Contents

[Robot Glossary 2](#_Toc520805899)

[Installing the OT app and Connecting to a PC 4](#_Toc520805900)

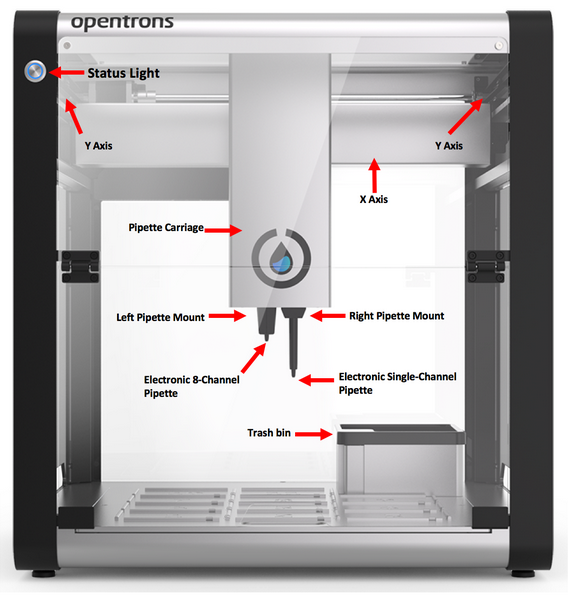
[Deck Calibration 6](#_Toc520805901)

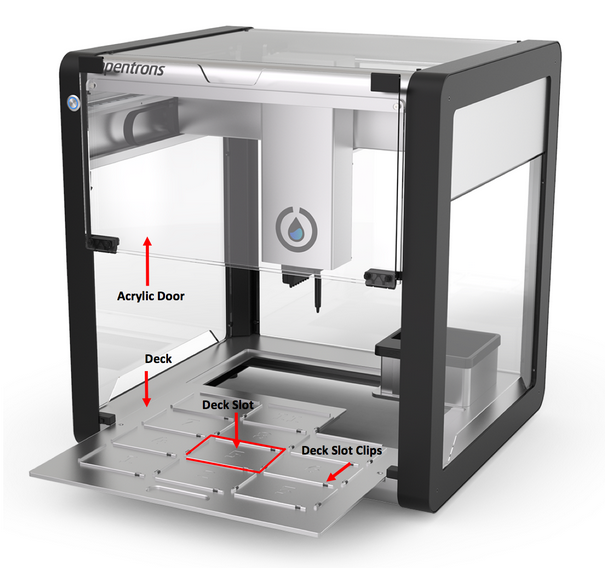
[Running a protocol 7](#_Toc520805902)

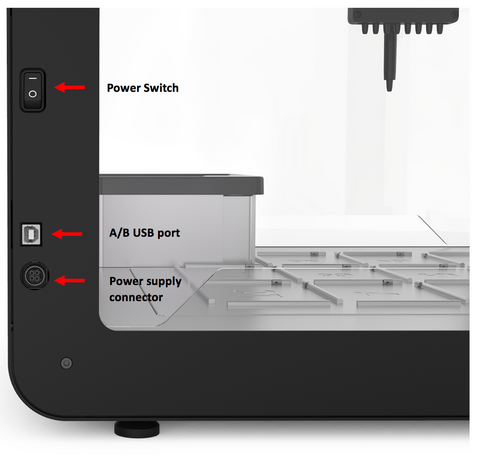
[How to run the Solution maker procedure on OT2 9](#_Toc520805903)

[OT2-Commands for Python script 10](#_Toc520805904)

# Robot Glossary







# Installing the OT app and Connecting to a PC

Download the OT2 App :- <https://support.opentrons.com/ot-2/getting-started-software-setup/download-the-app>

Or

Use the setup in P:\Employee Public Folders\dpatel\DOE ROBOT named “Opentrons-v3.3.0-beta.1-win-b6611-edge”

Create a desktop shortcut for the app if you don't already have one. Once there you will do the following:

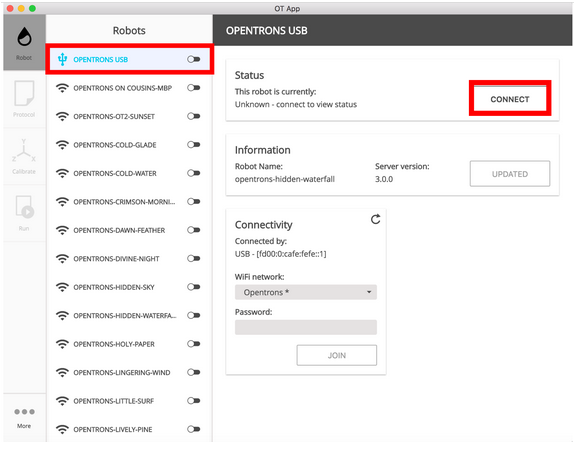
Right click Opentrons shortcut on Desktop  
*Properties > Shortcut*Set “Target” to `.../Opentrons.exe --discovery.enabled`

You can connect your computer to the OT-2 in two ways, wirelessly or hardwired to the side panel using the USB cable provided with your machine.

**Procedure to connect via USB:-**

1. Connect the Power Supply and turn on the robot with the power switch.
2. Open the OT2 app.
3. To connect the robot via USB, plug the A/B USB cable into the A/B USB port. Once you open the app, the available ports/robots will show up. Make sure you are connecting to OPENTRONS USB. You can select the connect button or the switch next to OPENTRONS USB.

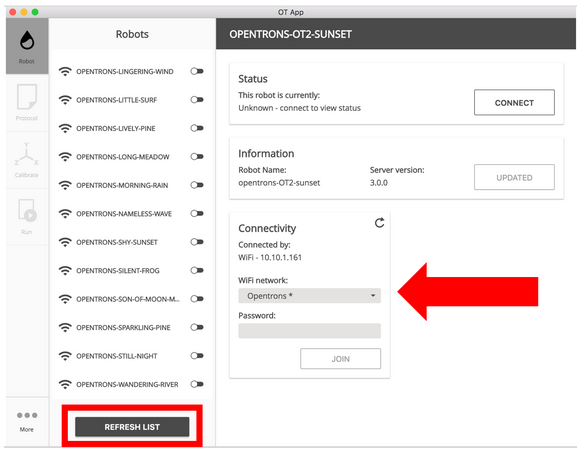
When the robot is successfully connected, the toggle button will turn blue.



**Procedure to connect via Wi-Fi:-**

You can also connect the robot via WI-FI. Make sure to refresh your port. Please enter the labs WI-FI and password. Locate your robot name under Information. Once the robot shows up with the WI-F- signal,you can select the connect button or the switch next to OPENTRONS-`robot\_name`

When the robot is successfully connected, the toggle button will turn blue.



# Deck Calibration

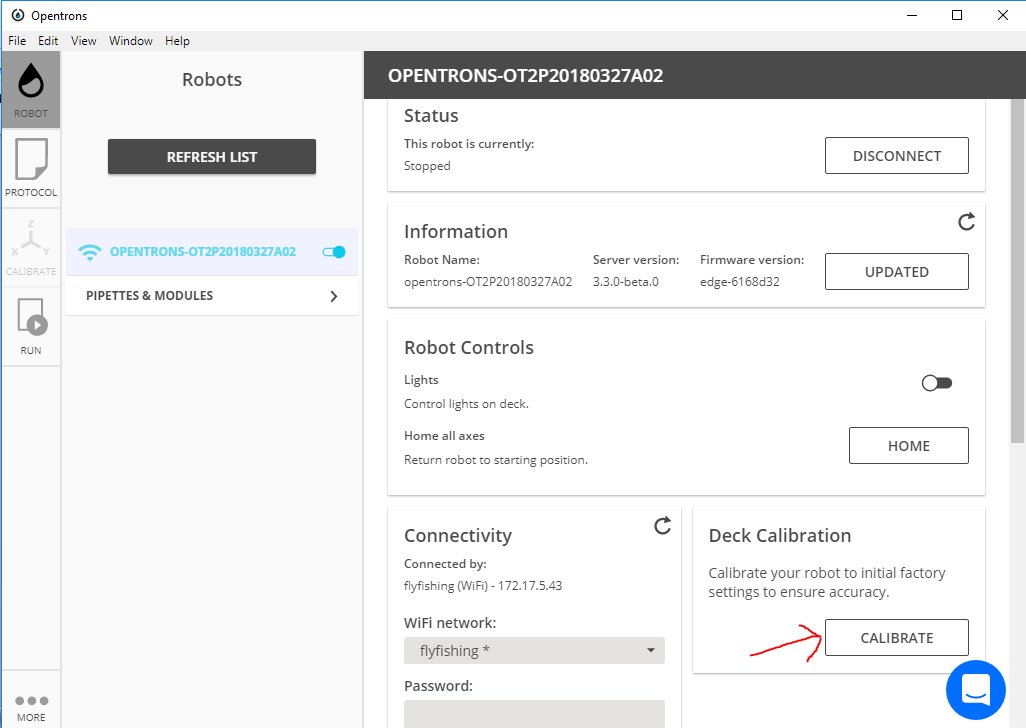
Deck calibration **ONLY** needs to be performed when you move the robot from one place to other. This calibrations tells the robot where the slots of the robot are located exactly.

**Materials Needed:**

* Mounted pipette
* Tip

**Procedure:**

1. Connect to the robot.
2. Click the Calibrate button under Deck Calibration as shown in the figure



1. Follow the on screen instructions.
2. As a last step, Remove the tip from the pipette and select "Finish and Restart Robot" to finish the calibration process and have the new settings take effect. It may take several minutes for your robot to restart.

# Running a protocol

**Materials Needed:**

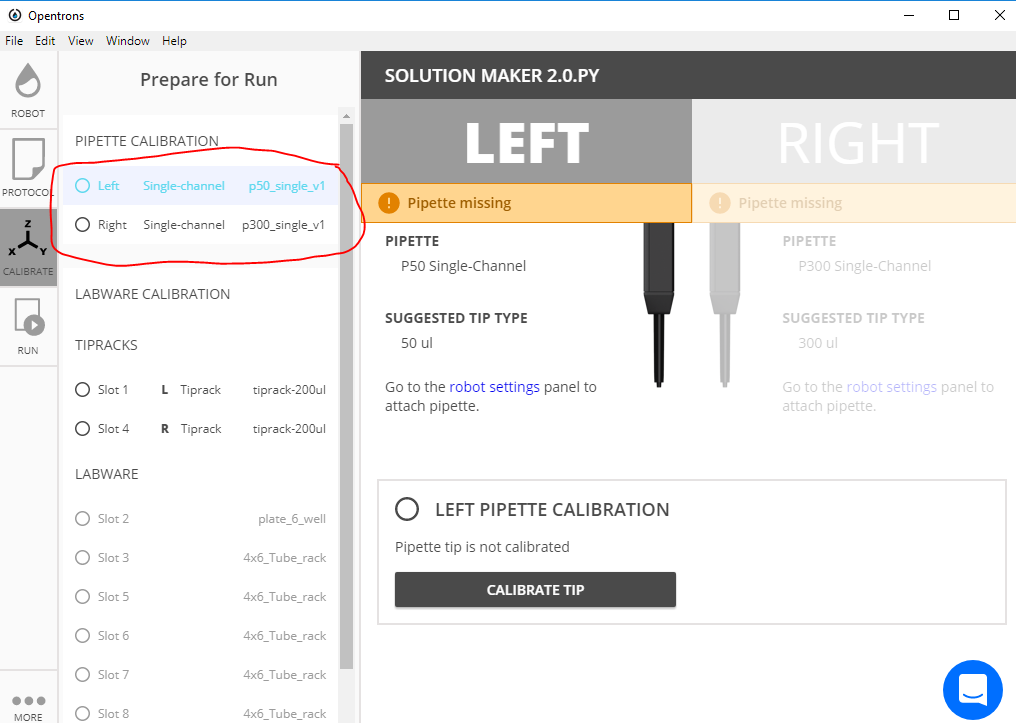
* A protocol (A python script with the commands for the robot to follow)
* Specific Labware described in the protocol
* Tips
* Pipettes described in the protocol

**Procedure:**

1. Open the OT2 app and connect to the robot.
2. Click the Protocols tab. Click the Open button and upload your protocol.
3. Continue to the calibration screen by either clicking the calibrate tab or the calibrate word under the check mark
4. Calibration :-
   1. First we begin with pipette Calibration aka tip probing

**Note:- Pipette calibration tells the robot where exactly the center axis of the robot is with respect to the robots home position. This is required only when the robot has malfunctioned in previous protocols. Meaning you done need to do a pipette calibration for every protocol.**

* 1. Click on the left or right or both (One after other) and perform the calibration steps as instructed on screen.

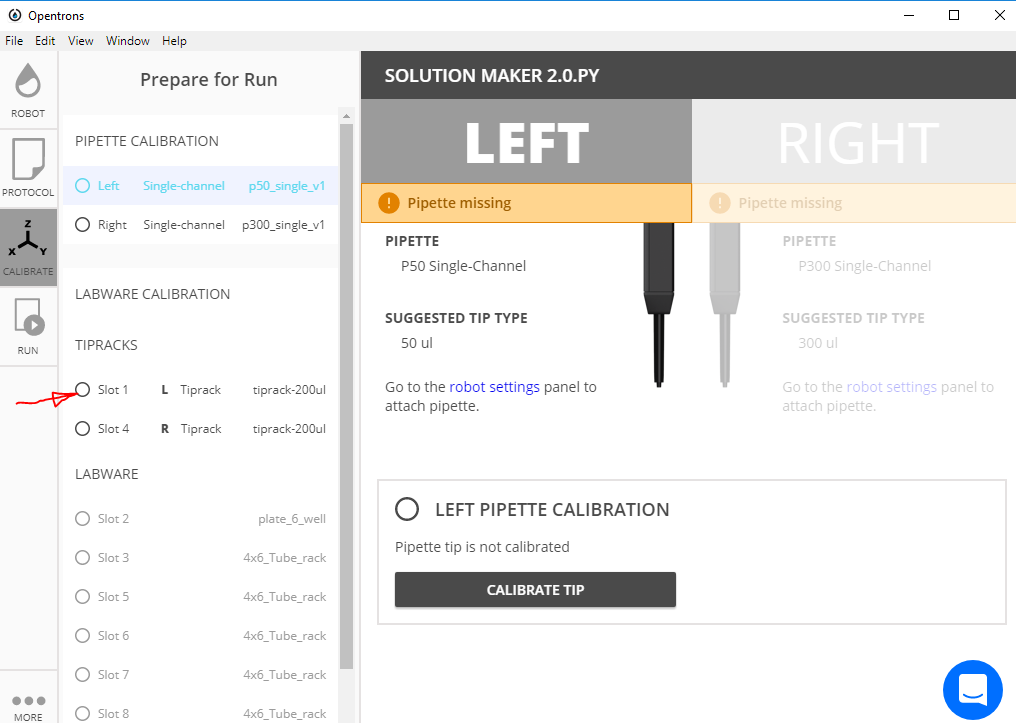


* 1. After tip probing, we perform labware calibration. Click on the first tip rack and follow the on Screen instructions to calibrate all the labwares.

**Note:- If your protocol uses more than one tip racks then the pipettes will pick up the tips and drop them in the same locations for all except the last tip rack. The tip picked up from the last tip rack will be used to calibrate the remaining labwares. The tip is then returned to the tip tack.**

**Note:- Once calibrated, labware location is saved. You won’t need to do labware calibration if you are rerunning/resetting the same protocol.**

**If you change the type of a labware in the python script for a certain slot, for example, changing the source of liquid from a 12-row-trough to a 6-well-plate, you don’t need to calibrate all the labwares again. Just stop the labware calibration procedure once that "changed" labware is calibrated. To stop the calibration process click the home button in the Robot Tab and manually remove the pipette tip**



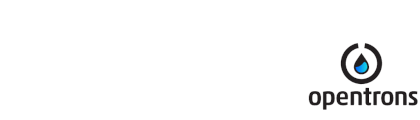
1. Click on the Run tab and click the Start Run button.

# How to run the Solution maker procedure on OT2

**Things Needed:**

* Matlab
* Sublime text. (Download link <https://www.sublimetext.com/3> )
* Excel
* OT2 App
* The following files.
  + Solution Maker 2.0.py (Python Script)
  + Solution Maker 2.0.xlsx (Excel File)
  + Solution\_Maker\_2.m (Matlab Script)

**Workflow**

****

1. First you input the list of volumes in the *Solution Maker 2.0.xlsx* file in the specified columns of the *Input* Sheet.

**NOTE:- For each component the volumes are bunched into groups of 24. This is because our custom labware that fits into the Slots of OT2 can accommodate 24 tubes at a time..**

**Out of the 11 available slots on the deck, 2 are dedicated to the Tip racks, 1 for the 6 well plate (Source), that leaves us with 8 available slots. So we can use a maximum of 8x24=192 volumes per component.**

1. Save and close the excel file. Run *Solution\_Maker\_2.m.*  You will be asked to enter the number of components and labware you are going to have. Enter those values and continue running the script. Matlab will sort and arrange the all volumes and export those to the same excel file.
2. Again, open *Solution Maker 2.0.xlsx.*  In the output sheet copy and paste the strings in cells C2 and C3 into the *Solution Maker 2.0.py*‘s line 65 and 67 respectively.   
   These will become the list of volumes and wells for our protocol.
3. Save *Solution Maker 2.0.py.*  and run it in the OT2 app.
4. Follow the calibration steps previously described for the labware calibration

**Note:- You don’t need to do the pipette calibration over and over again.**

# OT2-Commands for Python script

**Labware commands**

1. labware.load()

Example :- tubes = labware.load('T25-flask', '2')

more\_tubes = labware.load('T25-flask','2', 'any-name-you-want', share=True)

Arguments:- 1. Name of the labware (All names are case-sensitive)

2. Slot on the deck

3. optional argument- Used to give a unique name to a specific labware. Unique names are useful in a few scenarios. First, they allow the container to have independent calibration data from other containers in the same slot. In the example above, the container named ‘any-name-you-want’ will assume different calibration data from the unnamed plate, even though they are the same type and in the same slot.

Notes:- 1. Some containers might only take up half a slot. You must explicitly say share=True, as the last argument indicating that it is okay to share the slot.

1. labware.list()

Once the labware module is loaded, you can see a list of all containers currently inside the API by calling labware.list()

**General Commands**

1. wells(‘A1’,’B2’)

wells() takes the name of the well as an argument, and will return the well at that location. Returns multiple wells with multiple arguments

**Argument** – name of the well

**Note**- Can also use wells('A2',to='A7') to select wells from A2 to A7

1. rows(‘A’) cols(‘1’)

You can access a specific row or column by using the rows() and cols() methods on a container. These will return all wells within that row or column.

**Argument**- name of the row or column

**Pipette Commands**

1. instrument.**Pipette()**

Example:- instruments.P300\_Multi(mount="left",tip\_racks=[m300rack], aspirate\_flow\_rate=200, dispense\_flow\_rate=600)

Arguments:- 1. mount (str) – The mount of the pipette’s actuator on the Opentrons robot (‘left’ or ‘right’)

2. trash\_container (Container) – Sets the default location. drop\_tip() will put tips (Default: fixed-trash)

3. tip\_racks (list) – A list of Containers for this Pipette to track tips when calling pick\_up\_tip() (Default: [])

4. aspirate\_flow\_rate (int) – The speed (in ul/sec) the plunger will move while aspirating (Default: See Model Type)

5. dispense\_flow\_rate (int) – The speed (in ul/sec) the plunger will move while dispensing (Default: See Model Type)

Note:- m300rack is a labware defined for tip rack on a specific slot. Called using square bracket

Link to website- <https://docs.opentrons.com/pipettes.html>

<https://docs.opentrons.com/api.html>

**Liquid Control**

The Default height for pipette aspirate and dispense is 1mm form the bottom of the well

1. aspirate()

To aspirate is to pull liquid up into the pipette’s tip. When calling aspirate on a pipette, we can specify how many microliters, and at which location, to draw liquid from:

**Example-** pipette.aspirate(50, plate.wells('A1'))

**Notes-** **1**. If you don’t specify the location the pipette in this circumstance will aspirate from its current location

**2.** If you don’t specify the amount of liquid to aspirate it will by default fill up the remaining volume in its tip. For example, for a 300uL pipette, if you already have 100uL in the tip, the pipette will aspirate another 200uL.

1. dispense()

To dispense is to push out liquid from the pipette’s tip. It’s usage in the Opentrons API is nearly identical to aspirate(), in that you can specify microliters and location, only microliters, or only a location:

1. blow\_out()

To blow out is to push an extra amount of air through the pipette’s tip, so as to make sure that any remaining droplets are expelled. When calling blow\_out() on a pipette, we have the option to specify a location to blow out the remaining liquid. If no location is specified, the pipette will blow out from its current position.

**Example**- pipette.blow\_out()

1. touch\_tip()

To touch tip is to move the pipette’s currently attached tip to four opposite edges of a well, for the purpose of knocking off any droplets that might be hanging from the tip. When calling touch\_tip() on a pipette, we have the option to specify a location where the tip will touch the inner walls. If no location is specified, the pipette will touch tip inside its current location.

Example- 1. pipette.touch\_tip().Touch tip within current location

2. pipette.touch\_tip(-2).Touch tip 2mm below the top of the current location

3. pipette.touch\_tip(plate.wells('B1')). Touch tip within plate at B1

1. mix()

Mixing is simply performing a series of aspirate() and dispense() commands in a row on a single location.

**Arguments**- Repetitions(int)- Number of Repetitions

Volume(int)- Amount of liquid to be mixed

Location- where to mix it

**Example-** 1.pipette.mix(4, 100, plate.wells('A2')) .Mix 4 times, 100uL, in plate:A2

2. pipette.mix(3, 50). Mix 3 times, 50uL, in current location

3. pipette.mix(2). Mix 2 times, pipette's max volume, in current location

1. air\_gap()

Some liquids need an extra amount of air (in ul) in the pipette’s tip to prevent it from sliding out. A call to air\_gap() with a microliter amount will aspirate that much air into the tip.

**Example**- pipette.air\_gap(20)

1. move\_to()

Pipette’s are able to move\_to() any location on the deck.

**Example-** pipette.move\_to(tiprack.wells('A1')). Move to the first tip in our tip rack

1. delay()

To have your protocol pause for any given number of minutes or seconds, simply call delay() on your pipette. The value passed into delay() is the number of minutes or seconds the robot will wait until moving on to the next commands.

**Example-** 1**.** pipette.delay(seconds=2) . Pause for 2 seconds

2. pipette.delay(minutes=5) . Pause for 5 minutes

3. pipette.delay(minutes=5, seconds=2) . Pause for 5 minutes and 2 seconds

**Commands that go with pipettes**

1. **Pipette**.transfer()

The example below will transfer 100 uL from well 'A1' to well 'B1', automatically picking up a new tip and then disposing it when finished.

**Example**:- p300.transfer(100, plate.wells('A1'), plate.wells('B1'))

Note:- Moving 100uL from one well to another:

**Optional** **Arguments-** 1**.**new\_tip='always' or ’never’.Always/never pick up new tip

2. trash=False. By default, the transfer command will drop the pipette’s tips in the trash container. However, if you wish to instead return the tip to its tip rack, use this argument.

3.touch\_tip. Touch tip to each well's edge

4. blow\_out=True. Empty the tip completely during dispensing

5. air\_gap=20. Add 20uL of air after each aspirate.

6. Mix\_before=(2,50). A mix can be performed before every aspirate by setting mix\_before=. The value of mix\_before= must be a tuple, the 1st value is the number of repetitions; the 2nd value is the amount of liquid to mix. Used before aspiration.

7. Mix\_before=(2,50). Same logic. Used for mixing after dispensing.

**Hyperlink-** <https://docs.opentrons.com/complex%20commands.html?highlight=distribute>

Save time and tips with the distribute() and consolidate() commands. These are nearly identical to transfer(), except that they will combine multiple transfer’s into a single tip.

1. distribute()

Volumes from the same source well are combined within the same tip, so that one aspirate can provide for multiple dispenses. The following example will dispense 55ul (each) into all rows that have ‘2’ in them

**Example**:- pipette.distribute(55, plate.wells('A1'), plate.rows('2'), disposal\_vol=10)

**Note-** It is recommended to aspirate an extra amount of liquid to be disposed of after distributing. disposal\_vol=10 can be set as an optional argument. Includes extra liquid to make dispenses more accurate.

If you do not specify a disposal\_vol, the pipette will by default use a disposal\_vol equal to it’s min\_volume. This can be set by pipette.min\_volume = 20

1. consolidate()

Volumes going to the same destination well are combined within the same tip, so that multiple aspirates can be combined to a single dispense. The following example will aspirate 30ul from all rows that have ‘2’ in them and then dispense into A1

**Example**:- pipette.consolidate(30, plate.rows('2'), plate.wells('A1'))

**Note**- If there are multiple destination wells, the pipette will never combine their volumes into the same tip.

1. pick\_up\_tip()

Before any liquid handling can be done, your pipette must have a tip on it. This command will move the pipette over to the specified tip, the press down into it to create a vacuum seal. The example below picks up the tip at location 'A1'.

**Example**- pipette.pick\_up\_tip(tiprack.wells('A1'))

1. drop\_tip()

Once finished with a tip, the pipette will autonomously remove the tip when we call drop\_tip(). We can specify where to drop the tip by passing in a location. The example below drops the tip back at its originating location on the tip rack. If no location is specified, it will go to the fixed trash location on the deck.

**Example**- pipette.drop\_tip(tiprack.wells(‘A1’))

**Note-** If you wish to return the tip to its original pickup location after its work then just simply type return\_tip().