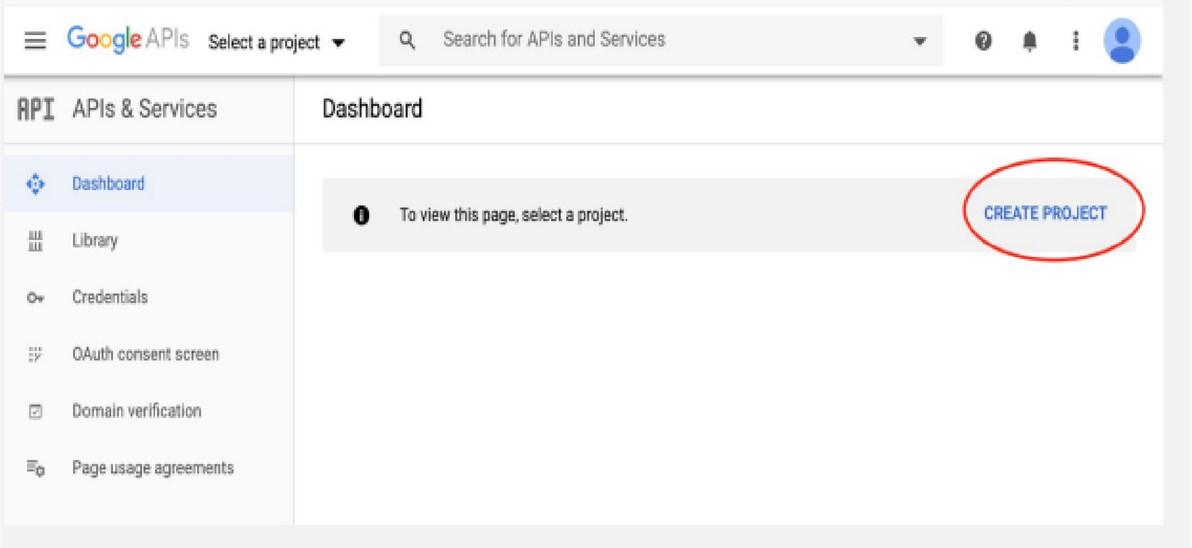
Real Time Data Transfer to Google Drive Using “Python”

To visualize our sensor(Ex: Ultrasonic sensor, Temperature sensor) data from any where , we are transferring the sensor data to our personal Google Drive.

For the above process, this are the following steps.

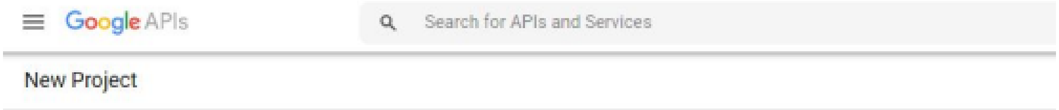
Step 1: First, we need to get the authentication files for Google Service API, so our Python code can access Google Drive. To do that, we need to :

i. Create a new project in **Google Developer Console**(website) by clicking “CREATE PROJECT” as follows.

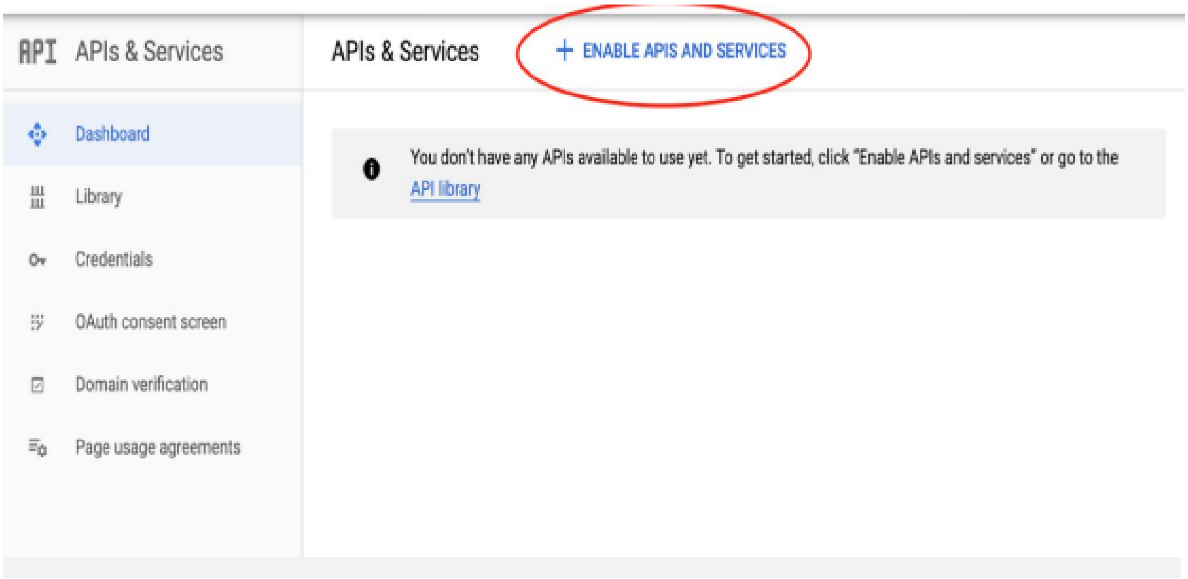


To view this page, select a project.

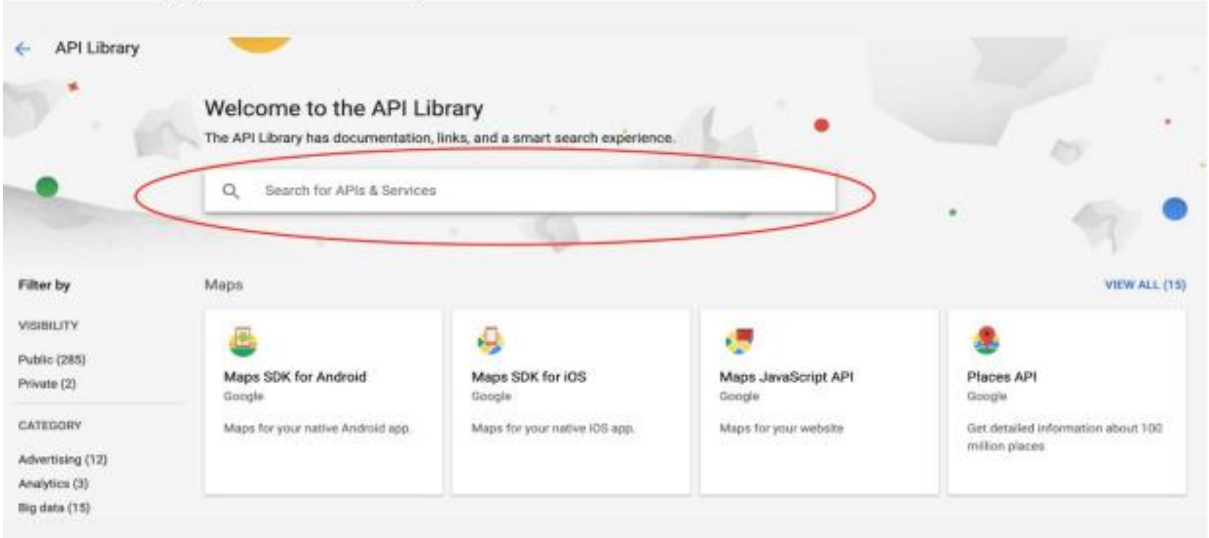
You can give your project a name or leave it as default.



ii. Enable APIs and Services by clicking the **“ENABLE APIS AND SERVICES”** as indicated by the red circle in the following picture.
That will bring you the API library as below



You don't have any APIs available to use yet. To get started, click "Enable APIs and services" or go to the API library



Welcome to the API Library

The API Library has documentation, links, and a smart search experience.

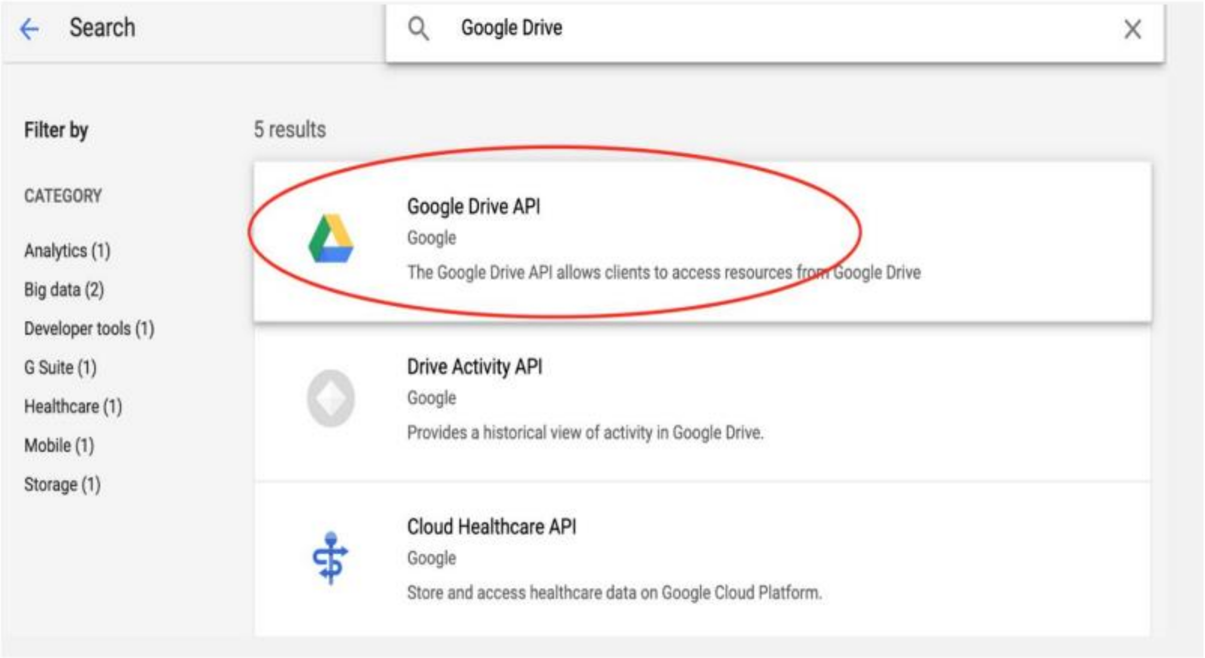
Maps for your native Android app.

Maps for your native iOS app.

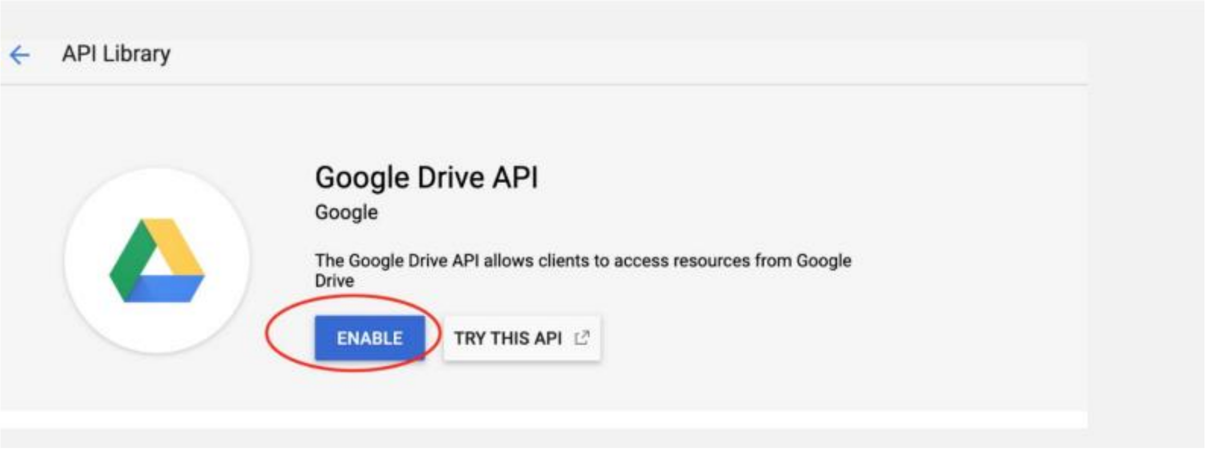
Maps for your website

Get detailed information about 100 million places

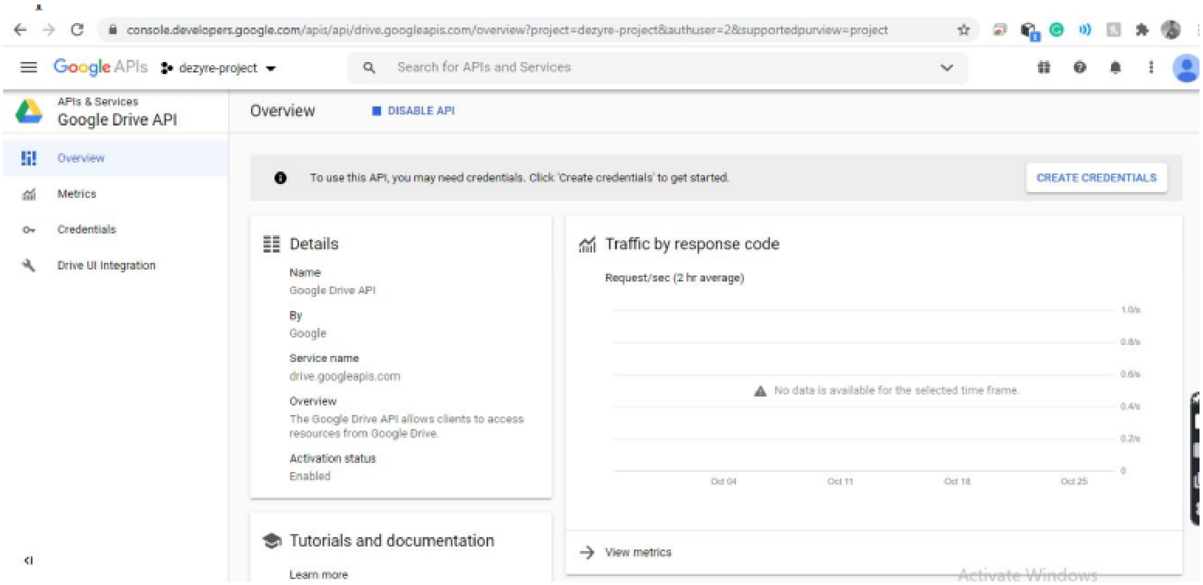
iii. Search "Google Drive" in the API library (indicated by the red circle in the picture). You'll get the following snapshot.



Click the “Google Drive API” icon and it will bring you to the next step as follows.


API Library
Google Drive API
Google
The Google Drive API allows clients to access resources from Google Drive
ENABLE
TRY THIS API

Then click “ENABLE”, which will enable your Google Drive API service. You’ll get to the next step as follows.



console.developers.google.com/apis/api/drive.googleapis.com/overview?project=dezyre-project&authuser=2&supportedpurview=project

To use this API, you may need credentials. Click 'Create credentials' to get started.

Details

Name
Google Drive API

By
Google

Service name
drive.googleapis.com

Overview
The Google Drive API allows clients to access resources from Google Drive.

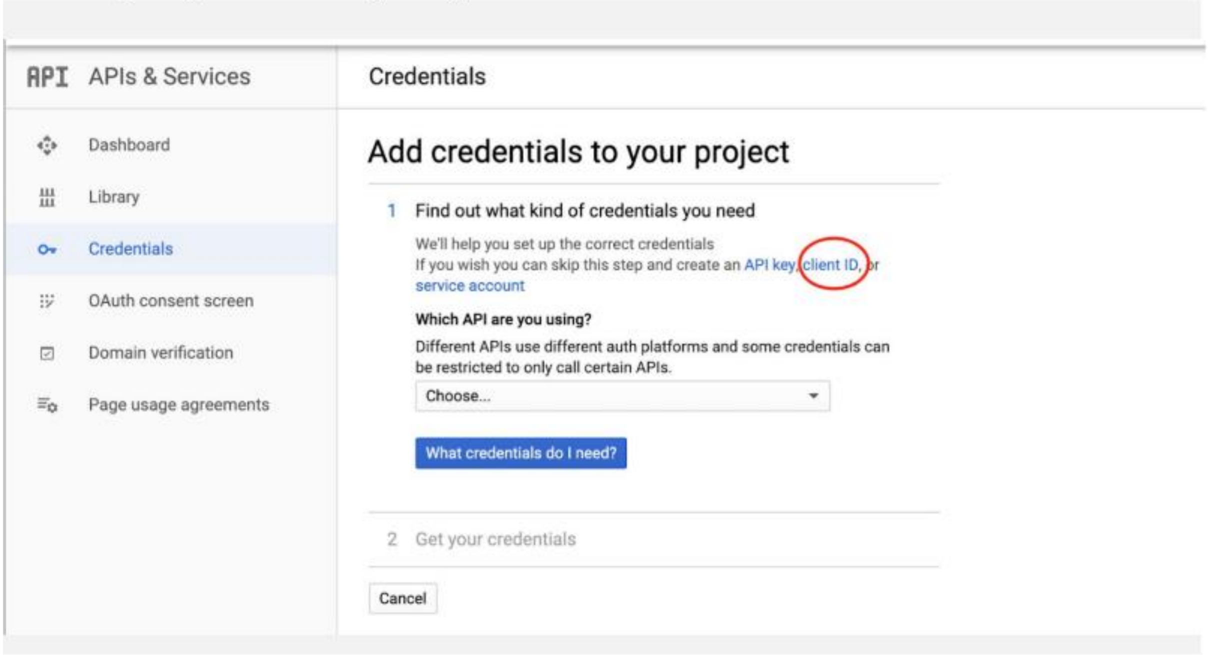
Activation status
Enabled

Traffic by response code

Request/sec (2 hr average)

No data is available for the selected time frame.

iv. Create credentials by clicking the “CREATE CREDENTIALS” icon (indicated by red circle in above snapshot). Here’s what you’ll get.



Add credentials to your project

1 Find out what kind of credentials you need

We'll help you set up the correct credentials
If you wish you can skip this step and create an API key, client ID, or service account

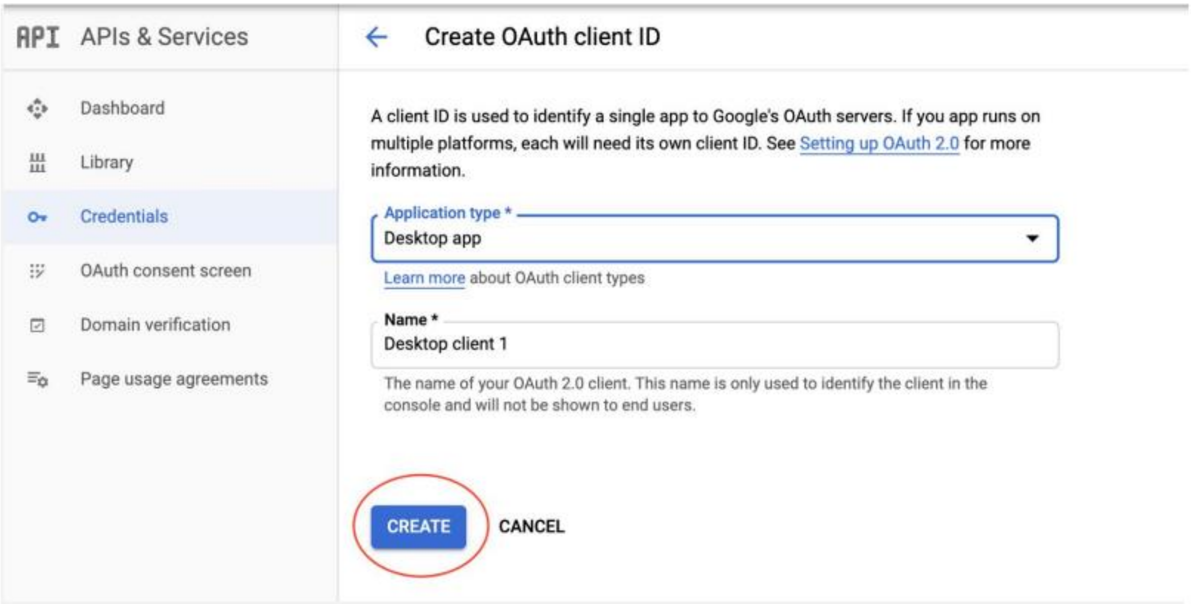
Which API are you using?

Different APIs use different auth platforms and some credentials can be restricted to only call certain APIs.

2 Get your credentials

In the above snapshot, we need to click “client ID” as that’s the Python program needs. Then click “CREATE” and download the JSON file as shown by the following snapshots

NOTE : In the above step sometimes it doesn’t show the client id options, at that time you just need to choose the Google drive in the choose options.



Create OAuth client ID

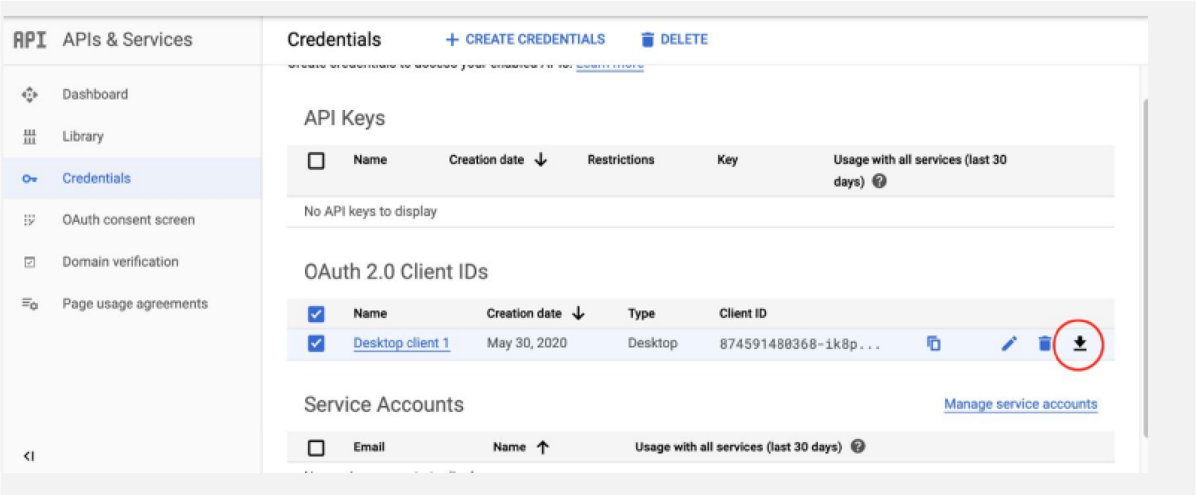
A client ID is used to identify a single app to Google's OAuth servers. If you app runs on multiple platforms, each will need its own client ID. See Setting up OAuth 2.0 for more information.

Application type * : Desktop app

Learn more about OAuth client types

Name * : Desktop client 1

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.



Credentials

API Keys

☐	Name	Creation date ↓	Restrictions	Key	Usage with all services (last 30 days)

No API keys to display

OAuth 2.0 Client IDs

☑	Name	Creation date ↓	Type	Client ID
☑	Desktop client 1	May 30, 2020	Desktop	874591480368-ik8p...

Service Accounts

☐	Email	Name ↑	Usage with all services (last 30 days)

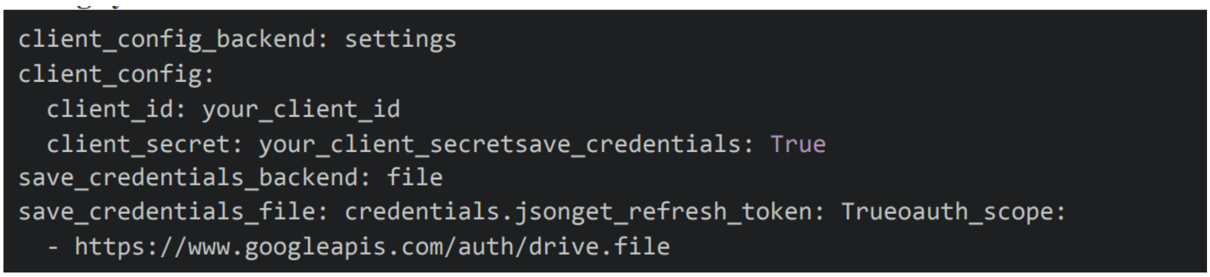
The downloaded JSON file is the one we need for our Python code to access Google Drive.

v. Now open the python Terminal and install the PyDrive Module.

By using this command : **pip install pydrive**

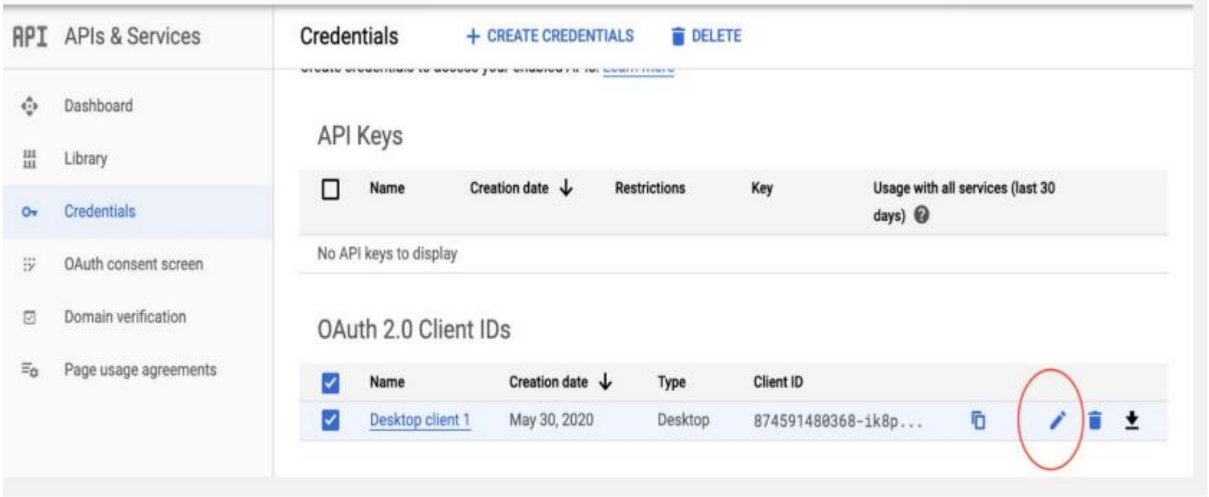
The google authentication code will do authentication and list all files in your Google Drive. Note that every time you run the following program, the code will open a web browser to ask you to input your Google account and password.

To avoid inputting passwords every time, we can create a **settings.yaml** file to save all the credentials.



```
client_config_backend: settings
client_config:
  client_id: your_client_id
  client_secret: your_client_secretsave_credentials: True
save_credentials_backend: file
save_credentials_file: credentials.jsonget_refresh_token: Trueoauth_scope:
  - https://www.googleapis.com/auth/drive.file
```

The client_id and client_secret can be found by clicking the editing icon in the following picture.



API Keys

□	Name	Creation date ↓	Restrictions	Key	Usage with all services (last 30 days)
No API keys to display					

OAuth 2.0 Client IDs

☑	Name	Creation date ↓	Type	Client ID
☑	Desktop client 1	May 30, 2020	Desktop	874591480368-ik8p...

The Python program will ask you to input your Google password again. Then it will create a credentials.json file. Next time, Python will just pick up that file to finish authentication automatically. Therefore, you don't need to type your password again.

Step 2 : Here is the python code.

```
#Libraries
import RPi.GPIO as GPIO
import time

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive

# Rename the downloaded JSON file to client_secrets.json which downloaded from
the googleapi
# The client_secrets.json file needs to be in the same directory as the
script.
gauth = GoogleAuth()
```

```
gauth.LocalWebserverAuth()
drive = GoogleDrive(gauth)

#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)
#set GPIO Pins
GPIO_TRIGGER = 23
GPIO_ECHO = 24
#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
#string name
def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)
    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    StartTime = time.time()
    StopTime = time.time()
    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()
        # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()
    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
    return distance


if __name__ == '__main__':
    while True:
        try:
            while True:
                dist = distance()
                print ("Measured Distance = %.1f cm" % dist)

                if(dist<10):
                    print("you are close to Object")


                    time.sleep(1)
                    # Reset by pressing CTRL + C
                dis =  str(dist) #{0:4d}
```

```
                 #{1:3}%".format (adcout, percent)
                reporttime = (time.strftime("%H:%M:%S"))
                csvresult = open("paste the path of the folder where code was 
saved/Ultrosonic_data22.csv","a")
                #csvresult.write(dis + ","+ closer +" , " + reporttime + "\n")
                csvresult.write(dis + ","+ reporttime + "\n")
                csvresult.close
                # # Upload files to your Google Drive
                upload_file_list = ["Ultrosonic_data22.csv"]
                for upload_file in upload_file_list:
                    gfile = drive.CreateFile({'parents': [{'id': 'google 
folder id where you want to store'}]})
                    # Read file and set it as a content of this instance.
                    gfile.SetContentFile(upload_file)
                    gfile.Upload() # Upload the file.
        except KeyboardInterrupt:
            print("Measurement stopped by User")
            GPIO.cleanup()
```

Note: while running the code, you must the save the **settings.yml file , client_secrets.json (**renaming the downloaded json file**)** in the same folder.

And you must change the code as per your sensor connections(Here I am using ultrasonic sensor with Raspbeery_pi board.)

google folder id where you want to store : To find the id open the folder in the google drive where you want to store after opening you will find the path of the folder in the google search bar as

https://drive.google.com/drive/folders/1T_dwqjhKSazj_3m-yIthSeZgmbjbG2KQ

The black part of the above url is your folder id

By : Nagarjuna Bantu.

Reference : https://www.projectpro.io/recipes/upload-files-to-google-drive-using-python