# 1 Step 1: Arm geometry

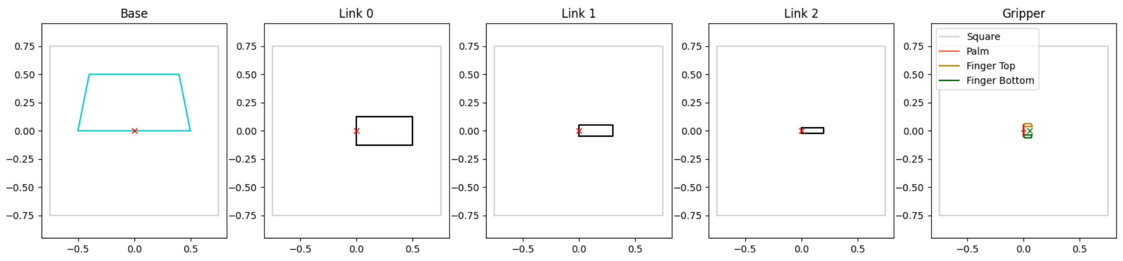
Build your arm pieces out of squares and wedges You will need to fill in get_pts_as_numpy_array in objects_in_world.py, if you haven't done so already.

The rest of the methods to fill in are in arm_forward_kinematics, the first section - set_transform_xxx

```
In [8]: # Syntax check of the methods used in this problem
        obj = read_object("Square") # method in objects_in_world
        base = afk.set_transform_base(obj)

        # Check this method in obj
        pts = get_pts_as_numpy_array(base)
        if pts.shape != (3, len(base["XYs"])):
            print("Should be a 3xn matrix")

        # Check this one works
        mat = mt.make_matrix_from_sequence(base["Matrix seq"])
        if mat.shape != (3, 3):
            print("Should be a 3x3 matrix")

        # The geometry methods - should take in an object, return an object, with Matrix seq st
        link = afk.set_transform_link(obj, 0.75, 0.5)

        palm = afk.set_transform_palm(obj, 0.5)

        finger = afk.set_transform_finger(obj, 0.5, (0.75, 0.1), True)
```

```
In [131]: base_size_param = (1.0, 0.5)
          link_sizes_param = [(0.5, 0.25), (0.3, 0.1), (0.2, 0.05)]
          palm_width_param = 0.1
          finger_size_param = (0.075, 0.025)

          # This function calls each of the set_transform_xxx functions, and puts the results
          # in a list (the gripper - the last element - is a list)
          arm_geometry = afk.create_arm_geometry(base_size_param, link_sizes_param, palm_width_param, f
          if len(arm_geometry) != 5:
              print("Wrong number of components, should be 5, got {len(arm_geometry)}")
          if len(arm_geometry[-1]) != 3:
              print("Wrong number of gripper components, should be 3, got {len(arm_geometry[-1])}")

          fig, axs = plt.subplots(1, len(arm_geometry), figsize=(4 * len(arm_geometry), 4))

          # Should be 5 windows, with the base, 3 links, and the gripper
          afk.plot_arm_components(axs, arm_geometry)
```

```
[{'type': 'scale', 'sx': 0.0125, 'sy': 0.0375}]
[{'type': 'scale', 'sx': 0.0125, 'sy': 0.0375}]
```

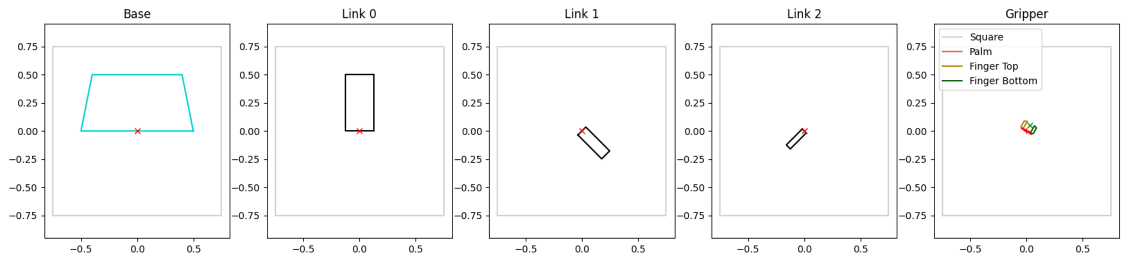

Base
Link 0
Link 1
Link 2
Gripper
Square
Palm
Finger Top
Finger Bottom
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.5
0.0
0.5

## 2 Step 2 - rotate links by angles

Now rotate all of the components by an angle in their own coorinate system

```
In [89]: # Syntax check for the methods used
         mat = afk.get_rotation_link(arm_geometry[1])
         if mat.shape != (3,3):
             print("Should return a 3x3 matrix")

         mat = afk.get_matrix_finger(arm_geometry[-1][1])
         if mat.shape != (3,3):
             print("Should return a 3x3 matrix")
```

```
In [107]: fig, axs = plt.subplots(1, len(arm_geometry), figsize=(4 * len(arm_geometry), 4))

          # Several different angles to check your results with
          angles_none = [0.0, 0.0, 0.0, [0.0, 0.0, 0.0]]
          angles_check_fingers = [np.pi/2, -np.pi/4, -3.0 * np.pi/4, [0.0, np.pi/4.0, -np.pi/4.0]]
          angles_check_wrist = [np.pi/2, -np.pi/4, -3.0 * np.pi/4, [np.pi/3.0, 0.0, 0.0]]
          angles_check = [np.pi/2, -np.pi/4, -3.0 * np.pi/4, [np.pi/3.0, np.pi/4.0, -np.pi/4.0]]

          # Setting the actual angles
          afk.set_angles_of_arm_geometry(arm_geometry, angles_check)

          # Draw the result - make sure to fix where this draws the fingers
          afk.plot_arm_components(axs, arm_geometry, b_with_angles=True)
```



## 3 Step 3 - matrices all the way down

Build all the matrices needed to do the forward kinematics

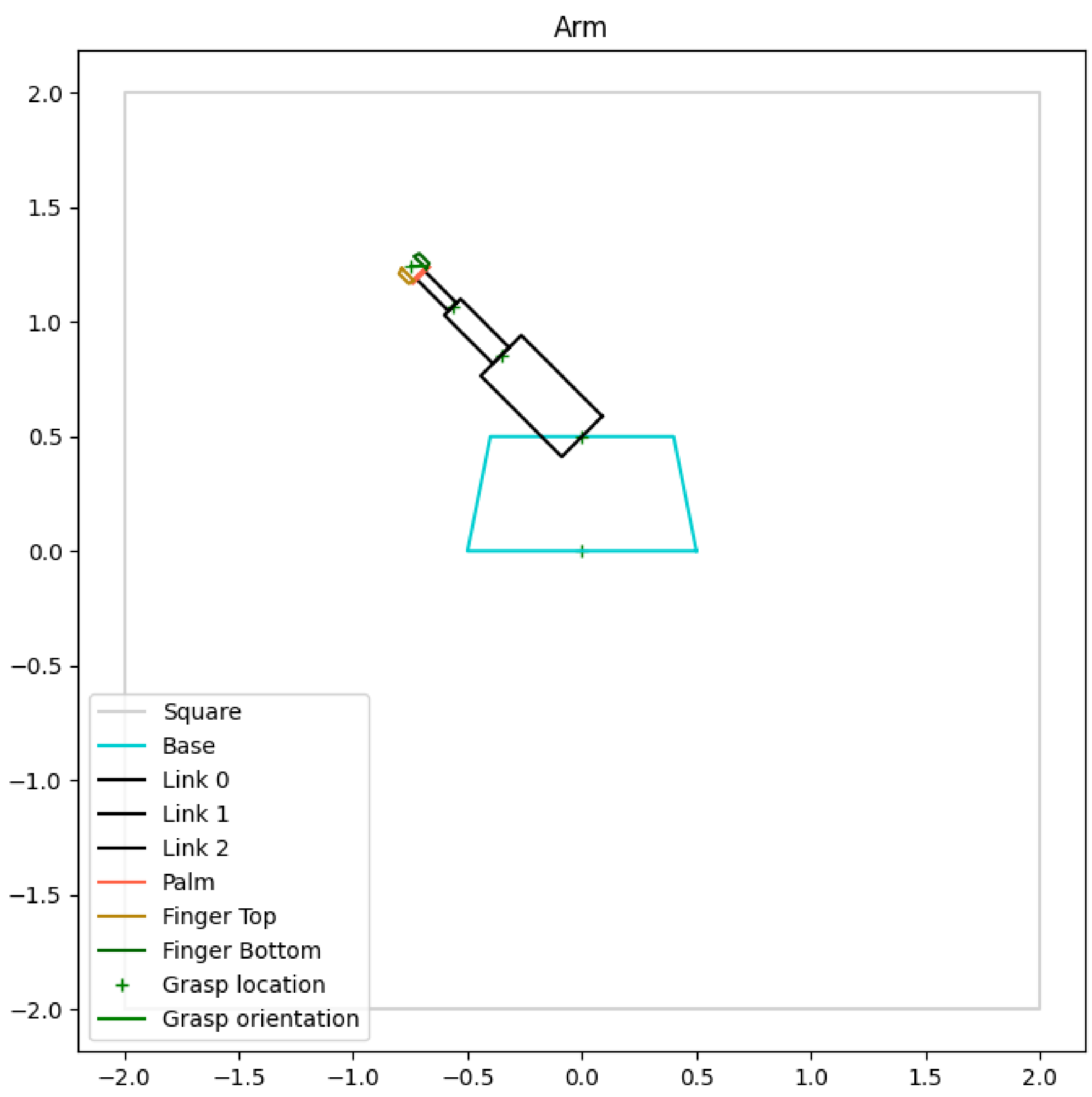
```
In [114]: # Syntax check of the methods used here
          mat = afk.get_matrix_base(arm_geometry[0])
          if mat.shape != (3,3):
              print("Should return a 3x3 matrix")

          mat = afk.get_matrix_link(arm_geometry[0])
          if mat.shape != (3,3):
              print("Should return a 3x3 matrix")

          matrices = afk.get_matrices_all_links(arm_geometry)
          if len(matrices) != len(arm_geometry):
              print("Should have one matrix for each piece of geometry")
```

```
1.5707963267948966
Arm link angle (Angle): 0.00 radians
1.5707963267948966
Arm link angle (Angle): 0.52 radians
Arm link angle (Angle): -0.79 radians
Arm link angle (Angle): 1.18 radians
```

```
In [134]: fig2, axs2 = plt.subplots(1, 1, figsize=(8, 8))

          # More angles to check
          angles_check_link_0 = [np.pi/4, 0.0, 0.0, [0.0, 0.0, 0.0]]
          angles_check_link_0_1 = [np.pi/4, -np.pi/4, 0.0, [0.0, 0.0, 0.0]]
          angles_gripper_check = [np.pi/6.0, -np.pi/4, 1.5 * np.pi/4, [np.pi/3.0, -np.pi/8.0, np.pi/6.0

          # Actually set the angles
          afk.set_angles_of_arm_geometry(arm_geometry, angles_check_link_0)
          # And get the matrices
          matrices = afk.get_matrices_all_links(arm_geometry)

          # Print the matrices - if you want
          np.set_printoptions(precision=2, suppress=True)
          for i, m in enumerate(matrices):
              print(f"Matrix {i} is\n{m}")

          # This is the part that mattters - draw the arm
          afk.plot_complete_arm(axs2, arm_geometry, matrices)
```

```
Matrix 0 is
[[1. 0. 0.]
```

```
 [0. 1. 0.]
 [0. 0. 1.]]
Matrix 1 is
[[ 0.  -1.   0. ]
 [ 1.   0.   0.5]
 [ 0.   0.   1. ]]
Matrix 2 is
[[-0.71 -0.71 -0.35]
 [ 0.71 -0.71  0.85]
 [ 0.    0.    1.  ]]
Matrix 3 is
[[-0.71 -0.71 -0.57]
 [ 0.71 -0.71  1.07]
 [ 0.    0.    1.  ]]
Matrix 4 is
[[-0.71 -0.71 -0.71]
 [ 0.71 -0.71  1.21]
 [ 0.    0.    1.  ]]
```


Arm
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Square
Base
Link 0
Link 1
Link 2
Palm
Finger Top
Finger Bottom
Grasp location
Grasp orientation