# **Smart Home Assistant**

In this project, you'll make a "smart" personal assistant to help out around some tasks around the house, such as turning lights on and off, or turning fans on and off.

You'll give your assistant a command in the same way you might talk to a person, and see how well it responds to your request.

## What you'll need to know:



i

🔁 How to use the Python shell

How to write Python scripts

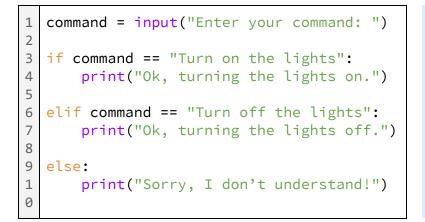


# Part 1: What can our assistant do?

Instead of using real light bulbs and appliances, we're going to build a **prototype** that will **1)** wait for a command, **2)** decide which task to perform, and **3)** tell you what it did.

A **prototype** is a simpler version of your project that lets you test out an idea

- 1. Create a new python script, and save it as smart\_home.py
- 2. Add the following code. Before running it, **predict** what you think it will do!



Can you guess what this **algorithm** will do when you run it?

Try reading it aloud, line-by-line, and discuss your predictions w/ your partner.

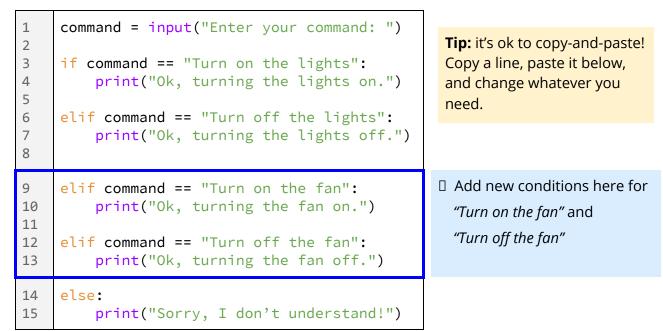
What do you think the if, elif, and else statements are for?

- 3. Run your script. In the **shell**, you should see: Enter your command:
- **4.** Type in a command such as "Turn on the lights" and press *Enter*.
- 5. Keep trying different commands (you'll need to run the script each time)

#### Part 2: Adding more conditions

Right now, our assistant can only control lights. What if we also want to control a fan? We'll need to program our assistant to handle more **conditions**.

6. Add another condition to the if-statement (start after line 8):

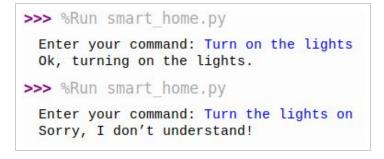


7. Run the script and test it out! Make sure all 4 conditions work.

How smart is our assistant, really?

What happens when you say "Turn the lights on" instead of "Turn on the lights"?

8. Run the script, and enter both commands to see what happens:



**Reflect**: why do you think this happens? What might you do to fix it?

## Part 3: How to train your robot

As you may have realized, there are *many* different ways you could ask someone to turn on and off the lights. Adding all the possible commands in our code would take forever!

Next, we'll try a better approach: teaching the computer to recognize commands for itself. This technology is called **Machine Learning** 



# Create a new project and add training data

- Learn how to create a new project in the *Machine Learning Reference Guide*.
   Refer to the *Adding training data* and *Recognizing text* sections in the following steps.
- Create a new project for recognizing text, and add training data for each possible outcome: lights\_on, lights\_off, fan\_on, & fan\_off.
- **3.** Try to think of a bunch of the different ways you might tell someone to turn on the lights. Be creative!
- **4.** Click on *Add example* to add at least 6 to 8 example commands for each of your labels:

lights_on	lights_off	fan_on	fan_off
lights on its dark in here	its too bright in here	fan on its hot	fan off its cold
lamp on turn the lamp on	lamp off lower the lights	it's hot in here I'm hot	I'm cold It's too cold in here
good morning	lights out goodnight	It's too hot in here	It's cold in here
turn the lights on	turn the lights off	turn the fan on	turn the fan off
turn on the light	turn off the light lights off	turn on the fan	turn off the fan
+ Add example	+ Add example	+ Add example	+ Add example

- 5. Click the *Back to Project* link, then click *Learn & Test*.
- **6.** Scroll down to the bottom, and click *"Train new machine learning model".* Wait for the training to complete (this might take a minute or two).
- Once the training has completed, you will have a machine learning model.
   Try typing some test commands in the text box below to test out your model!



A **machine learning model** is the result of training.

You can interact with a model it by giving it input (such as text, images, or numbers), and it will give you some answer based on what it has learned.

# Update your Python code

- 8. Highlight the **API key** text on your project page, and press *Ctrl+C* to copy it.
- **9.** Go back to your Python script, and change your code to use the new **model**:

```
1
    import ml4k
                                                    1. Add import ml4k at the top
2
                                                    2. Paste your API key using Ctrl+V
3
    API_KEY = "PASTE-API-KEY-HERE"
4
    model = ml4k.Model(API_KEY)
5
                                                    3. Use the classify function to
6
    command = input("Enter your command: ")
                                                       send the command to your model,
7
    result = model.classify(command)
                                                       and store the result in a variable
8
    label = result["class_name"]
                                                       called result.
9
                                                    4. Create a variable called label and
    if label == "lights_on":
10
                                                       set it to result["class_name"].
         print("0k, turning the lights on.")
11
                                                       This tells you which of the four
12
                                                       "buckets" the model chooses.
13
    elif label == "lights_off":
         print("0k, turning the lights off.")
14
                                                    5. Replace command with label, and
15
                                                       use the label names you created in
    elif label == "fan on":
16
                                                       your training data.
17
         print("Ok, turning the lights off.")
18
                                                    6. Remove the else section. The
    elif label == "fan_off":
19
                                                       model will only return one of the
         print("Ok, turning the lights off.")
20
                                                       four options.
```

**10.** Run your script! Try entering commands that you **didn't** enter during training. How well did it do? If you got an unexpected answer, try adding more training data.

#### Advanced: How confident is our assistant?

What happens if you enter the command "Turn up the music!"? You'll notice the model will still give you one of the four options. Sometimes, the model will give you an answer, but it won't be very confident about it. We need a way to tell our user when we don't know what to do with a certain command.

Use result["confidence"] to get a number between 0 and 100, and store it in a **variable** called that will tell you how confident the model is in its answer. See if you can add an **if statement** that *first* checks the confidence level and display an appropriate response. For help, refer to the **Variables**, **Boolean Expressions** and **If Statements** chapters in your **Python Reference Guide**.

#### Extra: Show an image instead of text!

- 1. Find some images online of a light bulb and fan on and off.
- 2. Add import webbrowser webbrowser at the top of your script
- 3. Instead of print(), use webbrowser.open("url-goes-here") to show the image!