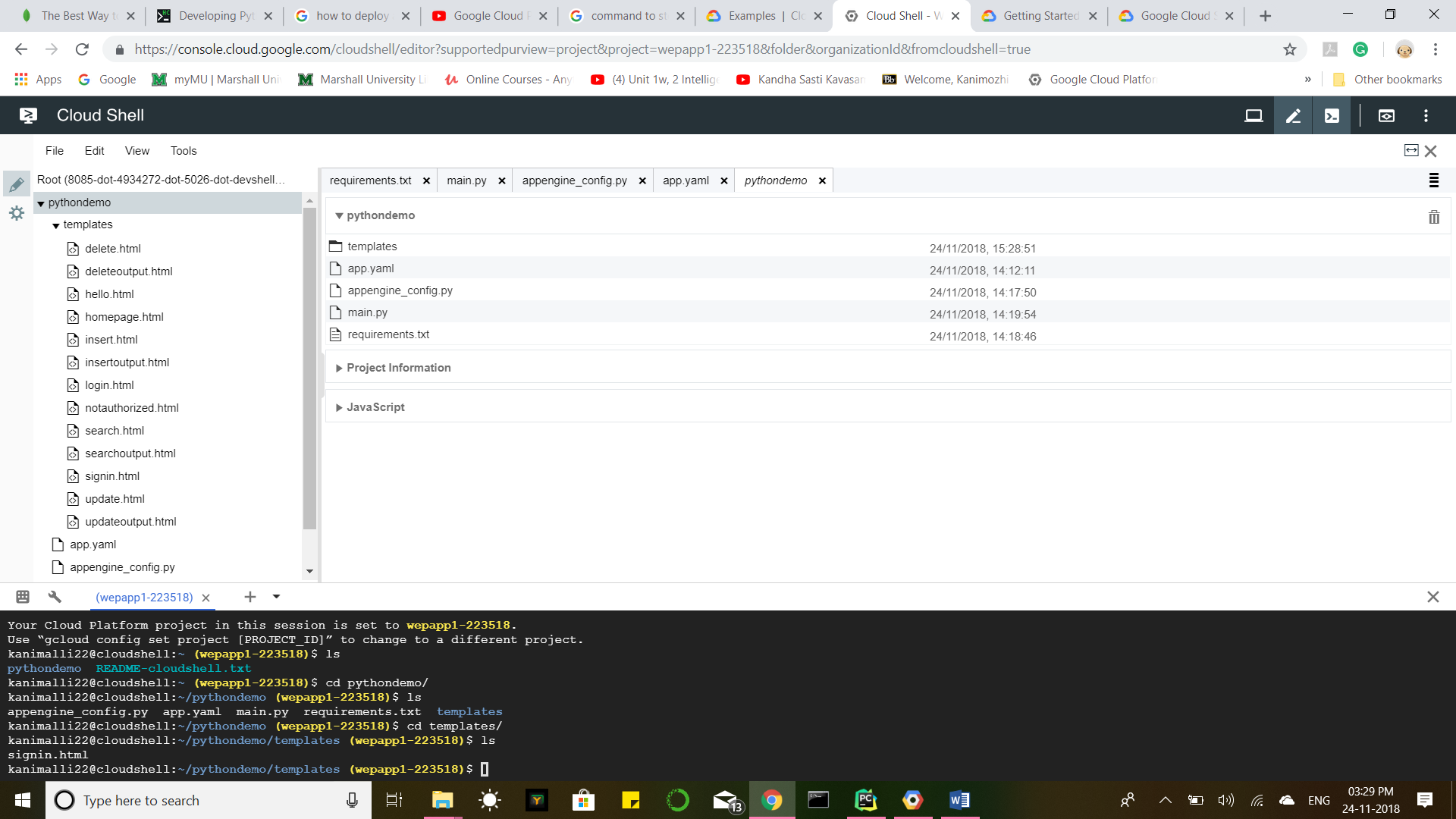
if request.method == 'POST':

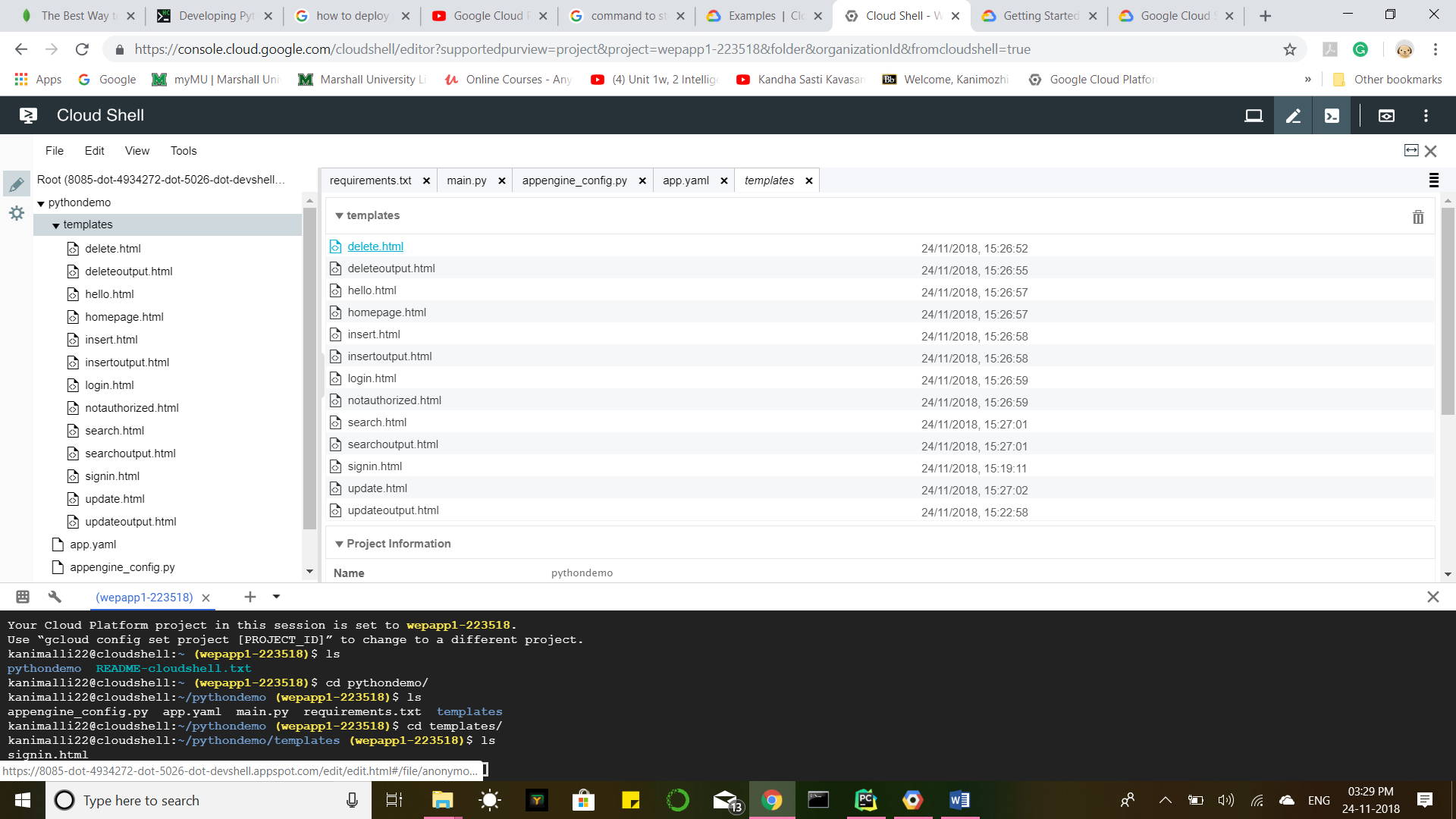
return render\_template("deleteoutput.html", mydocs=dlist)

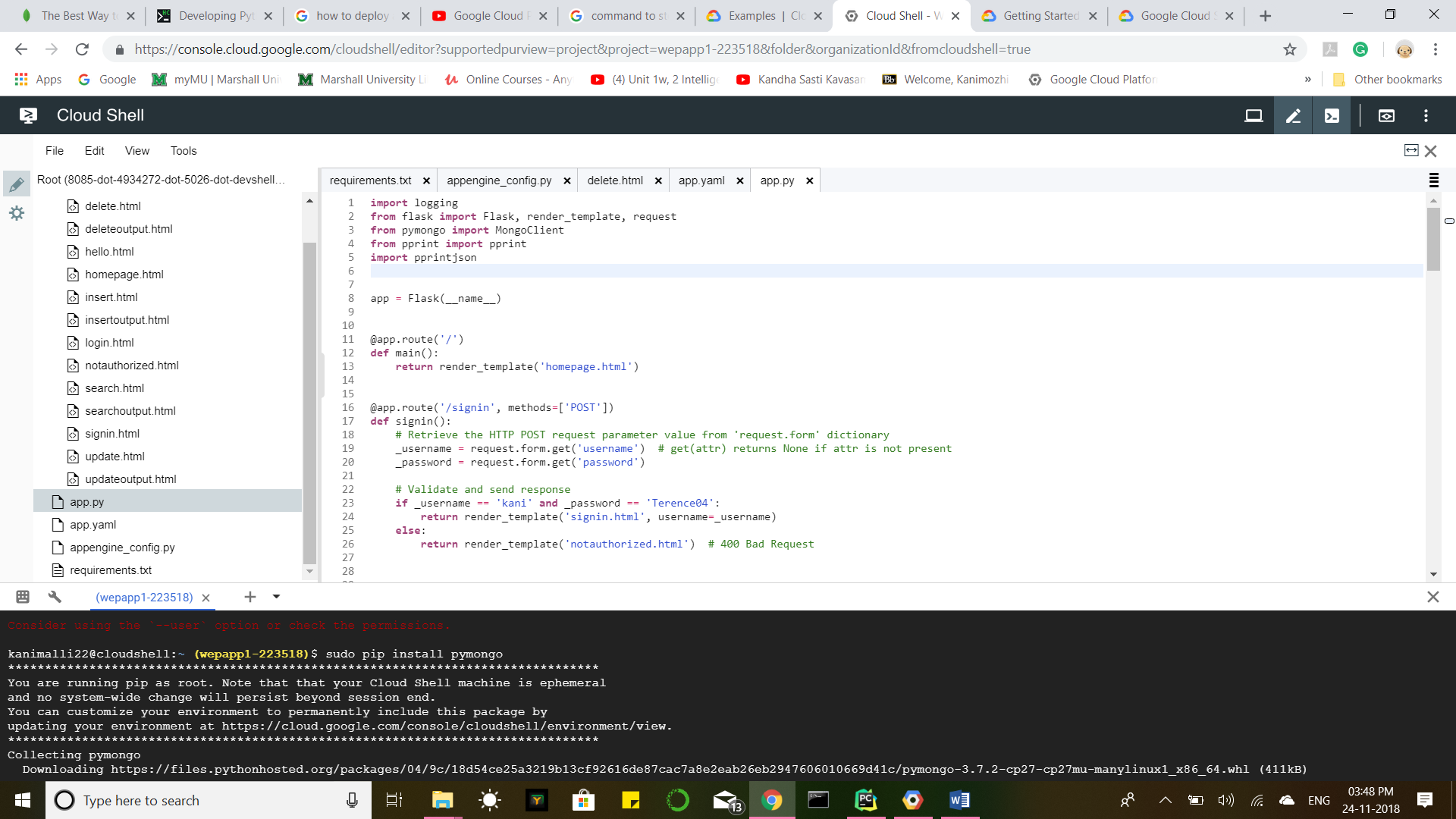
if \_\_name\_\_ == '\_\_main\_\_':

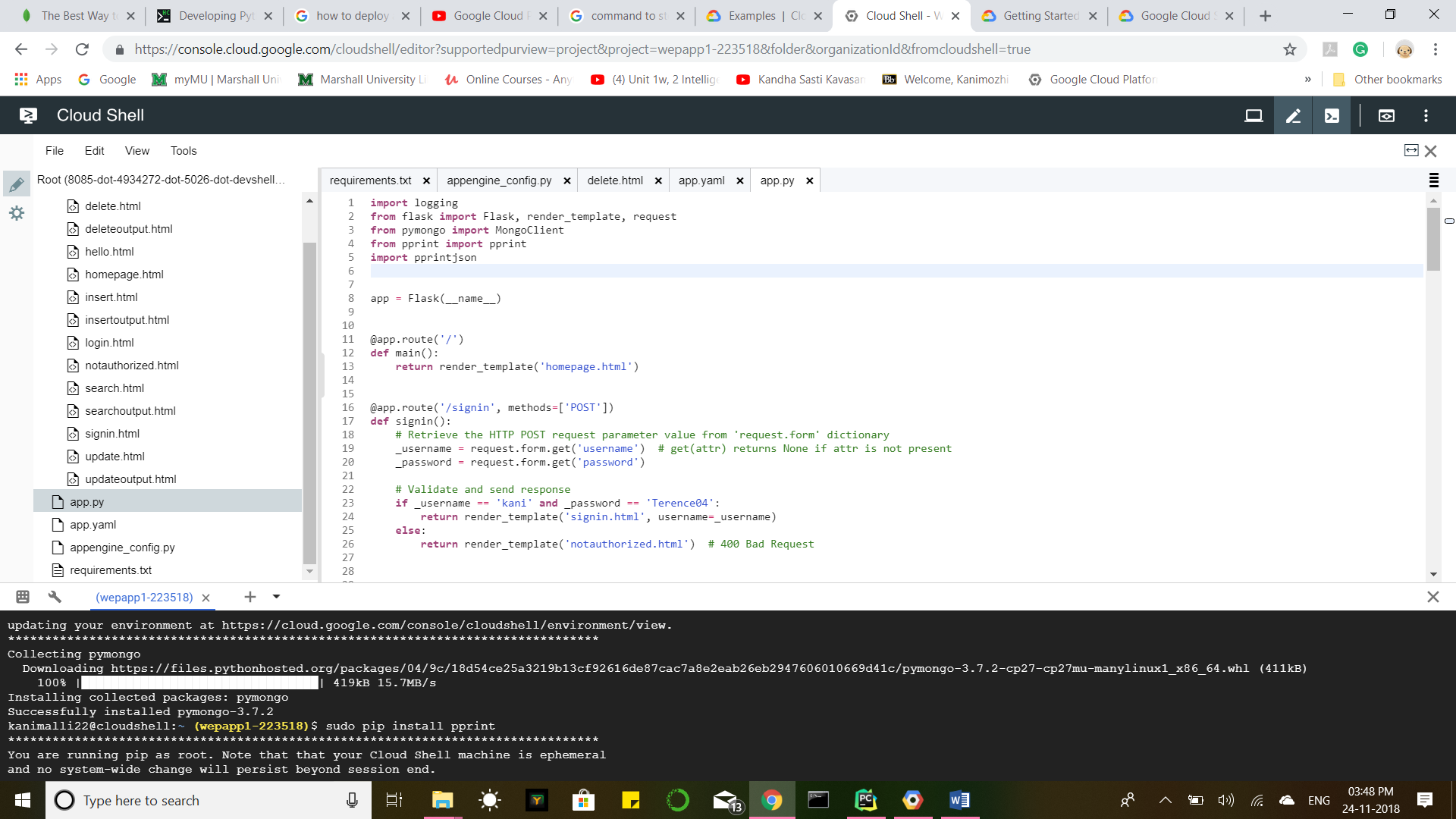
app.run(debug=True)











mongodb://kani:Terence04@clustermongodb-shard-00-00-xwcjz.gcp.mongodb.net:27017,clustermongodb-shard-00-01-xwcjz.gcp.mongodb.net:27017,clustermongodb-shard-00-02-xwcjz.gcp.mongodb.net:27017/test?ssl=true&replicaSet=ClusterMongoDB-shard-0&authSource=admin&retryWrites=true