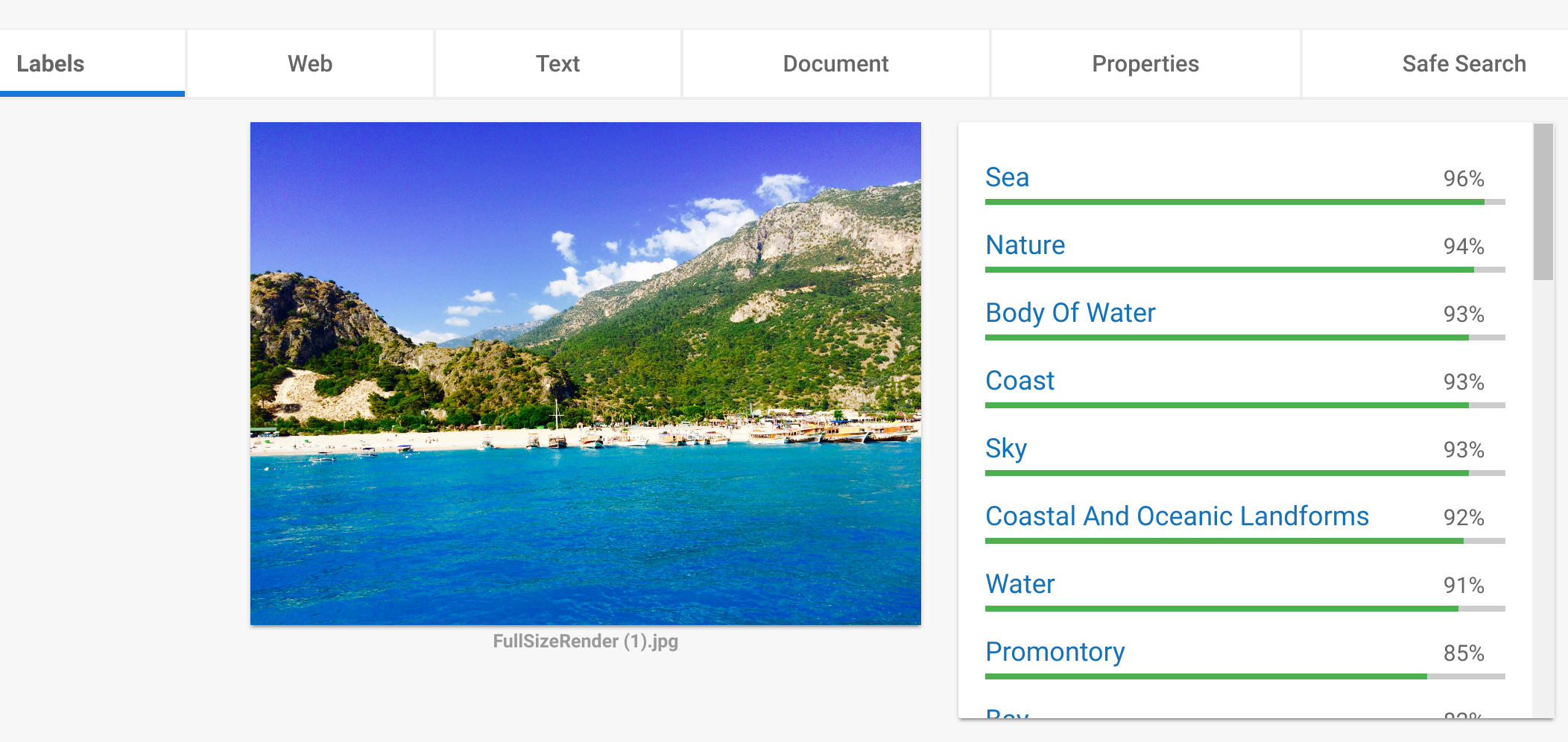
Vision API Python App - Student Guide

# Overview

APIs are produced to allow us to use functions produced by other developers. This means that we don’t have to produce a lot of basic code from scratch, and that our code can do much more impressive things.

Google have encapsulated their Machine Learning models in an API to allow developers to use their Vision technology. The Vision API can quickly classify images into thousands of categories and assign them sensible labels. It can even detect individual objects, faces, and pieces of text within an image.



The Vision API can give developers unique insights into sets of images, and as it is based on Machine Learning, the accuracy of the API improves over time. With the right development tools, we can create simple Python scripts that interface with the Vision API. Unlike a Web Application, all aspects of the program will be hosted locally, however there will be an exchange of information when the image is sent to the Vision API and the data is returned.

# Initial Setup

[This page](https://cloud.google.com/vision/docs/reference/libraries#client-libraries-install-python) will guide you through the setup process. Using the pip Python package installer, you can install the Vision client library with:

pip install --upgrade google-cloud-vision

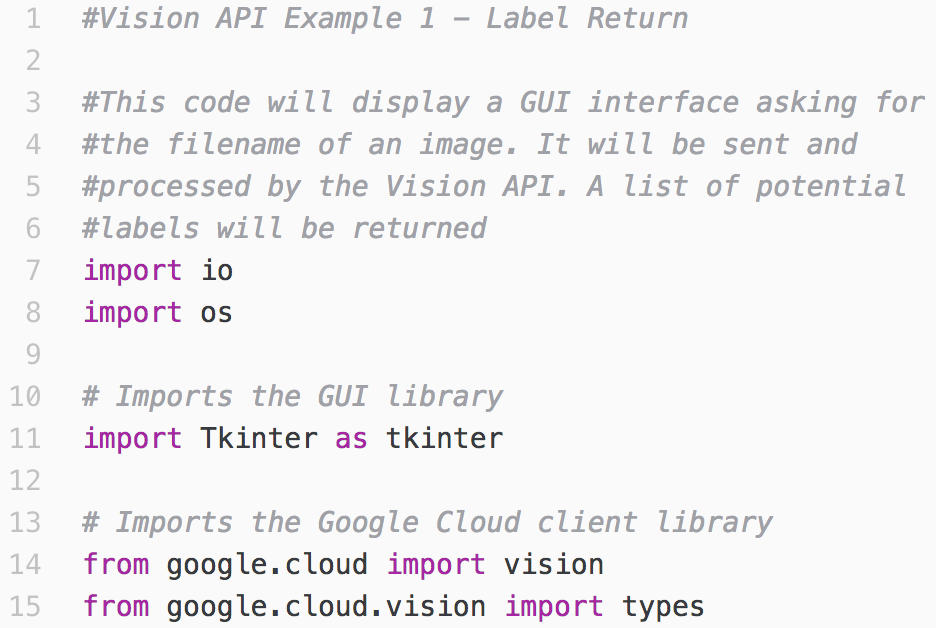
Next, follow the steps [here](https://cloud.google.com/sdk/docs/) to install the Google Cloud SDK to your computer.

Finally, run this command to authenticate your computer and the SDK.

gcloud auth application-default login

The following two sections will cover how the code interfaces with the Vision API and produces output to the GUI. The full code listings are available in the [Appendices](#_omcshxgt1p3o).

# Script 1 - Label Return

****

These lines import key libraries that will be used in the script. If your code produces an error for line 14 or 15, it is likely that the Vision client library or Google Cloud SDK was not installed properly.

****

The ‘Tkinter’ GUI library will be used to produce the UI for our app. We will require use of the button widget to activate the API and generate output for the user. This widget requires the name of a subroutine to be called when it is clicked. **btnclick()** serves that purpose.

It is good practice to produce feedback to the user if something has not been input correctly. If they call upon the API but have not indicated the filepath of an image, they will get an error. The button’s text is updated to reflect this on line 20. The if/else branching controls whether or not the code runs based on the presence of that particular error.

The following code initiates the Vision client, identifies the picture, sends it to the API and stores the returned data. On line 39, the label data is stripped from the API response and stored in a list. On line 42, a series of GUI labels are created to display the API response labels. These are displayed in the window. The button’s text updates to indicate that the process is complete.



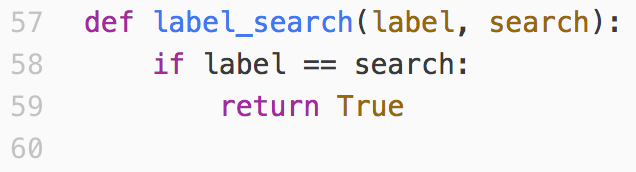
At the end of the script, a series of references are made to the Tkinter library. These lines initiate the GUI window and pack it with widgets that remain static.



# Script 2 - Label Search



This code follows on from the same library imports that were seen in the first script. Similarly to the previous example - an image is identified based on its filename and the Vision API’s response is formatted appropriately. The main difference is that instead of displaying each label as a visual widget, the label is passed to a function with the user’s search term. A boolean (True or False) flag is returned which will indicate whether or not there was a match. The function is seen below.





This code extends the GUI initialization from the previous script by also initializing the GUI elements for the search box and associated labels.

# 

# 

# Appendix A: Example 1: Label Return

#Vision API Example 1 - Label Return  
  
#This code will display a GUI interface asking for  
#the filename of an image. It will be sent and  
#processed by the Vision API. A list of potential  
#labels will be returned  
import io  
import os  
  
# Imports the GUI library  
import Tkinter as tkinter  
  
# Imports the Google Cloud client library  
from google.cloud import vision  
from google.cloud.vision import types  
  
#This subroutine defines what happens when the button is pressed  
def btnclick():  
 if ent.get() == "":  
 btn.configure(text="No Filename")  
 else:  
 btn.configure(text="Uploading")  
 # Instantiates a client  
 client = vision.ImageAnnotatorClient()  
 filename = ent.get()  
  
 # The name of the image file to annotate  
 file\_name = os.path.join(  
 os.path.dirname(\_\_file\_\_), filename)  
  
 # Loads the image into memory  
 with io.open(file\_name, 'rb') as image\_file:  
 content = image\_file.read()  
  
 image = types.Image(content=content)  
  
 # Performs label detection on the image file  
 response = client.label\_detection(image=image)  
 labels = response.label\_annotations  
  
 #Match each identified label with the one being searched for  
 for label in labels:  
 l = tkinter.Label(window, text=label.description)  
 l.pack()  
  
 btn.configure(text="Done")  
 btn.configure(text="Upload Image")  
  
# Instantiate a new GUI Window  
window = tkinter.Tk()  
window.title("Google Cloud Vision API")  
window.geometry("350x300")  
window.configure(background = "#ffffff")  
  
#Defines GUI Elements  
lbl = tkinter.Label(window, text="Google Cloud Vision API", fg="#383a39", bg="#ffffff", font=("Helvetica", 23))  
lbl3 = tkinter.Label(window, text="")  
lbl2 = tkinter.Label(window, text="Enter an image's filename and click 'Upload Image'")  
ent = tkinter.Entry(window)  
btn = tkinter.Button(window, text="Upload Image", command = btnclick)  
  
#Packs GUI Elements into window  
lbl.pack()  
lbl3.pack()  
lbl2.pack()  
ent.pack()  
btn.pack()  
  
window.mainloop()

# Appendix B: Example 2: Label Search

#Vision Example 2 - Label Search

#This code allows the user to enter a search term

# and compares it against the labels identified

#by the Vision API. If there is a match, it is

#displayed in the GUI.

import io

import os

# Imports the GUI library

import Tkinter as tkinter

# Imports the Google Cloud client library

from google.cloud import vision

from google.cloud.vision import types

# This subroutine defines what will happen when the button is clicked

def btnclick():

if ent.get() == "":

btn.configure(text="No Filename")

else:

lbl6.configure(text="Uploading")

# Initiates a client

client = vision.ImageAnnotatorClient()

filename = ent.get()

# The name of the image file to annotate

file\_name = os.path.join(

os.path.dirname(\_\_file\_\_), filename)

# Loads the image into memory

with io.open(file\_name, 'rb') as image\_file:

content = image\_file.read()

image = types.Image(content=content)

# Performs label detection on the image file

response = client.label\_detection(image=image)

labels = response.label\_annotations

if searchent.get() != 0:

lbl6.configure(text="Not found")

#Match each identified label with the one being searched for

for label in labels:

#If there is a match, lbl6 is updated

if label\_search(label.description, searchent.get()) == True:

lbl6.configure(text = "Label found")

btn.configure(text="Done")

btn.configure(text="Upload Image")

#Compares label with label to be searched for

#Returns True if there is a match

def label\_search(label, search):

if label == search:

return True

# Instantiate a new GUI Window

window = tkinter.Tk()

window.title("Google Cloud Vision API")

window.geometry("350x250")

window.configure(background = "#ffffff")

#Defines GUI Elements

lbl = tkinter.Label(window, text="Google Cloud Vision API", fg="#383a39", bg="#ffffff", font=("Helvetica", 23))

lbl3 = tkinter.Label(window, text="")

lbl2 = tkinter.Label(window, text="Enter an image's filename and click 'Upload Image'")

ent = tkinter.Entry(window)

btn = tkinter.Button(window, text="Upload Image", command = btnclick)

# GUI for search term

lbl4 = tkinter.Label(window, text="")

lbl5 = tkinter.Label(window, text = "Search Term")

searchent = tkinter.Entry(window)

lbl6 = tkinter.Label(window, text = "")

#Packs GUI Elements into window

lbl.pack()

lbl3.pack()

lbl2.pack()

ent.pack()

btn.pack()

#Search graphics

lbl4.pack()

lbl5.pack()

searchent.pack()

lbl6.pack()

window.mainloop()