Robotics Lab

Chand T. John
First Bytes & Code Longhorn
June 2019
1. Log into your account in the computer lab.
2. Click the 9 white dots at the bottom left corner of the screen.
3. Type `terminal` in the search box at the top.
4. Click on the icon labeled Terminal.
5. In the window that pops up, type 
/lusr/bin/pycharm and press Enter.
6. Wait for the app called PyCharm Community Edition to launch.

You might see a bunch of WARNINGs here. That's okay! Just ignore them.
7. Accept the license agreement if it asks you.
8. It doesn’t matter whether you choose to send usage statistics or not. Just pick a choice. I prefer not to send usage statistics.
9. Choose + Create New Project.
10. A window will appear that allows you to select a folder where you’d like to store files for the robotics program you’ll soon create.
11. Choose a folder name, for example, MyRobotics, and click the Create at the bottom right.
12. Click the **Close** button on the Tip of the Day pop-up window.
13. Go to the File menu and choose Settings...
14. Click the little triangle to the left of Project: MyRobotics (or whatever you named your project)
15. In the expanded menu, choose Project Interpreter.
16. Click the + sign at the top right.
17. Type `robopy` into the search box.
18. Select `robopy` in the list if it isn’t highlighted already.
19. Click Install Package.
20. A notification should pop up in the lower right once the robopy package has been installed.
21. Now type `numpy` into the search box.
22. Select `numpy` if it isn’t highlighted already.
23. Click Install Package.
24. A notification should pop up indicating that the numpy package was installed successfully.
25. Type \texttt{vtk} into the search box.
26. Select **vtk** if it isn’t highlighted already.
27. Click **Install Package**.
28. A notification should pop up indicating that the \texttt{vtk} package was installed successfully.
29. Close the Available Packages window.
30. Click OK to close the Settings window.
31. Right-click the name of your project under the Project panel.
32. Hover over “New” and choose “Python file”. A little window will appear.
33. Type `Main` and click OK.
34. Type this code into the Main.py window.

```python
import robopy.base.model as model
import numpy as np

def main():
    robot = model.Puma560()
    a = np.transpose(np.asmatrix(np.linspace(1, -180, 500)))
    b = np.transpose(np.asmatrix(np.linspace(1, 180, 500)))
    c = np.transpose(np.asmatrix(np.linspace(1, 90, 500)))
    d = np.transpose(np.asmatrix(np.linspace(1, 450, 500)))
    e = np.asmatrix(np.zeros((500, 1)))
    f = np.concatenate((d, b, e, c, d, axis=1))
    robot.animate(stances=f, frame_rate=30, unit='deg')

if __name__ == '__main__':
    main()
```
Make sure you typed in the program EXACTLY as shown. If the indentation isn’t the same, or if even a single punctuation mark is out of place or missing, the program probably won’t work!

Computers are really, really picky. They do EXACTLY what you say, and they don’t understand what you really meant for them to do.

If something doesn’t work in the next step, it’s probably because there is a typo in your program somewhere. Don’t worry, it happens to all of us. We’re humans, not robots. We humans make mistakes, but we’re also creative, and that’s all part of what makes humankind fascinating.
35. It’s time to run your first robotics program! In the Run menu, choose Run...
36. In the little window that pops up, choose Main and Run…
37. Watch the robot move as your program runs!
38. When the robot stops moving (or before that, if you’re in a hurry), close the window to stop the program.
39. Run your program again, but this time a faster way: click the Play button at the top right.

From now on, you can run your program this way, or by typing SHIFT+F10.
40. Click and drag the corners of the robot window to make it bigger.
Right-click and drag the mouse pointer inside the robot window to zoom in and out.
42. Click with the middle mouse button (or SHIFT+left-click) and drag the mouse pointer inside the robot window to slide the robot around.
43. Click and drag the mouse around inside the robot window to rotate the robot in 3D.
44. In the Project panel, click the little triangles to expand **venv**, then **lib**, then **python3.6**, then **site-packages**, then **robopy**, and finally, **base**. Double-click **model.py** to open it.
45. Examine the code in this file that lists the colors of the robot’s links.
46. There are lots of other colors we could use.

https://vtk.org/Wiki/VTK/Examples/Python/Visualization/NamedColorPatches
47. So, change the colors to ones you like!

For example:

```python
links = [Revolute(d=0, a=0, alpha=pi / 2, j=0, theta=0, offset=0, qlim=(-160 * pi / 180, 160 * pi / 180)),
         Revolute(d=0, a=0.4318, alpha=pi, j=0, theta=0, offset=0, qlim=(-45 * pi / 180, 225 * pi / 180)),
         Revolute(d=0.15005, a=0.0203, alpha=pi / 2, j=0, theta=0, offset=0,
                  qlim=(-225 * pi / 180, 45 * pi / 180)),
         Revolute(d=0.4318, a=0, alpha=pi / 2, j=0, theta=0, offset=0, qlim=(-110 * pi / 180, 170 * pi / 180)),
         Revolute(d=0, a=0, alpha=pi / 2, j=0, theta=0, offset=0, qlim=(-100 * pi / 180, 100 * pi / 180)),
         Revolute(d=0, a=0, alpha=0, j=0, theta=0, offset=0, qlim=(-226 * pi / 180, 226 * pi / 180))]

if base is None:
    base = tr trotx(-90, unit='deg')
else:
    assert isomorph(base, (4, 4))

file_names = SerialLink_setup_file_names()

colors = graphics.vtk_named_colors(['HotPink', 'Orchid', 'PaleVioletRed', 'LightPink', 'Magenta', 'DeepPink', 'White'])

super().__init__(links=links, base=base, name='puma_560', stl_files=file_names, colors=colors, param=param)
```

Again, remember to keep all the quotes and other punctuation as they were. If any punctuation is missing or out of place, the program might not run anymore! The computer just isn’t intelligent enough to know what you mean unless you type in everything just right!
48. Then run your program again...
(or SHIFT+F10)
49. And enjoy the fancy new colors of your robot!
50. You may now close model.py if you’d like.
51. Now, double-click serial_link.py to open it.
52. Scroll down to where it says def animate(...):
53. To get the program to start with a better view of the robot, add these lines of code:

```python
def animate(self, stances, unit='rad', frame_rate=25, gif=None):
    ""
    Animates SerialLink object over nx6 dimensional input matrix, with each row representing list of 6 joint angles.
    :param stances: nx6 dimensional input matrix.
    :param unit: unit of input angles. Allowed values: 'rad' or 'deg'
    :param frame_rate: frame_rate for animation. Could be any integer more than 1. Higher value runs through stances faster.
    :return: null
    ""
    if unit == 'deg':
        stances = stances * (pi / 180)

    self.pipeline = VtkPipeline(total_time_steps=stances.shape[0] - 1, gif_file=gif)
    self.pipeline.reader_list, self.pipeline.actor_list, self.pipeline.mapper_list = self._setup_pipeline_objs()
    self.pipeline.apply_stance(True, stances, apply_stance=True, actor_list=self.pipeline.actor_list)
    self.pipeline.add_actor(axisesCube(self.pipeline.ren))

    def execute(self, obj, event):
        nonlocal stances
        self.pipeline.timer_tick()
        self.pipeline.iren.AddObserver('TimerEvent', self.execute)

    render = self.pipeline.iren.FindPokedRenderer(0, 0)
    self.pipeline.iren.FlyTo(render, 0, -1, 6.9)
    self.pipeline.iren.GetRenderWindow().SetSize(800, 800)
    camera = self.pipeline.iren.GetActiveCamera()
    transform = camera.GetModelViewTransformObject()
    transform.Identity()
    transform.RotateX(10)
    transform.Translate(0.19981, -1.53877e-13, 3.46839)
    camera.ApplyTransform(transform)
    self.pipeline.iren.Animate()
```
54. Run the program again.
55. Watch the robot move from the new viewpoint.

TIP: Right-click and drag the mouse inside this window to get a good close-up view of the robot.
Now, let’s talk a bit about the robot’s structure.
The robot’s links

This robot has several links (parts that can move separately from each other).

In this robot, each link has a unique color.

Each of these is a link.
The robot’s joints

A joint is what connects two links together.

The links rotate about the joints to which they’re connected.

This robot has six joints, labeled J1, J2, J3, J4, J5, and J6 in the diagram to the left.
The robot’s joint angles

We can describe the robot’s position by listing the angles by which each joint of the robot is rotated.

I know that doesn’t make much sense, but the next several slides will show you what I mean.
J2 at 0° rotation

J2 at +45° rotation
J3 at 0° rotation

J3 at +45° rotation
J4

J4 at 0° rotation

J4 at +45° rotation
J5 at 0° rotation

J5 at 45° rotation
J6 at 0° rotation

J6 at +45° rotation
Joint angles to describe the robot’s pose

We can describe the robot’s current pose using its joint angles.

For example, in the image to the left, all 6 joints are at 0° rotation. If we use the variable \( q_1 \) to represent the angle by which \( J_1 \) is rotated, \( q_2 \) to represent the angle by which \( J_2 \) is rotated, and so on, then we would say that in the image to the left, the robot’s pose is: 
\[ (q_1 = 0°, q_2 = 0°, q_3 = 0°, q_4 = 0°, q_5 = 0°, q_6 = 0°). \]
Joint angles to describe the robot’s pose

In this image, all joints are at $0^\circ$ rotation except for J1, which is at $+45^\circ$ rotation. The robot’s pose is: $(q_1 = +45^\circ, q_2 = 0^\circ, q_3 = 0^\circ, q_4 = 0^\circ, q_5 = 0^\circ, q_6 = 0^\circ)$. 
Joint angles to describe the robot’s pose

In this image, all joints are at $0^\circ$ rotation except for J2, which is at $+45^\circ$ rotation. The robot’s pose is: $(q_1 = 0^\circ, q_2 = +45^\circ, q_3 = 0^\circ, q_4 = 0^\circ, q_5 = 0^\circ, q_6 = 0^\circ)$. 
Making the robot move involves nothing more than telling it to follow a sequence of poses!

\[
(q_1 = 0^\circ, q_2 = 0^\circ, q_3 = 0^\circ, q_4 = 0^\circ, q_5 = 0^\circ, q_6 = 0^\circ)
\]

\[
(q_1 = 0.01^\circ, q_2 = 0.02^\circ, q_3 = 0^\circ, q_4 = 0^\circ, q_5 = 0^\circ, q_6 = 0.03^\circ)
\]

\[
(q_1 = 0.02^\circ, q_2 = 0.04^\circ, q_3 = 0^\circ, q_4 = 0^\circ, q_5 = 0^\circ, q_6 = 0.06^\circ)
\]

\[
(q_1 = 0.03^\circ, q_2 = 0.06^\circ, q_3 = 0^\circ, q_4 = 0^\circ, q_5 = 0^\circ, q_6 = 0.09^\circ)
\]

\[\ldots\]
In fact, that’s what your program does!
Your program creates sequences of joint angles:

Make a list, `a`, of 500 evenly spaced numbers between $+1^\circ$ and $-180^\circ$.

Make a list, `b`, of 500 evenly spaced numbers between $+1^\circ$ and $+180^\circ$.

Make a list, `c`, of 500 evenly spaced numbers between $+1^\circ$ and $+90^\circ$.

Make a list, `d`, of 500 evenly spaced numbers between $+1^\circ$ and $+450^\circ$.

Make a list, `e`, of 500 zeros ($0^\circ$).
Your program creates sequences of joint angles:

```python
import robopy.base.model as model
import numpy as np

def main:
    robot = model.Puma560()
    a = np.transpose(np.asmatrix(np.linspace(1, -180, 500)))
    b = np.transpose(np.asmatrix(np.linspace(1, 180, 500)))
    c = np.transpose(np.asmatrix(np.linspace(1, 90, 500)))
    d = np.transpose(np.asmatrix(np.linspace(1, 450, 500)))
    e = np.asmatrix(np.zeros((500, 1)))
    f = np.concatenate((d, b, a, e, c, d), axis=1)

    robot.animate(stances=f, frame_rate=30, unit='deg')
```

Carry the joint J1 through the angles in `d`. Carry J2 through the angles in `b`. Carry J3 through the angles in `a`, etc.
Your program creates sequences of joint angles:

Carry the joint J1 through the angles in $d$. Carry J2 through the angles in $b$. Carry J3 through the angles in $a$, etc.
56. Close the robot window if you haven’t already.
57. Change your program to make the robot stand completely still in its zero pose (where all joint angles are at 0° rotation).

Change this to:

```python
def main():
    robot = model.Puma560()
    a = np.transpose(np.asmatrix(np.linspace(1, -180, 500)))
    b = np.transpose(np.asmatrix(np.linspace(1, 180, 500)))
    c = np.transpose(np.asmatrix(np.linspace(1, 90, 500)))
    d = np.transpose(np.asmatrix(np.linspace(1, 450, 500)))
    e = np.asmatrix(np.zeros((500, 1)))
    f = np.concatenate((a, b, c, d, e), axis=1)
    robot.animate(stances=f, frame_rate=30, unit='deg')
```

Run the program to see if it worked!
Then close the robot window to stop the program.
58. Make a program that makes the robot wave “Hi.”

```python
zeros = np.asmatrix(np.zeros((500, 1)))
q1 = zeros
q2 = zeros
q3 = zeros
q4 = zeros
q5 = zeros
q6 = zeros
f = np.concatenate((q1, q2, q3, q4, q5, q6), axis=1)

Change this to:

```python
wave_right = np.asmatrix(np.linspace(-45, 45, 25))
wave_left = np.asmatrix(np.linspace(45, -45, 25))
wave = np.concatenate((wave_right, wave_left), axis=1)
wave_2x = np.concatenate((wave, wave), axis=1)
wave_4x = np.concatenate((wave_2x, wave_2x), axis=1)
q5 = np.transpose(np.concatenate((wave_4x, wave_4x, wave_2x), axis=1))
```

Run the program to see if it worked!
Then close the robot window to stop the program.
59. Make a program that makes the robot spin its gripper around and around.

Run the program to see if it worked!
Then close the robot window to stop the program.

zeros = np.asmatrix(np.zeros((500, 1)))
q1 = zeros
q2 = zeros
q3 = zeros
q4 = zeros
q5 = zeros
q6 = zeros
f = np.concatenate((q1, q2, q3, q4, q5, q6), axis=1)

Change this to:

spin = np.asmatrix(np.linspace(0, 360, 25))
spin_twice = np.concatenate((spin, spin), axis=1)
spin_4x = np.concatenate((spin_twice, spin_twice), axis=1)
spin_8x = np.concatenate((spin_4x, spin_4x), axis=1)
q6 = np.transpose(np.concatenate((spin_8x, spin_8x, spin_4x), axis=1))
60. Make a program that makes the robot spin around, pointing in various directions. This is reminiscent of Oprah Winfrey’s famous quote, “You get a car! You get a car! Everybody gets a car!”

Change this to:

```python
zeros = np.asmatrix(np.zeros((500, 1)))
q1 = zeros
q2 = zeros
q3 = zeros
q4 = zeros
q5 = zeros
q6 = zeros
f = np.concatenate((q1, q2, q3, q4, q5, q6), axis=1)
```

```python
turn = np.asmatrix(np.linspace(0, 360, 200))
half_turn = np.asmatrix(np.linspace(0, 180, 100))
q1 = np.transpose(np.concatenate((turn, turn, half_turn), axis=1))
point_down = np.asmatrix(np.linspace(0, -90, 25))
point_up = np.asmatrix(np.linspace(-90, 0, 25))
point_cycle = np.concatenate((point_down, point_up), axis=1)
point_2x = np.concatenate((point_cycle, point_cycle), axis=1)
point_4x = np.concatenate((point_2x, point_2x), axis=1)
q3 = np.transpose(np.concatenate((point_4x, point_4x, point_2x), axis=1))
```

Run the program to see if it worked!
Then close the robot window to stop the program.
Try coming up with your own robot movements!

Can you make the robot dance?

Can you make it do a softball pitch?