

## Google Cloud Skills Boost for Partners

[Main menu](#)

## Build Custom Processors with Document AI

Course · 6 hours 16%  
15 minutes complete

## Course overview

## Build Custom Processors with Document AI

## Optical Character Recognition (OCR) with Document AI (Python)

## Form Parsing with Document AI (Python)

## Using Specialized Processors with Document AI (Python)

## Uptraining with Document AI Workbench

## Custom Document Extraction with

Course &gt; Build Custom Processors with Document AI &gt;

Quick tip: Review the prerequisites before you run the lab

[End Lab](#)

00:28:59

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked.  
[Learn more.](#)[Open Google Cloud console](#)

Username

student-00-e1d0743047f2e

Password

rEZzcmHoLfvr

Project ID

qwiklabs-gcp-04-4b89663:

# Form Parsing with Document AI (Python)

Lab 45 minutes No cost Intermediate

★★★★★

This lab may incorporate AI tools to support your learning.

Lab instructions and tasks

100/100

GSP1139

Overview

Objectives

Setup and requirements

Task 1. Enable the Document AI API

Task 2. Create a Form Parser processor

Task 3. Download the sample form

Task 4. Extract form key/value pairs

Task 5. Parse tables

Congratulations!

GSP1139

 Google Cloud Self-Paced Labs
[Previous](#)[Next >](#)

## Overview

[Document AI](#) is a document understanding solution that takes unstructured data (e.g. documents, emails, invoices, forms, etc.) and makes the data easier to understand, analyze, and consume. The API provides structure through content classification, entity extraction, advanced searching, and more.

In this lab, you will learn how to use the Document AI Form Parser to parse a handwritten form with Python. You will use a simple medical intake form as an example, but this procedure will work with any generalized form supported by DocAI.

## Objectives

In this lab, you will learn how to perform the following tasks:

- Extract data from a scanned form using the Document AI Form Parser
- Extract key/value pairs from a form using the Document AI Form Parser
- Extract and export CSV data from a form using the Document AI Form Parser

## Setup and requirements

### Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources are made available to you.

Simulation or demo environment. It does so by giving you new, temporary credentials you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

**Note:** Use an Incognito (recommended) or private browser window to run this lab. This prevents conflicts between your personal account and the student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab—remember, once you start, you cannot pause a lab.

**Note:** Use only the student account for this lab. If you use a different Google Cloud account, you may incur charges to that account.

## How to start your lab and sign in to the Google Cloud console

1. Click the **Start Lab** button. If you need to pay for the lab, a dialog opens for you to select your payment method. On the left is the Lab Details pane with the following:

- The Open Google Cloud console button
- Time remaining
- The temporary credentials that you must use for this lab
- Other information, if needed, to step through this lab

2. Click **Open Google Cloud console** (or right-click and select **Open Link in Incognito Window** if you are running the Chrome browser).

The lab spins up resources, and then opens another tab that shows the Sign in page.

*Tip:* Arrange the tabs in separate windows, side-by-side.

**Note:** If you see the **Choose an account** dialog, click **Use Another Account**.

3. If necessary, copy the **username** below and paste it into the **Sign in** dialog.

You can also find the Username in the Lab Details pane.

4. Click **Next**.

5. Copy the **Password** below and paste it into the **Welcome** dialog.

You can also find the Password in the Lab Details pane.

6. Click **Next**.

**Important:** You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.

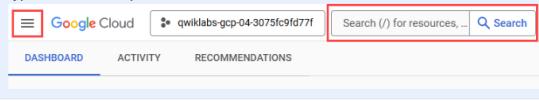
**Note:** Using your own Google Cloud account for this lab may incur extra charges.

7. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Google Cloud console opens in this tab.

**Note:** To access Google Cloud products and services, click the **Navigation menu** or type the service or product name in the **Search** field.



## Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.

2. Click through the following windows:

- Continue through the Cloud Shell information window.
- Authorize Cloud Shell to use your credentials to make Google Cloud API calls.

When you are connected, you are already authenticated, and the project is set to your **Project\_ID**, `qwiklabs-gcp-04-4b8966377718`. The output contains a line that declares the **Project\_ID** for this session:

```
[REDACTED]
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

4. Click **Authorize**.

**Output:**

```
ACTIVE: *
ACCOUNT: student-00-e1d0743047f2@qwiklabs.net

To set the active account, run:
$ gcloud config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

**Output:**

```
[core]
project = qwiklabs-gcp-04-4b8966377718
```

**Note:** For full documentation of `gcloud`, in Google Cloud, refer to [the gcloud CLI overview guide](#).

## Task 1. Enable the Document AI API

1. Open Cloud Shell by clicking the **Activate Cloud Shell** button at the top of the console.

2. In Cloud Shell, run the following commands to enable the API for Document AI.

```
gcloud services enable documentai.googleapis.com
```

You should see something like this:

```
Operation "operations/..." finished successfully.
```

You will also need to install [Pandas](#), an Open Source Data Analysis library for Python.

3. Run the following command to install Pandas.

```
pip3 install --upgrade pandas
```

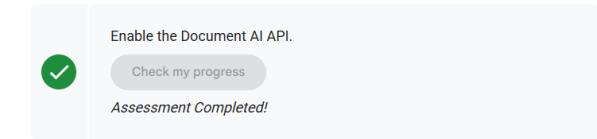
```
pip3 install --upgrade google-cloud-documentai
```

You should see something like this:

```
...
Installing collected packages: google-cloud-documentai
Successfully installed google-cloud-documentai-2.15.0
```

Now, you're ready to use the Document AI API!

Click **Check my progress** to verify the objective.



## Task 2. Create a Form Parser processor

You must first create a Form Parser processor instance to use in the Document AI Platform for this tutorial.

1. In the Cloud console, open the **Navigation menu** (≡), click **View All Products > Artificial Intelligence > Document AI**.

2. Click **Explore Processors**, and inside **Form Parser**, click **Create Processor**.

4. Click **Create** to create your processor

5. Copy your Processor ID. You must use this in your code later.

Click **Check my progress** to verify the objective.

Create a processor

Check my progress

Assessment Completed!

## Test the processor in the Cloud Console

You can test out your processor in the console by uploading a document.

1. Right click the image below, and select **Save Image As** to download the sample form.

FakeDoc M.D.

**HEALTH INTAKE FORM**

Please fill out the questionnaire carefully. The information you provide will be used to complete your health profile and will be kept confidential.

Name: Sally Walker DOB: 09/04/1980  
Address: 24 Barney Lane City: Toronto State: NJ Zip: 07082  
Email: sally.walker@gmail.com Phone #: (906) 917-3486  
Gender: F Marital Status: Single Occupation: Software Engineer  
Referred By: None  
Emergency Contact: Eva Walker Emergency Contact Phone: (906) 334-8940

Describe your medical concerns (symptoms, diagnoses, etc):  
Runny nose, mucus in throat, weakness,  
aches, chills, fever

Are you currently taking any medication? (If yes, please describe):  
Vyvanse (25mg) daily for attention

you just downloaded.

Your Form Parser processor will process the document and return the parsed form data. It should look something like this:

← Form Parser analysis NEW DOCUMENT

KEY VALUE PAIR TABLE ENTITY

Filter Type to filter

Date:	9/14/19
DOB:	09/04/1986
Name:	Sally Walker
Address:	24 Barney Lane
State:	NJ
City:	Toronto
Zip:	07082
Phone #:	(906) 917-3486
Email:	Sally.Walker@gmail.com
Marital Status:	Single
Occupation:	Software Engineer
Gender:	F

FakeDoc M.D.

**HEALTH INTAKE FORM**

Please fill out the questionnaire carefully. The information you provide will be used to complete your health profile and will be kept confidential.

Name: Sally Walker DOB: 09/04/1980  
Address: 24 Barney Lane City: Toronto State: NJ Zip: 07082  
Email: sally.walker@gmail.com Phone #: (906) 917-3486  
Gender: F Marital Status: Single Occupation: Software Engineer  
Referred By: None  
Emergency Contact: Eva Walker Emergency Contact Phone: (906) 334-8940

Describe your medical concerns (symptoms, diagnoses, etc):  
Runny nose, mucus in throat, weakness,  
aches, chills, fever

Are you currently taking any medication? (If yes, please describe):  
Vyvanse (25mg) daily for attention

## Task 3. Download the sample form

In this section, you will download a sample document which contains a simple medical intake form.

1. Run the following command to download the sample form to your Cloud Shell.

```
gcloud storage cp gs://cloud-samples-data/documentai/code-labs/form-parser/intake-form.pdf .
```

2. Confirm the file is downloaded to your Cloud Shell using the below command:

## Task 4. Extract form key/value pairs

In this section, you will use the online processing API to call the Form Parser processor you created previously. Then, you will extract the key value pairs found in the document.

Online processing is for sending a single document and waiting for the response. You can also use batch processing if you want to send multiple files or if the file size exceeds the [online processing maximum pages](#).

The code for making a process request is identical for every processor type aside from the Processor ID. The `Document` response object contains a list of pages from the input document. Each `page` object contains a list of form fields and their locations in the text.

The following code iterates through each page and extracts each key, value and

1. In Cloud Shell, create a file called `form_parser.py` and paste the following code into it:

```
import pandas as pd
from google.cloud import documentai_v1 as documentai

def online_process(
    project_id: str,
    location: str,
    processor_id: str,
    file_path: str,
    mime_type: str,
) -> documentai.Document:
    """
    Processes a document using the Document AI Online Processing
    API.
    """
    opts = {"api_endpoint": f"{location}-"

        # Instantiates a client
        documentai_client =
        documentai.DocumentProcessorServiceClient(client_options=opts)

        # The full resource name of the processor, e.g.:
        # projects/project-id/locations/location/processor/processor-
        id
        # You must create new processors in the Cloud Console first
        resource_name = documentai_client.processor_path(project_id,
        location, processor_id)

        # Read the file into memory
        with open(file_path, "rb") as image:
            image_content = image.read()

        # Load Binary Data into Document AI RawDocument Object
        raw_document = documentai.RawDocument(
            content=image_content, mime_type=mime_type
        )

        # Configure the process request
        request = documentai.ProcessRequest(


            # Use the Document AI client to process the sample form
            result =
            documentai_client.process_document(request=request)

            return result.document

    def trim_text(text: str):
        """
        Remove extra space characters from text (blank, newline, tab,
        etc.)
        """
        return text.strip().replace("\n", " ")

PROJECT_ID = "YOUR_PROJECT_ID"
LOCATION = "YOUR_PROJECT_LOCATION" # Format is 'us' or 'eu'
PROCESSOR_ID = "FORM_PARSER_ID" # Create processor in Cloud
```

```

Console

# The local file in your current working directory

list
# for supported file types
MIME_TYPE = "application/pdf"

document = online_process(
    project_id=PROJECT_ID,
    location=LOCATION,
    processor_id=PROCESSOR_ID,
    file_path=FILE_PATH,
    mime_type=MIME_TYPE,
)

names = []
name_confidence = []
values = []
value_confidence = []

for page in document.pages:
    for field in page.form_fields:
        # Get the extracted field names

names.append(trim_text(field.field_name.text_anchor.content))

name_confidence.append(field.field_name.confidence)

values.append(trim_text(field.field_value.text_anchor.content))
value_confidence.append(field.field_value.confidence)

# Create a Pandas Dataframe to print the values in tabular
format.
df = pd.DataFrame(
{
    "Field Name": names,
    "Field Name Confidence": name_confidence,
    "Field Value": values,
    "Field Value Confidence": value_confidence,
}
)
print(df)

```

2. Replace YOUR\_PROJECT\_ID, YOUR\_PROJECT\_LOCATION, YOUR\_PROCESSOR\_ID, and the FILE\_PATH with appropriate values for your environment.

the previous step. If you didn't rename the file, it should be `intake-form.pdf`, which you will need to update in the code.

3. Run the following command to execute the script:

```
python3 form_parser.py
```

You should see the following output:

	Field Name	Field Name Confidence
0	Phone #:	0.9999
1	Emergency Contact:	0.9999
2	Marital Status:	0.9999
3	Gender:	0.9999
4	Occupation:	0.9999
5	Referred By:	0.9998
6	Date:	0.9998
7	DOB:	0.9997
8	Address:	0.9991
11	State:	0.9969
...		

## Task 5. Parse tables

The Form Parser is also able to extract data from tables within documents. In this section, you will download a new sample document and extract data from the table. Since you are loading the data into Pandas, this data can be output to a CSV file and

many other formats with a single method call.

## Download the Sample Form with Tables

- Run the following command to download the sample form to your Cloud Shell.

```
gcloud storage cp gs://cloud-samples-data/documentai/codelabs/form-parser/form_with_tables.pdf .
```

- Confirm the file is downloaded to your Cloud Shell using the below command:

```
ls form_with_tables.pdf
```

## Extract Table Data

The processing request for table data is exactly the same as for extracting key-value pairs. The difference is which fields you extract the data from in the response. Table data is stored in the `pages[ ].tables[ ]` field.

This example extracts information about from the table header rows and body rows for

- Create a file called `table_parsing.py` and paste the following code into it:

```
# type: ignore[1]
"""
Uses Document AI online processing to call a form parser
processor
Extracts the tables and data in the document.
"""

from os.path import splitext
from typing import List, Sequence

import pandas as pd
from google.cloud import documentai


def online_process(
    project_id: str,
    location: str,
    processor_id: str,
    file_path: str,
    mime_type: str,
    content_type="application/pdf"
):
    """
    Processes a document using the Document AI Online Processing
    API.
    """

    opts = {"api_endpoint": f"{location}-documentai.googleapis.com"}

    # Instantiates a client
    documentai_client = documentai.DocumentProcessorServiceClient(client_options=opts)

    # The full resource name of the processor, e.g.:
    # projects/project-id/locations/location/processor/processor-
    id
    # You must create new processors in the Cloud Console first
    resource_name = documentai_client.processor_path(project_id,
    location, processor_id)

    # Read the file into memory
    with open(file_path, "rb") as image:
        image_content = image.read()

    # Create a raw document object
    raw_document = documentai.RawDocument(
        content=image_content, mime_type=mime_type
    )

    # Configure the process request
    request = documentai.ProcessRequest(
        name=resource_name, raw_document=raw_document
    )

    # Use the Document AI client to process the sample form
    result = documentai_client.process_document(request=request)

    return result.document
```

```

def get_table_data(
    rows: Sequence[documentai.Document.Page.Table TableRow],
    text: str
) -> List[List[str]]:
    """
    Get Text data from table rows
    """
    all_values: List[List[str]] = []
    current_row_values: List[str] = []
    last_index = -1
    for cell in row.cells:
        current_row_values.append(
            text_anchor_to_text(cell.layout.text_anchor,
            text)
        )
    all_values.append(current_row_values)
    return all_values

def text_anchor_to_text(text_anchor):
    documentai.Document.TextAnchor, text: str) -> str:
    """
    Document AI identifies table data by their offsets in the
    entirety of the
    document's text. This function converts offsets to a string.
    """
    response = ""
    # If a text segment spans several lines, it will
    # be stored in different text segments.
    for segment in text_anchor.text_segments:
        start_index = int(segment.start_index)
        end_index = int(segment.end_index)

```

```

PROJECT_ID = "YOUR_PROJECT_ID"
LOCATION = "YOUR_PROJECT_LOCATION" # Format is 'us' or 'eu'
PROCESSOR_ID = "FORM_PARSER_ID" # Create processor before
running sample

# The local file in your current working directory
FILE_PATH = "form_with_tables.pdf"
# Refer to https://cloud.google.com/document-ai/docs/file-types
# for supported file types
MIME_TYPE = "application/pdf"

document = online_process(
    project_id=PROJECT_ID,
    location=LOCATION,
    processor_id=PROCESSOR_ID,
    file_path=FILE_PATH,
    mime_type=MIME_TYPE,
)
header_row_values: List[List[str]] = []

# Input Filename without extension
output_file_prefix = splitext(FILE_PATH)[0]

for page in document.pages:
    for index, table in enumerate(page.tables):
        header_row_values = get_table_data(table.header_rows,
        document.text)
        body_row_values = get_table_data(table.body_rows,
        document.text)

        # Create a Pandas Dataframe to print the values in
        # tabular format.
        df = pd.DataFrame(
            data=body_row_values,
            columns=pd.MultiIndex.from_arrays(header_row_values),
        )

        print(f"Page {page.page_number} - Table {index}")
        print(df)

        # Save each table as a CSV file
        output_filename = f"
```

2. Replace YOUR\_PROJECT\_ID, YOUR\_PROJECT\_LOCATION, YOUR\_PROCESSOR\_ID, and the FILE\_PATH with appropriate values for your environment.

**Note** that the FILE\_PATH is the name of the file you downloaded to Cloud Shell in the previous step. If you didn't rename the file, it should be form\_with\_tables.pdf, which is the default value and doesn't need to be changed.

3. Run the following command to execute the script:

```
python3 table_parsing.py
```

You should see the following output:

```
Page 1 - Table 0
Item    Description
1       Item 1
2       Item 2
3       Item 3
```

You should also have a new CSV file in the directory you are running the code from.

4. Run the following command to list the files in your current working directory:

```
ls
```

You should see the following output:

```
form_with_tables_pg1_tb0.csv
```

## Congratulations!

Congratulations, in this lab you've successfully used the Document AI API to extract data from a handwritten form. You also learned how to use the Document AI Python client library to extract key-value pairs from a form and how to extract tabular data from a form with tables.

## Next steps / Learn more

Check out the following resources to learn more about Document AI and the Python Client Library:

- [Document AI Documentation](#)
- [Document AI Python Client Library](#)
- [Document AI Samples](#)

## Google Cloud training and certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

**Manual Last Updated November 13, 2024**

**Lab Last Tested November 13, 2024**

Copyright 2025 Google LLC. All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.