opentrons

You are currently viewing the documentation for the Opentrons OT-1. To view documentation for the OT-2, click here.

Liquid Handling

The `instruments` module gives your protocol access to the `Pipette`, which is what you will be primarily using to create protocol commands.

Creating a Pipette

```
'''
Examples in this section require the following
'''
from opentrons import instruments
```

Axis and Max Volume

To create a `Pipette`, you must give it an axis and a max_volume. The axis can be either `'a'` or `'b'`, and the volume is whatever your hand pipette is calibrated for. In this example, we are using a 200uL pipette.

```
pipette = instruments.Pipette(
    axis='b',
    name='my-p200',
    max_volume=200)
```

Minimum Volume

The minimum allowed volume can be set for each pipette. If your protocol attempts to aspirate or dispense a volume below this volume, the API will give you a warning.

```
pipette = instruments.Pipette(
    axis='b',
    name='my-p200',
    max_volume=200,
    min_volume=20)
```

Channels

Pipettes can also be assigned a number of channels, either `channel=1` or `channel=8`. If you do not specify, it will default to `channel=1` channel.

```
pipette = instruments.Pipette(
    axis='b',
    name='my-p200-multichannel',
    max_volume=200,
    min_volume=20,
```

The speeds at which the pipette will aspirate and dispense can be set through `aspirate_speed` and `dispense_speed`. The values are in millimeters/minute, and default to `aspirate_speed=300` and `dispense_speed=500`.

```
pipeipette = instruments.Pipette(
    axis='b',
    name='my-p200-multichannel',
    max_volume=200,
    min_volume=20,
    channels=8,
    aspirate_speed=200,
    dispense_speed=600)
```

Tip Handling

When we handle liquids with a pipette, we are constantly exchanging old, used tips for new ones to prevent cross-contamination between our wells. To help with this constant need, we describe in this section a few methods for getting new tips, and removing tips from a pipette.

This section demonstrates the options available for controlling tips

```
'''
Examples in this section expect the following
'''
from opentrons import containers, instruments

trash = containers.load('point', 'D2')
tiprack = containers.load('tiprack-200ul', 'B1')

pipette = instruments.Pipette(axis='a')
```

Pick Up Tip

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

```
pipette.pick_up_tip(tiprack.wells('A1'))
```

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 below example drops the tip back at its originating location on the tip rack.

```
pipette.drop_tip(tiprack.wells('A1'))
```

Instead of returning a tip to the tip rack, we can also drop it in a trash container.

```
pipette.pick_up_tip(tiprack.wells('A2'))
pipette.drop_tip(trash)
```

Return Tip

When we need to return the tip to its originating location on the tip rack, we can simply call `return_tip()`. The example below will

opentrons

Tips Iterating

Automatically iterate through tips and drop tip in trash by attaching containers to a pipette

```
'''
Examples in this section expect the following
'''
from opentrons import containers, instruments

trash = containers.load('point', 'D2')
tip_rack_1 = containers.load('tiprack-200ul', 'B1')
tip_rack_2 = containers.load('tiprack-200ul', 'B2')
```

Attach Tip Rack to Pipette

Tip racks and trash containers can be "attached" to a pipette when the pipette is created. This give the pipette the ability to automatically iterate through tips, and to automatically send the tip to the trash container.

Trash containers can be attached with the option `trash_container=TRASH_CONTAINER`.

Multiple tip racks are can be attached with the option `tip_racks=[RACK_1, RACK_2, etc... ]`.

```
pipette = instruments.Pipette(
    axis='b',
    tip_racks=[tip_rack_1, tip_rack_2],
    trash_container=trash
)
```

Note:

The `tip_racks=` option expects us to give it a Python list, containing each tip rack we want to attach. If we are only attaching one tip rack, then the list will have a length of one, like the following:

`tip_racks=[tiprack]`

Iterating Through Tips

Now that we have two tip racks attached to the pipette, we can automatically step through each tip whenever we call `pick_up_tip()`. We then have the option to either `return_tip()` to the tip rack, or we can `drop_tip()` to remove the tip in the attached trash container.

```
pipette.pick_up_tip()  # picks up tip_rack_1:A1
pipette.return_tip()
pipette.pick_up_tip()  # picks up tip_rack_1:A2
pipette.drop_tip()     # automatically drops in trash

# use loop to pick up tips tip_rack_1:A3 through tip_rack_2:H12
for i in range(94 + 96):
    pipette.pick_up_tip()
    pipette.return_tip()
```

If we try to `pick_up_tip()` again when all the tips have been used, the Opentrons API will show you an error.

```
# this will raise an exception if run after the previous code block
# pipette.pick_up_tip()
```

Select Starting Tip

Calls to `pick_up_tip()` will by default start at the attached tip rack's `'A1'` location. If you however want to start automatic tip iterating at a different tip, you can use `start_at_tip()`.

```
pipette.reset()

pipette.start_at_tip(tip_rack_1['C3'])
pipette.pick_up_tip()  # pick up C3 from "tip_rack_1"
pipette.return_tip()
```

Get Current Tip

Get the source location of the pipette's current tip by calling `current_tip()`. If the tip was from the `'A1'` position on our tip rack, `current_tip()` will return that position.

```
print(pipette.current_tip())  # is holding no tip

pipette.pick_up_tip()
print(pipette.current_tip())  # is holding the next available tip

pipette.return_tip()
print(pipette.current_tip())  # is holding no tip
```

will print out...

```
None
<Well D3>
None
```

Liquid Control

This is the fun section, where we get to move things around and pipette! This section describes the `Pipette` object's many liquid-handling commands, as well as how to move the `robot`.

```
'''
Examples in this section expect the following
'''
from opentrons import containers, instruments

plate = containers.load('96-flat', 'B1')
pipette = instruments.Pipette(axis='b', max_volume=200)
```

Aspirate

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

```
pipette.aspirate(50, plate.wells('A1'))  # aspirate 50uL from plate:A1
```

from it's current location (which we previously set as `plate.wells('A1')`).

```
pipette.aspirate(50)                    # aspirate 50uL from current position
```

Now our pipette's tip is holding 100uL.

We can also specify only the location to aspirate from. If we do not tell the pipette how many micoliters to aspirate, it will by default fill up the remaining volume in it's tip. In this example, since we already have 100uL in the tip, the pipette will aspirate another 100uL

```
pipette.aspirate(plate.wells('A2'))     # aspirate until pipette fills from plate:A2
```

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:

```
pipette.dispense(50, plate.wells('B1')) # dispense 50uL to plate:B1
pipette.dispense(50)                    # dispense 50uL to current position
pipette.dispense(plate.wells('B2'))     # dispense until pipette empties to plate:B2
```

That final dispense without specifying a micoliter amount will dispense all remaining liquids in the tip to `plate.wells('B2')`, and now our pipette is empty.

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 it's current position.

```
pipette.blow_out()                  # blow out over current location
pipette.blow_out(plate.wells('B3')) # blow out over current plate:B3
```

Touch Tip

To touch tip is to move the pipette's currently attached tip to the 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 it's current location.

```
pipette.touch_tip()                  # touch tip within current location
pipette.touch_tip(-2)                # touch tip 2mm below the top of the current location
pipette.touch_tip(plate.wells('B1')) # touch tip within plate:B1
```

Mix

Mixing is simply performing a series of `aspirate()` and `dispense()` commands in a row on a single location. However, instead of having to write those commands out every time, the Opentrons API allows you to simply say `mix()`.

The mix command takes three arguments: `mix(repetitions, volume, location)`

opentrons

Air Gap

Some liquids need an extra amount of air 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.

```
pipette.aspirate(100, plate.wells('B4'))
pipette.air_gap(20)
```

Moving

Demonstrates the different ways to control the movement of the Opentrons liquid handler during a protocol run.

```
'''
Examples in this section expect the following
'''
from opentrons import containers, instruments, robot

tiprack = containers.load('tiprack-200ul', 'A1')
plate = containers.load('96-flat', 'B1')

pipette = instruments.Pipette(axis='b')
```

Move To

Pipette's are able to `move_to()` any location on the deck.

For example, we can move to the first tip in our tip rack:

```
pipette.move_to(tiprack.wells('A1'))
```

You can also specify at what height you would like the robot to move to inside of a location using `top()` and `bottom()` methods on that location.

```
pipette.move_to(plate.wells('A1').bottom())  # move to the bottom of well A1
pipette.move_to(plate.wells('A1').top())     # move to the top of well A1
pipette.move_to(plate.wells('A1').bottom(2)) # move to 2mm above the bottom of well A1
pipette.move_to(plate.wells('A1').top(-2))   # move to 2mm below the top of well A1
```

The above commands will cause the robot's head to first move upwards, then over to above the target location, then finally downwards until the target location is reached. If instead you would like the robot to mive in a straight line to the target location, you can set the movement strategy to `'direct'`.

```
pipette.move_to(plate.wells('A1'), strategy='direct')
```

> **Note:**
>
> Moving with `strategy='direct'` will run the risk of colliding with things on your deck. Be very careful when using the option.

Usually the `strategy='direct'` option is useful when moving inside of a well. Take a look at the below sequence of movements, which first move the head to a well, and use 'direct' movements inside that well, then finally move on to a different well.

```
pipette.move_to(plate.wells('A1'))
```

opentrons

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.

```
pipette.delay(seconds=2)             # pause for 2 seconds
pipette.delay(minutes=5)             # pause for 5 minutes
pipette.delay(minutes=5, seconds=2)  # pause for 5 minutes and 2 seconds
```