

# [ML on GCP C9] Getting Started with Dialogflow

2 hours

Free

[Rate Lab](#)

## Overview

This lab shows you how to build a simple Dialogflow agent, walking you through the most important features of Dialogflow. You'll learn how to:

- [Create a Dialogflow account](#) and [your first Dialogflow agent](#), which lets you define a natural language understanding model.

## Setup

For each lab, you get a new GCP project and set of resources for a fixed time at no cost.

1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, **02:00:00** and make sure you can finish in that time block.


There is no pause feature. You can restart if needed, but you have to start at the beginning.


3. When ready, click  .


4. Note your lab credentials. You will use them to sign in to Cloud Platform Console.

Open Google Console

**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.](#)

**Username**  
google2876526\_student@qwiklabs.n 

**Password**  
TG959yrKDX 

**GCP Project ID**  
qwiklabs-gcp-0855e773352d3560 

[New to labs? View our introductory video!](#)

5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

If you use other credentials, you'll get errors or **incur charges**.

7. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you are finished with the lab or want to restart it.  
This clears your work and removes the project.

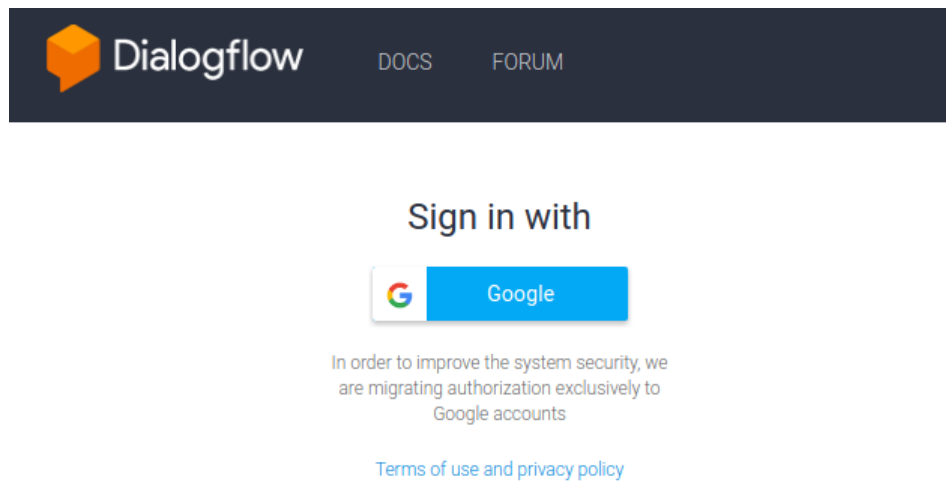
## Create your Dialogflow account

This section describes how to create and log in to a Dialogflow account

### Create your Dialogflow account

Now that you're signed into your Qwiklabs student account in an incognito (private browser) window, you can sign into Dialogflow [here](#) by following these steps:

1. Click **Google**.



2. Be sure to sign in with your Qwiklabs account
3. Allow Dialogflow to access your Google account. [See a list of the permissions and what they're used for.](#)

Lastly, you'll be taken to Dialogflow's terms of service, which you'll need to accept in order to use Dialogflow.

## Please review your account settings

### Country or territory \*

United States

### Email preferences

Stay up-to-date with occasional emails from our team

- ☒ **News and tips**  
Learn about new features, enhancements and tips
- ☒ **Feedback and testing**  
Participate in surveys and pilots to improve Dialogflow

### Terms of Service \*

- ☒ **Yes, I have read and accept the agreement.**

By proceeding and clicking the button below, you agree to adhere to the [Terms of Service](#).

Additionally, you may have access to certain Firebase services. You agree that your use of Firebase services will adhere to the applicable [Firebase Terms of Service](#). If you integrate any apps with Firebase on this project, by default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your Firebase settings at anytime.

ACCEPT

## Next steps

Next, you'll create your first Dialogflow agent and test it out.

## Create and query your first agent

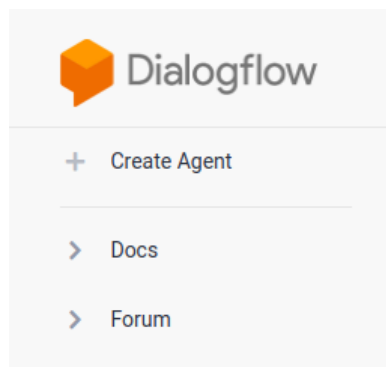
This section describes how to create and try out your first Dialogflow agent.

**Note:** Before you start, make sure you've [created a Dialogflow account](#).

### Create your first Dialogflow agent

To create a Dialogflow agent:

1. Open a browser and [log in to Dialogflow](#).
2. Click **Create agent** in the left menu.



3. Enter your agent's name, default language, and default time zone, then click the **Create** button.

## NewAgent

CREATE

### DEFAULT LANGUAGE

English — en

Primary language for your agent. Other languages can be added later.

### DEFAULT TIME ZONE

(GMT-8:00) America/Los\_Angeles

Date and time requests are resolved using this timezone.

### GOOGLE PROJECT

Create a new Google project

Enables Cloud functions, Actions on Google and permissions management.

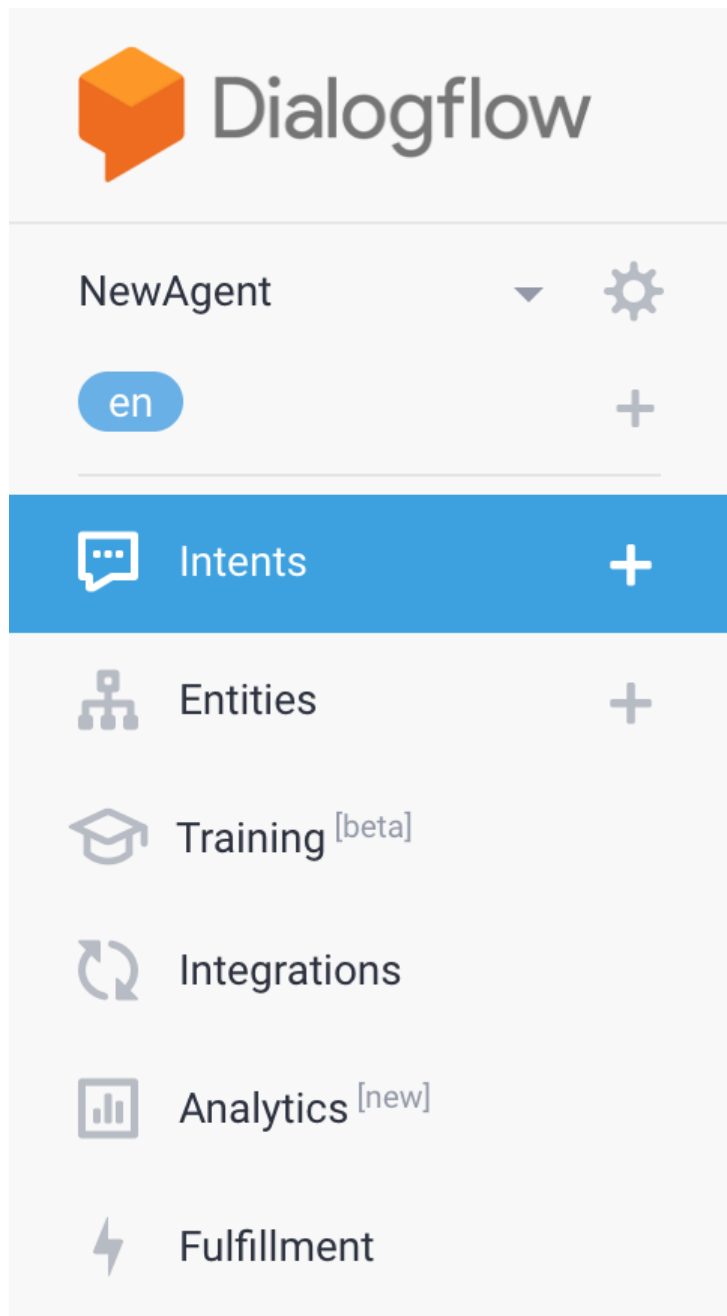
### API VERSION



**Dialogflow V2 API** <sup>[beta]</sup>


Use [Dialogflow V2 API](#) as default for the agent. Your webhook will receive [V2 format requests](#) and should return [V2 format responses](#).


## The Dialogflow console





You should now see the Dialogflow console and the menu panel on the left. If you're working on a smaller screen and the menu is hidden, click on the menu button in the upper left corner. The settings button takes you to the current [agent's settings](#).

The middle of the page will show the list of intents for the agent. By default, Dialogflow agents start with two intents. Your agent matches the **Default Fallback Intent** when it doesn't understand what your users say. The **Default Welcome Intent** greets your users. These can be altered to customize the experience.


 Intents CREATE INTENT




 Default Fallback Intent

 Default Welcome Intent

Try it now



 Please use test console above to try a sentence.

 See how it works in [Google Assistant](#). 

On the right is the Dialogflow simulator. This lets you try out your agent by speaking or typing messages.

## Query your agent

Agents are best described as NLU (Natural Language Understanding) modules. These can be included in your app, product, or service and transform natural user requests into actionable data.

hi



See how it works in [Google Assistant](#).

## Agent

USER SAYS

[COPY CURL](#)

hi



DEFAULT RESPONSE



Greetings! How can I assist?

INTENT

[Default Welcome Intent](#)

ACTION

input.welcome

## DIAGNOSTIC INFO

Time to try out your agent! In the **Dialogflow simulator** on the right, click into the text field that says **Try it now**, type **hi**, and press enter.

You just spoke to your Dialogflow agent! You may notice your agent understood you. Since your input matched to the Default Welcome Intent, you received one of the default replies inside the welcome intent.



In the case that your agent doesn't understand you, the Default Fallback Intent is matched and you receive one of the default replies inside that intent.

The Default Fallback Intent reply prompts the user to reframe their query in terms that can be matched. You can change the responses within the Default Fallback Intent to provide example queries and guide the user to make requests that can match an intent.

## Create your first intent

Dialogflow uses intents to categorize a user's intentions. Intents have **Training Phrases**, which are examples of what a user might say to your agent. For example, someone wanting to know the name of your agent might ask, "What is your name?", "Do you have a name?", or just say "name". All of these queries are unique but have the same intention: to get the name of your agent.

To cover this query, create a "name" intent:

1. Click on the plus add next to **Intents** in the left menu.
2. Add the name "name" into the **Intent name** text field.
3. In the **Training Phrases** section, click **Add Training Phrases** enter the following, pressing enter after each entry:
  - What is your name?
  - Do you have a name?
  - name
4. In the **Responses** section, click **Add Response** enter the following response:
  - My name is Dialogflow!

Name

SAVE

Training phrases ?

Search in user says

” name

” Do you have a name?

” What is your name?

Action & parameters ?

Responses ?

DEFAULT

GOOGLE ASSISTANT

+

Text response

1 My name is Dialogflow!

2 Enter a text response variant

5. Click the **Save** button.

**Try it out!**

What's your name?



See how it works in [Google Assistant](#). [↗](#)

Agent

Domains

USER SAYS

[COPY CURL](#)

What's your name?



DEFAULT RESPONSE



[PLAY](#)

My name is Dialogflow!

INTENT

[Name](#)

ACTION

*Not available*

SHOW JSON

Now try asking your agent for its name. In the simulator on the right, type "What's your name?" and press enter.

Your agent now responds to the query correctly. Notice that even though your query was a little different from the training phrase ("What's your name?" versus "What is your name?"), Dialogflow still matched the query to the right intent.

Dialogflow uses training phrases as examples for a machine learning model to match users' queries to the correct intent. The [machine learning](#) model checks the query against every intent in the agent, gives every intent a score, and the highest-scoring intent is matched. If the highest scoring intent has a very low score, the fallback intent is matched.

## Extract data with entities

This section describes how to extract data from a user's query.

### Add parameters to your intents

Parameters are important and relevant words or phrases in a user's query that are extracted so your agent can provide a proper response. You'll create a new intent with parameters for spoken and programming languages to explore how these can match specific intents and be included in your responses.

1. Create a new intent by clicking on the plus add next to **Intents** in the left menu.
2. Name the intent "Languages" at the top of the intent page.
3. Add the following as Training phrases:

- I know English
- I speak French
- I know how to write in German

#### • Languages

SAVE

##### Training phrases ?

Search in user says 🔍 ^

” Add user expression

” I know English

” I speak French

” I know how to write in German

##### Action & parameters ?

Enter action name

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?
<input type="checkbox"/>	language	@sys.language	\$language	<input type="checkbox"/>

Dialogflow automatically detects known parameters in your Training phrases and creates them for you.

Below the **Training phrases** section, Dialogflow fills out the parameter table with the information it gathered:

- The parameter is optional (not required)
- named language
- corresponds to the [system entity](#) type [@sys.language](#)
- has the value of \$language
- is not a [list](#)

**Note:** If entities aren't automatically detected, you can highlight the text in the Training phrase and [manually annotate the entity](#).

## Use parameter data

The screenshot shows the 'Responses' section in Dialogflow. At the top, there's a 'Responses' header with a help icon and an upward arrow. Below it, a 'DEFAULT' tab is selected, followed by a plus sign to add more. A table titled 'Text response' contains two rows: row 1 with the text 'Wow! I didn't know you knew \$language' and row 2 with the text 'Enter a text response variant'. To the right of the table is an 'ADD RESPONSES' button. Below the button is a toggle switch labeled 'Set this intent as end of conversation', which is currently turned off.

	Text response	
1	Wow! I didn't know you knew \$language	
2	Enter a text response variant	

ADD RESPONSES

☐ Set this intent as end of conversation

The value of a parameter can be used in your responses. In this case, you can use \$language in your responses and it will be replaced with the language specified in the query to your agent.

4. In the **Responses** section, add the following response and click the **Save** button:

- Wow! I didn't know you knew \$language

**Try it out!**

I know Russian



See how it works in [Google Assistant](#). [↗](#)

Agent

Domains

USER SAYS

[COPY CURL](#)

I know Russian



DEFAULT RESPONSE



[PLAY](#)

Wow! I didn't know you knew Russian

INTENT

[Languages](#)

ACTION

*Not available*

PARAMETER

VALUE

language

Russian

Now, query your agent with "I know Russian" in the simulator in the right panel.

You can see in the bottom of the simulator output that Dialogflow correctly extracted the language parameter with the value "Russian" from the query. In the response, you can see "Russian" was correctly inserted where the parameter value was used.

## Create your own entities

You can also create your own entities, which function similarly to Dialogflow's system entities.

To create an entity:

1. Click on the plus add next to **Entities** in the left menu.
2. Enter "ProgrammingLanguage" for the name of the entity.
3. Click on the text field and add the following entries:
  - JavaScript
  - Java
  - Python
4. When you enter an entry, pressing tab moves your cursor into the synonym field.  
Add the following synonyms for each entry:

## ProgrammingLanguage



Define synonyms ?



Allow automated expansion

JavaScript	JavaScript, js
Java	Java
Python	Python, py

Click the **Save** button.

Each entity type has to have the following:

- a name to define the category (ProgrammingLanguage)
- one or more entries (JavaScript)
- one or more synonyms (js, JavaScript)

Dialogflow can handle simple things like plurality and capitalization, but make sure to add all possible synonyms for your entries. The more you add, the better your agent can determine your entities.

## Add your new entities

Now that we've defined our entity for programming languages, add Training Phrases to the "Languages" intent:

1. Click on **Intents** in the left menu, and then click on the "Languages" intent.
2. Add the following as Training phrases:

” I know how to code in Java

” I know javascript

- I know javascript
- I know how to code in Java

3. You should see the programming languages automatically annotated in the Training phrases you entered. This adds the ProgrammingLanguage parameter to the table, which is below the **Training phrases** section.

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?
<input type="checkbox"/>	language	@sys.language	\$language	<input type="checkbox"/>
<input type="checkbox"/>	ProgrammingLanguage	@ProgrammingLanguage	\$ProgrammingLanguage	<input type="checkbox"/>

## Text response

- 1 Wow! I didn't know you knew \$language
- 2 \$ProgrammingLanguage is cool.

4. In the **Responses** section, add "\$ProgrammingLanguage is cool" and then click the **Save** button.

**Try it out!**



I know how to code in py



See how it works in [Google Assistant](#).

Agent

Domains

USER SAYS

[COPY CURL](#)

I know how to code in py



DEFAULT RESPONSE



[PLAY](#)

Python is cool.

INTENT

[Languages](#)

ACTION

*Not available*

PARAMETER

VALUE

language

ProgrammingLanguag... Python

# Manage state with contexts

This section describes how to track conversational states with follow-up intents and contexts.

## Add contexts to conversational state

1. Click on **Intents** in the left menu, and then click on the "Languages" intent.
2. Extend one of the original Text response in the **Response** section to the following:
  - Wow! I didn't know you knew \$language. How long have you known \$language?

• Languages

SAVE

Training phrases ?

Search in user s

” I know how to write in German

” I speak French

” I know English

Action & parameters ?

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?
<input type="checkbox"/>	language	@sys.language	\$language	<input type="checkbox"/>

Responses ?

DEFAULT

GOOGLE ASSISTANT

+

Text response

?

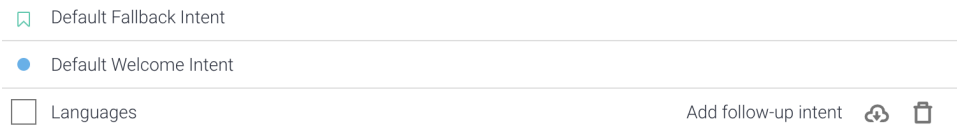
1

Wow! I didn't know you knew \$language. How long have you known \$language?

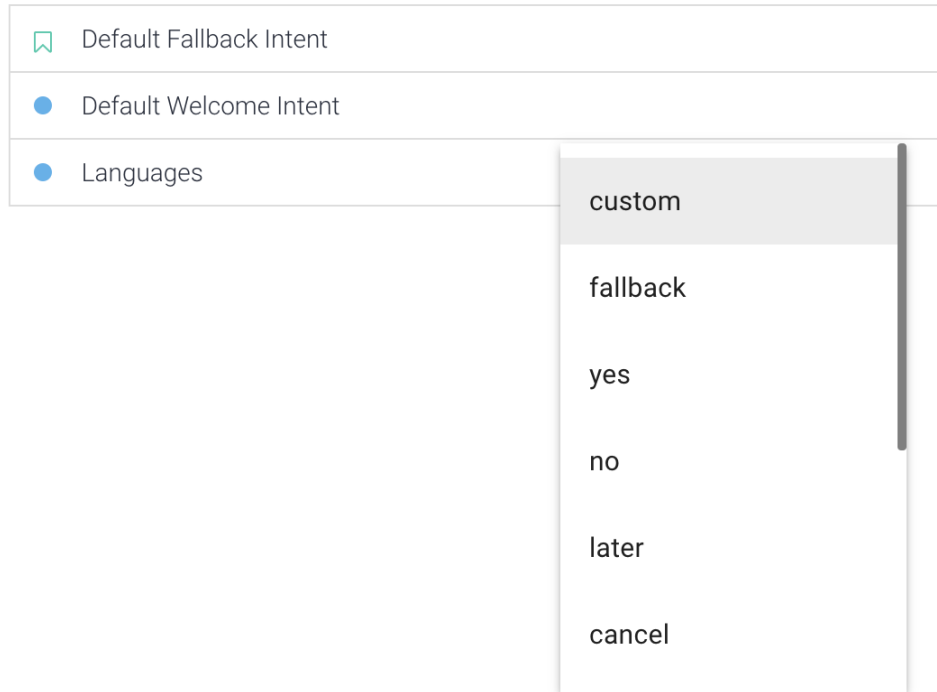
2

Enter a text response variant

3. Click the **Save** button.
4. Click on **Intents** in the left menu.
5. Hover over the "Languages" intent and click on **Add follow-up intent**:



6. Click on **Custom** in the revealed list:



Dialogflow automatically names the follow-up intent "Languages - custom", and the arrow indicates the relationship between the intents.



## Intent matching with follow-up intents

Follow-up intents are only matched after the parent intent has been matched. Since this intent is only matched after the "Languages" intent, we can assume that the user has just been asked the question "How long have you known \$language?". You'll now add Training Phrases indicating users' likely answers to that question.

## • Languages - custom

SAVE

### Training phrases ?

Search in user s  

” Add user expression

” for 5 years 

” about 4 days

” 3 years

### Action & parameters ?



Languages.Languages-custom

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?
<input type="checkbox"/>	duration	@sys.duration	\$duration	<input type="checkbox"/>

1. Click on **Intents** in the left menu and then click on the "Languages - custom" intent.
2. Add the following Training Phrases:
  - 3 years
  - about 4 days
  - for 5 years
3. Click the **Save** button.

### Try it out

Try this out in the **Dialogflow simulator** on the right. First, match the "Languages" intent by entering the query I know French. Then, answer the question How long have your known \$language? with about 2 weeks.

I know French

about 2 weeks

See how it works in Google Assistant.

Agent

Domains

USER SAYS

COPY CURL

I know French

DEFAULT RESPONSE

PLAY

Wow! I didn't know you knew French. How long have you known French?

CONTEXTS

RESET CONTEXTS

languages-followup

INTENT

Languages

ACTION

Not available

PARAMETER

VALUE

language

French

See how it works in Google Assistant.

Agent

Domains

USER SAYS

COPY CURL

about 2 weeks

DEFAULT RESPONSE

Not available

CONTEXTS

RESET CONTEXTS

languages-followup

INTENT

Languages - custom

ACTION

Languages.Languages-custom

PARAMETER

VALUE

duration

{"amount":2,"unit":"wk"}

Despite there being no response for the second query ("about 2 weeks"), we can see our query is matched to the correct intent ("Languages - custom") and the duration parameter is correctly parsed ("2 weeks").

## Intents and contexts

Now that your follow-up intent is being matched correctly, you need to add a response. In "Languages - custom" you've only asked for the duration the user has known the language, and not the referenced language itself.

To respond with a parameter gathered from the "Languages" intent, you need to know how follow-up intents work. Follow-up intents use contexts to keep track of if a parent intent has been triggered. If you inspect the "Languages" intent, you'll see "Languages-followup" listed as an **Output context**, prefaced by the number 2:

## • Languages

SAVE



### Contexts ?



Add input context

2

Languages-followup



Add output context



After the "Languages" intent is matched, the context "Languages-followup" is attached to the conversation for two turns. Therefore, when the user responds to the question, "How long have you known \$language?", the context "Languages-followup" is active. Any intents that have the same **Input context** are heavily favored when Dialogflow matches intents.

1. Click on **Intents** in the left navigation and then click on the "Languages - custom" intent.

## • Languages - custom

SAVE



### Contexts ?



Languages-followup



Add input context

Add output context



You can see that the intent has the same input context ("Languages-followup") as the output context of "Languages". Because of this, "Languages - custom" is much more likely to be matched after the "Languages" intent is matched.

## Contexts and parameters

Contexts store parameter values, which means you can access the values of parameters defined in the "Languages" intent in other intents like "Languages - custom".

1. Add the following response to the "Languages - custom" intent and click the **Save** button:

- I can't believe you've known #languages-followup.language for \$duration!

## • Languages - custom

SAVE



### Responses ?



DEFAULT

GOOGLE ASSISTANT











#### Text response



- 1 I can't believe you've know #Languages-followup.language for \$duration!
- 2 Enter a text response variant

**Save** the changes. Now you can query your agent again and get the proper response. First enter "I know French", and then respond to the question with "1 month".

I know French	1 month
 See how it works in <a href="#">Google Assistant</a> . 	 See how it works in <a href="#">Google Assistant</a> . 
<div><div>Agent</div><div>Domains</div></div>	<div><div>Agent</div><div>Domains</div></div>
USER SAYS <a href="#">COPY CURL</a> I know French	USER SAYS <a href="#">COPY CURL</a> 1 month
<div> DEFAULT RESPONSE  <a href="#">PLAY</a></div> <p>Wow! I didn't know you knew French. How long have you known French?</p>	<div> DEFAULT RESPONSE  <a href="#">PLAY</a></div> <p>I can't believe you've know French for 1 mo</p>
CONTEXTS <a href="#">RESET CONTEXTS</a> <div>languages-followup</div>	CONTEXTS <a href="#">RESET CONTEXTS</a> <div>languages-followup</div>
INTENT <a href="#">Languages</a>	INTENT <a href="#">Languages - custom</a>
ACTION <i>Not available</i>	ACTION Languages.Languages-custom
PARAMETER      VALUE	PARAMETER      VALUE
language      French	duration      {"amount":1,"unit":"mo"}

You should see that the language parameter value is retrieved from the context.

## Next steps

If you have any questions or thoughts, let us know on the [Dialogflow Google Plus Community](#). We'd love to hear from you!

Now that you've completed your first agent, you can extend your response logic with fulfillment and consider which additional platforms you want to support via Dialogflow's one-click integrations.

Fulfillment allows you to provide programmatic logic behind your agent for gathering third-party data or accessing user-based information.

- [Fulfillment](#)
- [How to get started with fulfillment](#)
- [Integrate your service with fulfillment](#)
- [Integrate your service with Actions on Google](#)

Dialogflow's integrations make your agent available on popular platforms like Facebook Messenger, Slack and Twitter.

- [Integrations Overview](#)
- [Facebook Messenger](#)
- [Slack](#)
- [Twitter](#)

You might also want to check out:

- [Contexts](#)
- [Dialogflow and Actions on Google](#)

## End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.



Last Tested Date: 12-14-2018

Last Updated Date: 12-14-2018

©2019 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.