OT-2 APIv2 exercises with solutions

June 10, 2020

1 Notes

1. For these exercise the aim is not to produce an executable script. Therefore, we will stick to a simulated environment.
2. The suggested solutions were not coded in the most elegant way possible.
3. Levels do not reflect the complexities of the task. Rather, it reflects the complexity of problems to be solved in Python.
4. Since the outputs from the `print()` function were too long, the `print()` functions were commented out in the suggested solutions. Please remove the hashtag before executing the script in Spyder.

2 Add water to lyophilized primers

2.1 Exercise

Task description: Assuming we just received 20 lyophilized primers, and we wish to automate the process of rehydrating them. We already put them on an Opentrons tube rack and we knew the amount of water that should go into each tube.

Well	Volume (µL)
A1	302
A2	256
A3	226
A4	310
A5	301
A6	268
B1	274
B2	295
B3	296
B4	156
B5	347
B6	332
C1	318
C2	267
C3	223
C4	289

Let us write a script to do so.

We start with: - Slot 1: Opentrons 1000 µL normal tips - Slot 2: Opentrons 6 Tube Rack with Falcon 50 mL Conical
Well A1: 20 mL of water - Slot 3: Opentrons 24 Tube Rack with Generic 2 mL Screwcap

Our robot is equipped with:
Right mount: P1000 single-channel pipette

2.2 Assistance

Please use the following dictionary:

[1]:
```
vol_info = {
    'A1': 302,
    'A2': 256,
    'A3': 226,
    'A4': 310,
    'A5': 301,
    'A6': 268,
    'B1': 274,
    'B2': 295,
    'B3': 296,
    'B4': 156,
    'B5': 347,
    'B6': 332,
    'C1': 318,
    'C2': 267,
    'C3': 223,
    'C4': 289
}
```

2.3 Suggested solution

[2]:
```
from opentrons import simulate
protocol = simulate.get_protocol_api(version = '2.4')

tip_rack = protocol.load_labware('opentrons_96_tiprack_1000ul', 1)
tube_rack_50 = protocol.
 ↪load_labware('opentrons_6_tuberack_falcon_50ml_conical', 2)
tube_rack_screw_caps = protocol.
 ↪load_labware('opentrons_24_tuberack_generic_2ml_screwcap', 3)

r_pipette = protocol.load_instrument(
    'p1000_single',
    'right',
    tip_racks=[tip_rack]
)
```

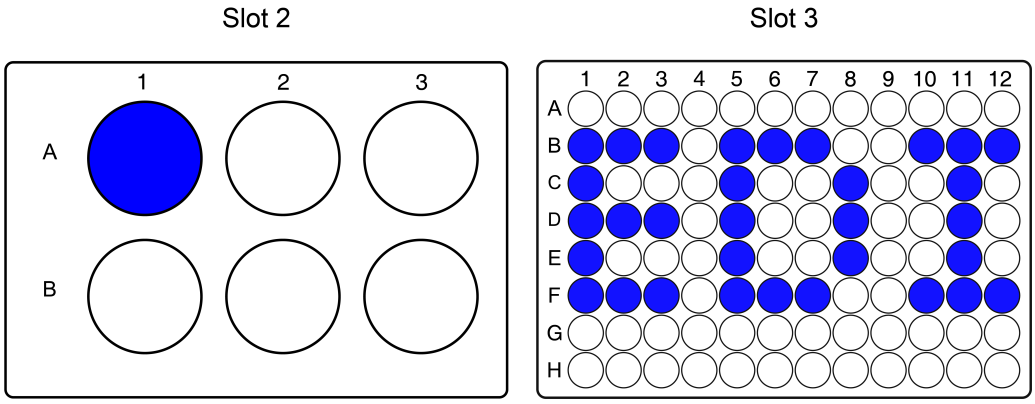
```
vol_info = {
    'A1': 302,
    'A2': 256,
    'A3': 226,
    'A4': 310,
    'A5': 301,
    'A6': 268,
    'B1': 274,
    'B2': 295,
    'B3': 296,
    'B4': 156,
    'B5': 347,
    'B6': 332,
    'C1': 318,
    'C2': 267,
    'C3': 223,
    'C4': 289
}

for dest_well, vol in vol_info.items():
    r_pipette.transfer(vol,
                tube_rack_50.wells_by_name()['A1'],
                tube_rack_screw_caps.wells_by_name()[dest_well]
               )

# print(*protocol.commands(), sep = '\n')
```

```
C:\Users\User\.opentrons\deck_calibration.json not found. Loading defaults
C:\Users\User\.opentrons\robot_settings.json not found. Loading defaults

Loading json containers…
Json container file load complete, listing database
Found 0 containers to add. Starting migration…
Database migration complete!
```

3 Create an "EDI" pattern on a 96-well plate (Level 1)

3.1 Exercise

Let us create the following pattern on a 96-well plate.



Task description:
Generate a pattern of "EDI" on a 96-well flat bottom plate by pipetting dyes from the source well to the destination wells.

We start with: - Slot 1: Opentrons 96 Tip Rack 300 µL - Slot 2: Opentrons 6 Tube Rack with Falcon 50 mL Conical
Well A1: 20 mL of dye - Slot 3: Corning 96 Well Plate 360 µL Flat

Our robot is equipped with:
Right mount: P300 single-channel pipette

3.2 Assistance

You can copy the following code which already contains the destination well names for the different letters.

[3]:
```
letter_E = ['B1','B2','B3','C1','D1','D2','D3','E1','F1','F2','F3']
letter_D = ['B5','B6','B7','C5','C8','D5','D8','E5','E8','F5','F6','F7'],
letter_I = ['B10','B11','B12','C11','D11','E11','F10','F11','F12']
```

3.3 Suggested solution

[4]:
```
from opentrons import simulate
protocol = simulate.get_protocol_api(version = '2.4')

tip_rack = protocol.load_labware('opentrons_96_tiprack_300ul', 1)
tube_rack = protocol.load_labware('opentrons_6_tuberack_falcon_50ml_conical', 2)
plate = protocol.load_labware('corning_96_wellplate_360ul_flat', 3)
```

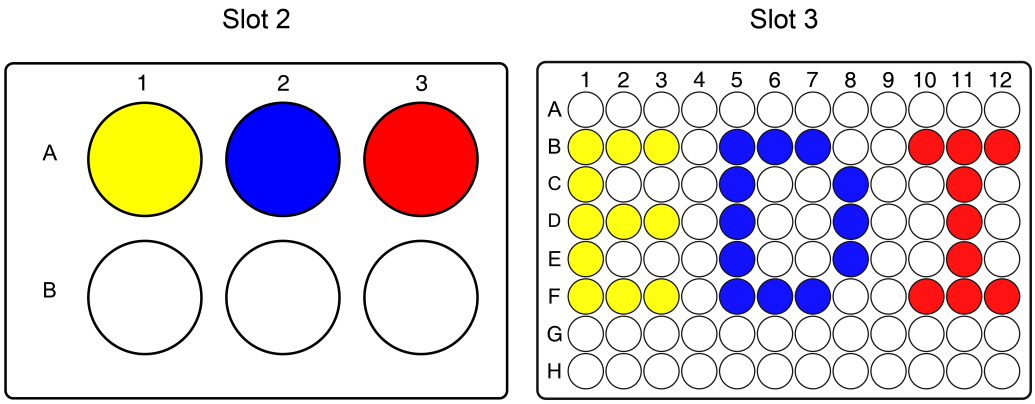
```
r_pipette = protocol.load_instrument(
    'p300_single',
    'right',
    tip_racks=[tip_rack]
)

letter_E = ['B1','B2','B3','C1','D1','D2','D3','E1','F1','F2','F3']
letter_D = ['B5','B6','B7','C5','C8','D5','D8','E5','E8','F5','F6','F7']
letter_I = ['B10','B11','B12','C11','D11','E11','F10','F11','F12']

letters = letter_E + letter_D + letter_I

for dest_well in letters:
    r_pipette.transfer(200,
                  tube_rack.wells_by_name()['A1'],
                  plate.wells_by_name()[dest_well]
                 )

# print(*protocol.commands(), sep = '\n')
```

```
C:\Users\User\.opentrons\deck_calibration.json not found. Loading defaults
C:\Users\User\.opentrons\robot_settings.json not found. Loading defaults
```

4 Create an "EDI" pattern on a 96-well plate (Level 2)

4.1 Exercise

This is similar to above. However, for level 2, each letter will have a different color, i.e. you have to keep pipetting colored liquid from one source to different destination wells.



We start with: - Slot 1: Opentrons 300 µL normal tips
- Slot 2: Opentrons 6 Tube Rack with Falcon 50 mL Conical

Well A1: 20 mL of yellow dye, for constructing the letter “E”
Well A2: 20 mL of blue dye, for constructing the letter “D”
Well A3: 20 mL of red dye, for constructing the letter “I”
- Slot 3: Corning 96 Well Plate 360 µL Flat

Our robot is equipped with:
Right mount: P300 single-channel pipette

4.2 Hint

To define a 1-to-1 relation between a source well and the destination well, a dictionary object will be useful.

4.3 Suggested solution

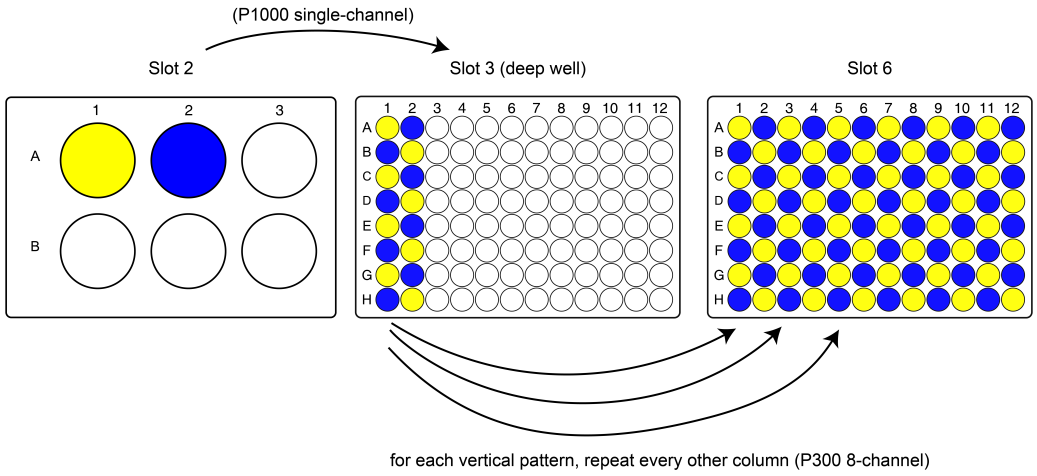
[5]:
```
from opentrons import simulate
protocol = simulate.get_protocol_api(version = '2.4')

tip_rack = protocol.load_labware('opentrons_96_tiprack_300ul', 1)
tube_rack = protocol.load_labware('opentrons_6_tuberack_falcon_50ml_conical', 2)
plate = protocol.load_labware('corning_96_wellplate_360ul_flat', 3)

r_pipette = protocol.load_instrument(
    'p300_single',
    'right',
    tip_racks=[tip_rack]
)

pattern_info = {
        'A1':['B1','B2','B3','C1','D1','D2','D3','E1','F1','F2','F3'],
        'A2':['B5','B6','B7','C5','C8','D5','D8','E5','E8','F5','F6','F7'],
        'A3':['B10','B11','B12','C11','D11','E11','F10','F11','F12']
        }

for source_well, dest_wells in pattern_info.items():
    for dest_well in dest_wells:
        r_pipette.transfer(200,
                    tube_rack.wells_by_name()[source_well],
                    plate.wells_by_name()[dest_well]
                   )

# print(*protocol.commands(), sep = '\n')
```

```
C:\Users\User\.opentrons\deck_calibration.json not found. Loading defaults
C:\Users\User\.opentrons\robot_settings.json not found. Loading defaults
```

5 Create a checkered pattern on a 96-well plate (Level 1)

5.1 Exercise

Let us perform the following task.



Task description:
Generate a checkered pattern on a 96-well flat bottom plate by pipetting dyes from the source tubes to an intermediate plate, then from the intermediate plate to the destination plate.

We start with: - Slot 1: Opentrons 96 Tip Rack 1000 µL - Slot 4: Opentrons 96 Tip Rack 300 µL - Slot 2: Opentrons 6 Tube Rack with Falcon 50 mL Conical
Well A1: 20 mL of yellow dye
Well A2: 20 mL of blue dye
- Slot 3: USA Scientific 96 Deep Well Plate 2.4 mL - Slot 6: Corning 96 Well Plate 360 µL Flat

Our robot is equipped with:
Left mount: P1000 single-channel pipette
Right mount: P300 8-channel pipette

5.2 Hints

1. When using an 8-channel pipette, it is helpful to use the `labware.columns_by_name()` function. It will help to read the Doc and test it out before attempting at the exercise.
2. Recall that for 8-channel pipettes, a groups of wells can be used as an argument for the `pipete.transfer()` function.

5.3 Suggested solution

[6]:
```
from opentrons import simulate
protocol = simulate.get_protocol_api(version = '2.4')

tip_rack_1000  = protocol.load_labware('opentrons_96_tiprack_1000ul', 1)
tip_rack_300 = protocol.load_labware('opentrons_96_tiprack_300ul', 4)
```

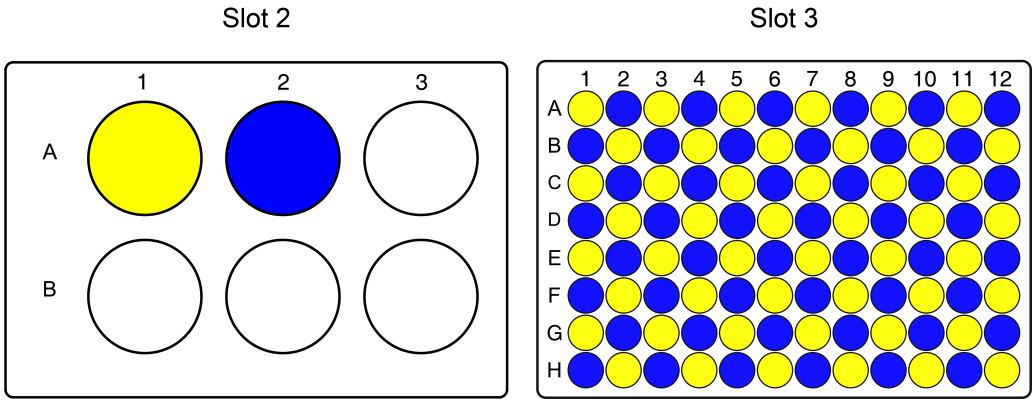
```
tube_rack = protocol.load_labware('opentrons_6_tuberack_falcon_50ml_conical', 2)
deep_well_plate = protocol.load_labware('usascientific_96_wellplate_2.
 ↪4ml_deep', 3)
plate = protocol.load_labware('corning_96_wellplate_360ul_flat', 6)

l_pipette = protocol.load_instrument(
    'p1000_single',
    'right',
    tip_racks=[tip_rack_1000]
)

r_pipette = protocol.load_instrument(
    'p300_multi',
    'left',
    tip_racks=[tip_rack_300]
)

# Step 1: deep-well plate

deep_well_pattern_info = {
    'A1':  ['A1', 'C1', 'E1', 'G1', 'B2', 'D2', 'F2', 'H2'],
    'A2':  ['B1', 'D1', 'F1', 'H1', 'A2', 'C2', 'E2', 'G2']
}
for source_well, dest_wells in deep_well_pattern_info.items():
    for dest_well in dest_wells:
            l_pipette.transfer(1500,
                     tube_rack.wells_by_name()[source_well],
                     deep_well_plate.wells_by_name()[dest_well]
                    )

# Step 2: final pattern
for col_name in ['1', '3', '5', '7', '9', '11']:
    r_pipette.transfer(200,
                 deep_well_plate.columns_by_name()['1'],
                 plate.columns_by_name()[col_name]
                )

for col_name in ['2', '4', '6', '8', '10', '12']:
    r_pipette.transfer(200,
                 deep_well_plate.columns_by_name()['2'],
                 plate.columns_by_name()[col_name]
                )

# print(*protocol.commands(), sep = '\n')
```

```
C:\Users\User\.opentrons\deck_calibration.json not found. Loading defaults
C:\Users\User\.opentrons\robot_settings.json not found. Loading defaults
```

6 Create a checkered pattern on a 96-well plate (Level 2)

6.1 Exercise

The outcome should be the same as Level 1. However, in this exercise we are doing this the slow and inefficient way - we will only use the P1000 single-channel pipette for all the pipetting actions.



Task description:
Generate a checkered pattern on a 96-well flat bottom plate by pipetting dyes from the source tubes to the destination plate.

We start with: - Slot 1: Opentrons 96 Tip Rack 1000 µL - Slot 2: Opentrons 6 Tube Rack with Falcon 50 mL Conical
Well A1: 20 mL of yellow dye
Well A2: 20 mL of blue dye
- Slot 3: Corning 96 Well Plate 360 µL Flat

Our robot is equipped with:
Left mount: P1000 single-channel pipette

6.2 Hints

1. Since we are using a P1000 pipette, we call dispense a volume of 200 µL multiple times from the same pipette. It is helpful to read the Docs for the `pipete.distribute()` function.
2. When using the `pipete.distribute()` function, we cannot use a for-loop to loop through the wells like what we previously did, we have to generate a list of well **objects**. Let's think about how to generate those well objects using for-loops.
3. When dealing with large amount of wells, sometimes it is easier to use well indices. It is not as simple in this case however. In fact, using well names is simplier in this case. We can use a nested for loop to generate a list of well names.

6.3 Assistance

Please check out the following function to generate a list of well names by inputting a list of rows and a list of columns. It will be helpful to include this in your script.

[7]:
```
def generate_wells_by_col_and_row(row_list, col_list):
    well_names = []
    for row in row_list:
        for col in col_list:
            well_names.append(row + str(col))
    return well_names

generate_wells_by_col_and_row( ['A', 'C', 'E', 'G'], [1, 3, 5, 7, 9, 11] )
```

[7]:
```
['A1',
 'A3',
 'A5',
 'A7',
 'A9',
 'A11',
 'C1',
 'C3',
 'C5',
 'C7',
 'C9',
 'C11',
 'E1',
 'E3',
 'E5',
 'E7',
 'E9',
 'E11',
 'G1',
 'G3',
 'G5',
 'G7',
 'G9',
 'G11']
```

6.4 Suggested solution

[8]:
```
from opentrons import simulate
protocol = simulate.get_protocol_api(version = '2.4')

tip_rack_1000  = protocol.load_labware('opentrons_96_tiprack_1000ul', 1)

tube_rack = protocol.load_labware('opentrons_6_tuberack_falcon_50ml_conical', 2)
plate = protocol.load_labware('corning_96_wellplate_360ul_flat', 3)
```

```
l_pipette = protocol.load_instrument(
    'p1000_single',
    'right',
    tip_racks=[tip_rack_1000]
)

def generate_wells_by_col_and_row(row_list, col_list):
    well_names = []
    for row in row_list:
        for col in col_list:
            well_names.append(row + str(col))
    return well_names

# Create the list of well names using the function
yellow_well_names = generate_wells_by_col_and_row( ['A', 'C', 'E', 'G'],
                                                   [1, 3, 5, 7, 9, 11] ) + \
                    generate_wells_by_col_and_row( ['B', 'D', 'F', 'H'],
                                                   [2, 4, 6, 8, 10, 12] )

# the backward slash sybmol means continue the code on the next line

# Simply invert the combination of rows and columns for blue wells
blue_well_names = generate_wells_by_col_and_row( ['B', 'D', 'F', 'H'],
                                                 [1, 3, 5, 7, 9, 11] ) + \
                    generate_wells_by_col_and_row(['A', 'C', 'E', 'G'],
                                                 [2, 4, 6, 8, 10, 12] )

# Note that we just created the list of well names, but we still need to␣
 ↪generate a list of well objects

# Create lists of wells using the lists of well names

yellow_wells = []
for well_name in yellow_well_names:
    yellow_wells.append(plate.wells_by_name()[well_name])

blue_wells = []
for well_name in blue_well_names:
    blue_wells.append(plate.wells_by_name()[well_name])

# Distribute dyes for yellow wells
l_pipette.distribute(200,
                     tube_rack.wells_by_name()['A1'],
                     yellow_wells
                    )
```

```
# Distribute dyes for blue wells
l_pipette.distribute(200,
                     tube_rack.wells_by_name()['A2'],
                     blue_wells
                    )

# print(*protocol.commands(), sep = '\n')
```

```
C:\Users\User\.opentrons\deck_calibration.json not found. Loading defaults
C:\Users\User\.opentrons\robot_settings.json not found. Loading defaults
```

[8]: <InstrumentContext: p1000_single_v1 in RIGHT>