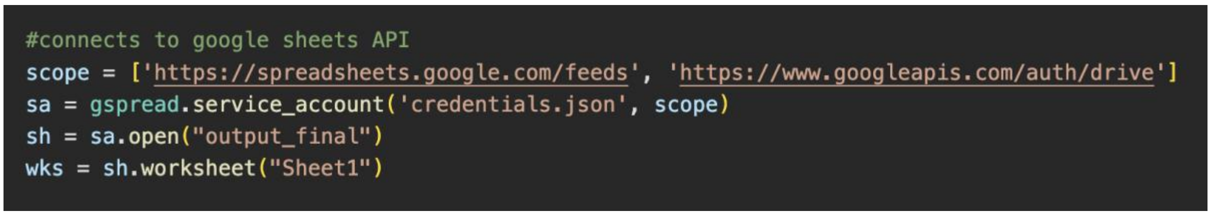
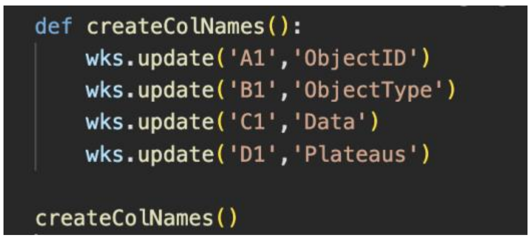


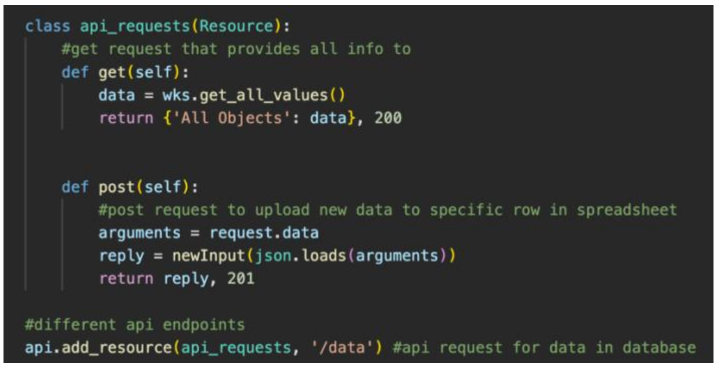
```
#connects to google sheets API
scope = ['https://spreadsheets.google.com/feeds', 'https://www.googleapis.com/auth/drive']
sa = gspread.service_account('credentials.json', scope)
sh = sa.open("output_final")
wks = sh.worksheet("Sheet1")
```

The first part of the code in the document, establishes a connection to the google sheets API and opens the worksheet as **wks**.



```
def createColNames():
    wks.update('A1','ObjectID')
    wks.update('B1','ObjectType')
    wks.update('C1','Data')
    wks.update('D1','Plateaus')

createColNames()
```

The method updates the specified values in the google sheets and inserts the columns names.

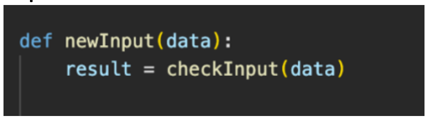


```
class api_requests(Resource):
    #get request that provides all info to
    def get(self):
        data = wks.get_all_values()
        return {'All Objects': data}, 200


    def post(self):
        #post request to upload new data to specific row in spreadsheet
        arguments = request.data
        reply = newInput(json.loads(arguments))
        return reply, 201

#different api endpoints
api.add_resource(api_requests, '/data') #api request for data in database
```

The class api_requests, contains handles the API endpoint for the application. When the user calls the api with a get request, all data from the google worksheet is being returned in a json format

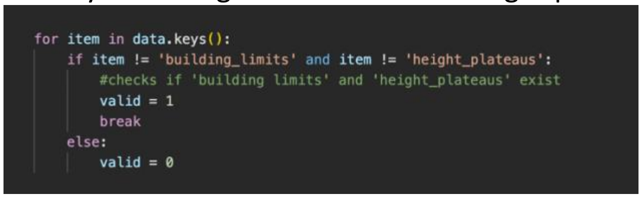
When a post request is called, the input of the post body is stored and then forwarded to the ‘newInput’ method.

The ‘newInput’ function first calls the checkInput function which task it is to check if the input is valid



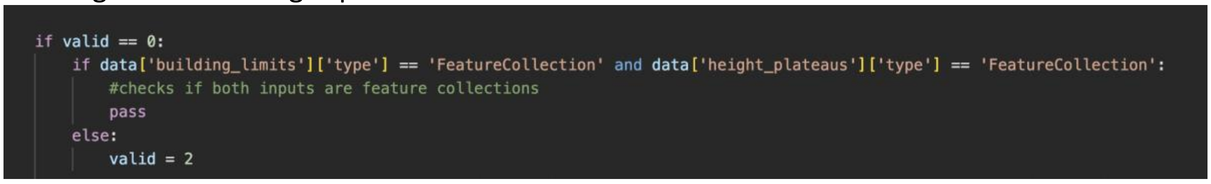
```python
def newInput(data):
    result = checkInput(data)
```

Now we will get to the different parts of the checkInput function. This first for loop checks if the keys ‘building limits’ as well as ‘height plateaus’ exist in the data that has been input.



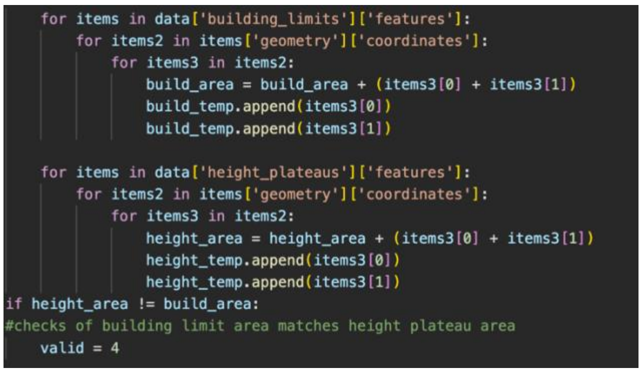
```python
for item in data.keys():
    if item != 'building_limits' and item != 'height_plateaus':
        #checks if 'building limits' and 'height_plateaus' exist
        valid = 1
        break
    else:
        valid = 0
```

The next part of the code checks if the type of the input is a collection of features for both, building limits and height plateaus.s



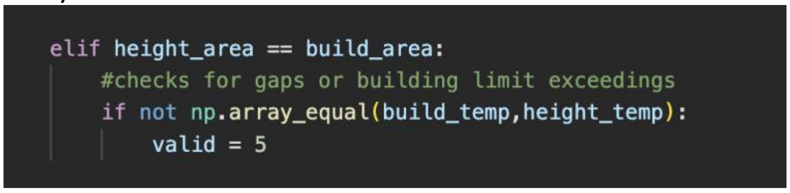
```python
if valid == 0:
    if data['building_limits']['type'] == 'FeatureCollection' and data['height_plateaus']['type'] == 'FeatureCollection':
        #checks if both inputs are feature collections
        pass
    else:
        valid = 2
```

The next part retrieves all individual coordinates and stores them in a temporary array. At the end the sum is calculated and compared between height plateaus and building limit.



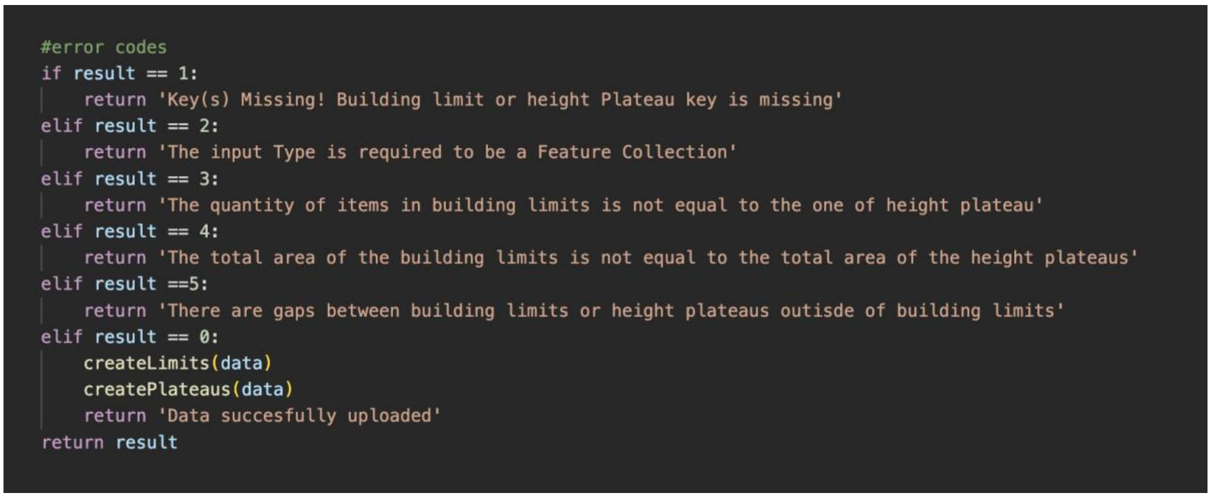
```python
    for items in data['building_limits']['features']:
        for items2 in items['geometry']['coordinates']:
            for items3 in items2:
                build_area = build_area + (items3[0] + items3[1])
                build_temp.append(items3[0])
                build_temp.append(items3[1])

    for items in data['height_plateaus']['features']:
        for items2 in items['geometry']['coordinates']:
            for items3 in items2:
                height_area = height_area + (items3[0] + items3[1])
                height_temp.append(items3[0])
                height_temp.append(items3[1])
if height_area != build_area:
#checks of building limit area matches height plateau area
    valid = 4
```

This following method checks for gaps by comparing the values within the 2 coordinate arrays.



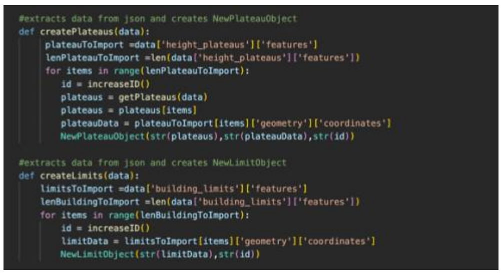
```python
elif height_area == build_area:
    #checks for gaps or building limit exceedings
    if not np.array_equal(build_temp,height_temp):
        valid = 5
```

The result of ‚valid‘ is then returned to ‚newInput‘ and interpreted accordingly to the value it returned. Based on this the json data is then either uploaded or an error is returned to the user



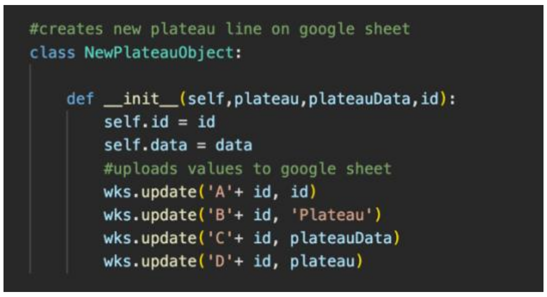
```
#error codes
if result == 1:
    return 'Key(s) Missing! Building limit or height Plateau key is missing'
elif result == 2:
    return 'The input Type is required to be a Feature Collection'
elif result == 3:
    return 'The quantity of items in building limits is not equal to the one of height plateau'
elif result == 4:
    return 'The total area of the building limits is not equal to the total area of the height plateaus'
elif result ==5:
    return 'There are gaps between building limits or height plateaus outisde of building limits'
elif result == 0:
    createLimits(data)
    createPlateaus(data)
    return 'Data succesfully uploaded'
return result
```

We the have two methods that extract all building limits as well as all height plateaus. At the end of this function a new object is created



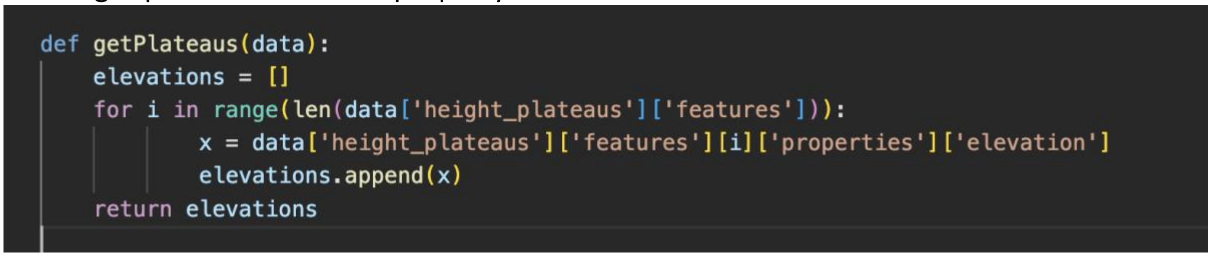
```
#extracts data from json and creates NewPlateauObject
def createPlateaus(data):
    plateauToImport =data['height_plateaus']['features']
    lenPlateauToImport =len(data['height_plateaus']['features'])
    for items in range(lenPlateauToImport):
        id = increaseID()
        plateaus = getPlateaus(data)
        plateaus = plateaus[items]
        plateauData = plateauToImport[items]['geometry']['coordinates']
        NewPlateauObject(str(plateaus),str(plateauData),str(id))

#extracts data from json and creates NewLimitObject
def createLimits(data):
    limitsToImport =data['building_limits']['features']
    lenBuildingToImport =len(data['building_limits']['features'])
    for items in range(lenBuildingToImport):
        id = increaseID()
        limitData = limitsToImport[items]['geometry']['coordinates']
        NewLimitObject(str(limitData),str(id))
```

With the object creation the data is being written to the google sheet



```
#creates new plateau line on google sheet
class NewPlateauObject:

    def __init__(self,plateau,plateauData,id):
        self.id = id
        self.data = data
        #uploads values to google sheet
        wks.update('A'+ id, id)
        wks.update('B'+ id, 'Plateau')
        wks.update('C'+ id, plateauData)
        wks.update('D'+ id, plateau)
```

The last function that is specific to the height plateaus, extracts the ‘elevation’ element from the height plateau ‘elevation’ property



```
def getPlateaus(data):
    elevations = []
    for i in range(len(data['height_plateaus']['features'])):
            x = data['height_plateaus']['features'][i]['properties']['elevation']
            elevations.append(x)
    return elevations
```