Topic	ROBOTIC ARM 2	
Class Description	Students will learn how to add a conveyor belt, attach a gripper hand with the UR5e robotic arm and control it using the keyboard.	
Class	PRO C289	
Class time	50 mins	
Goal	● Adding a conveyor belt in the factory environment. ● Attaching the Robotiq3fGripper hand with the UR5e robot. ● Controlling the gripper hand using keyboard controls.	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	Warm-Up Teacher -Led-Activity 1 Student-Led Activity 1 Wrap-Up	5 mins 20 mins 20 mins 5 mins
Credit & Permissions:	This project uses Webots, an open-source mobile robot simulation software developed by Cyberbotics Ltd. License	
WARM-UP SESSION - 10 mins		
Teacher Action	Student Action	

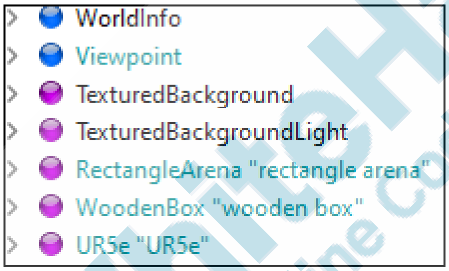
Hey <student's name>. How are you? It’s great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Activity Details **Following are the session deliverables:** ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
TEACHER-LED ACTIVITY - 15 mins	
Teacher Initiates Screen Share	
<u>ACTIVITY</u> ● **Adding a conveyor belt to the environment.** ● **Adding a solid node over the conveyor.** ● **Adding a physics node to the solid node.**	
Teacher Action	**Student Action**
Do you remember what we did in the last class? Great, if you have any doubts from the last class, please ask. *Note : Teacher will clear the doubts, if students have any.* Great, if you don’t have questions from the previous	**ESR :** Yes, we learned how to control the UR5e robotic arm using the keyboard.

classes, let's continue with today's class and attach a **gripper hand** with our robot so that it can actually **pick and drop** the objects.

But before that, let's add a **conveyor belt** to our **factory environment** so that we can move material from one place to another.

For that, let's first open the [teacher boilerplate link]() and download all the files from here.

Once you have opened the downloaded files in the webots software, let's click on the **add object** or the **+ sign button,** to add a new node.

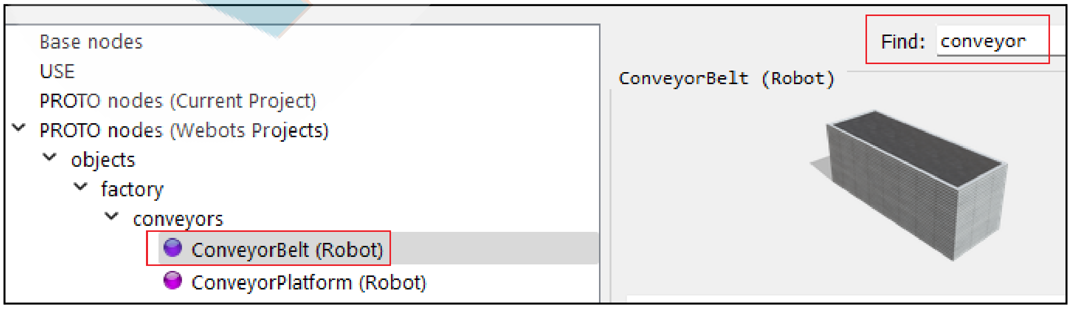


WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
WoodenBox "wooden box"
UR5e "UR5e"

A window will appear. To add the conveyor belt,

a) Search the word **'conveyor'** in the **Find** textbox.
b) Double click on the **conveyor belt (Robot)** node listed under **PROTO** nodes as,

PROTO nodes → objects → factory → conveyors → conveyor belt (Robot)

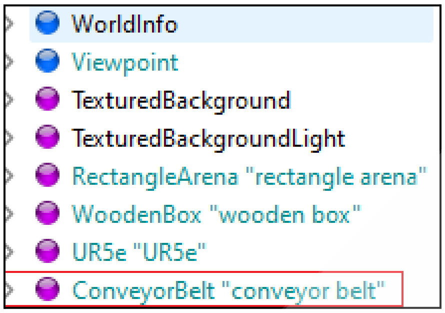


Base nodes
USE
PROTO nodes (Current Project)
PROTO nodes (Webots Projects)
objects
factory
conveyors
ConveyorBelt (Robot)
ConveyorPlatform (Robot)

Find: conveyor

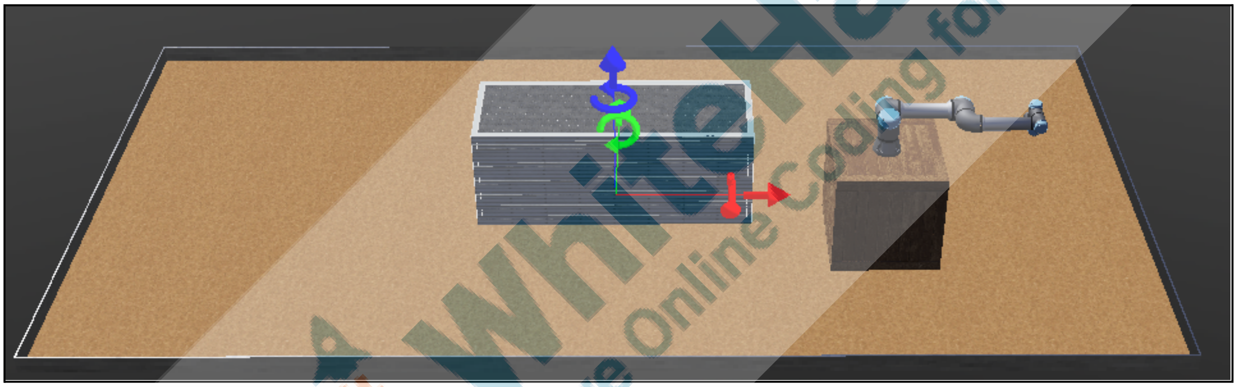
ConveyorBelt (Robot)

You will see a **conveyor belt node** added in the **scene tree**.

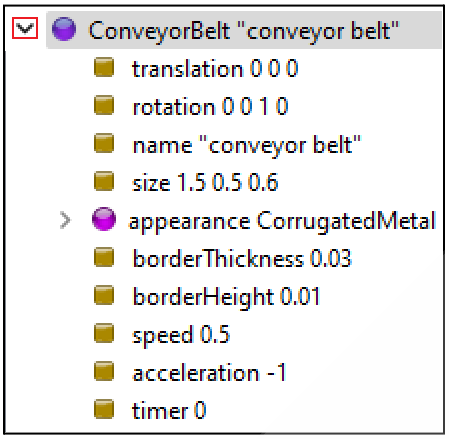


WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
WoodenBox "wooden box"
UR5e "UR5e"
ConveyorBelt "conveyor belt"

The **conveyor belt, wooden box and UR5e** arm robot arrangement will look as shown in the graphic.



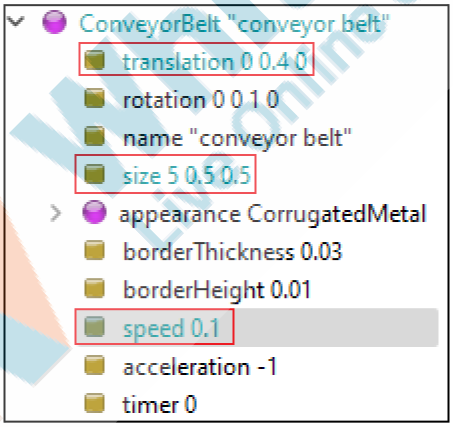
Clearly, we need to change certain properties of the conveyor belt. For that, let's first expand the **conveyor belt node** using the **'>' sign**.



```
ConveyorBelt "conveyor belt"
    translation 0 0 0
    rotation 0 0 1 0
    name "conveyor belt"
    size 1.5 0.5 0.6
  > appearance CorrugatedMetal
    borderThickness 0.03
    borderHeight 0.01
    speed 0.5
    acceleration -1
    timer 0
```

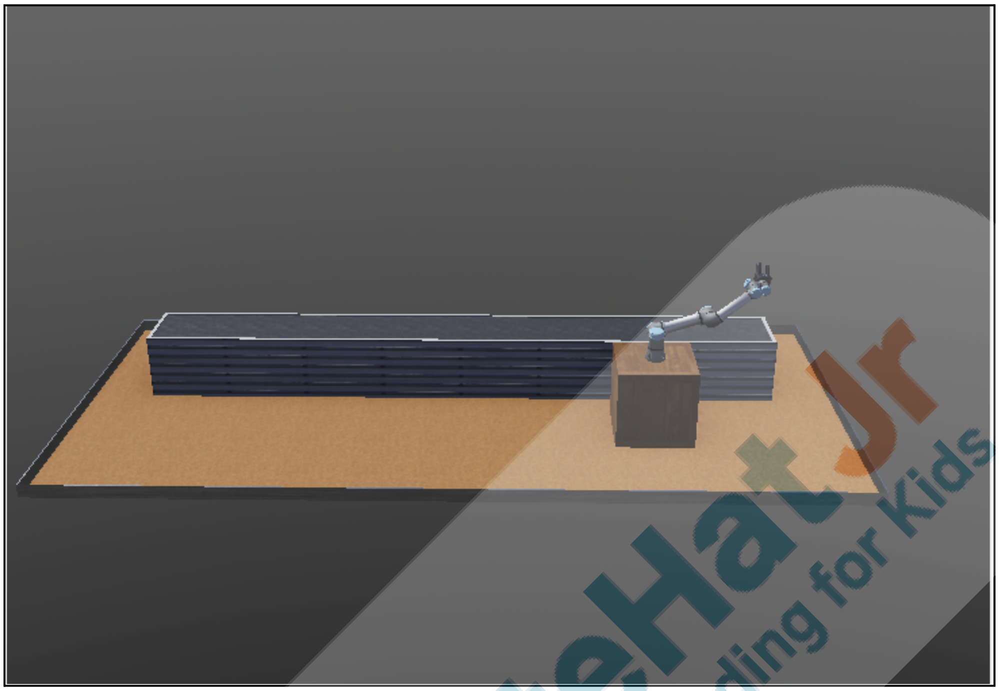
Do the following changes as,

a) Change the **translation** to **0** in **x direction**, **0.4** in **y direction** and **0** in **z direction**.
b) Change the **size** to **5 m** in **x direction, 0.5 m** in **y** and **z direction**.
c) Change the **speed** to **0.1 m/s**.



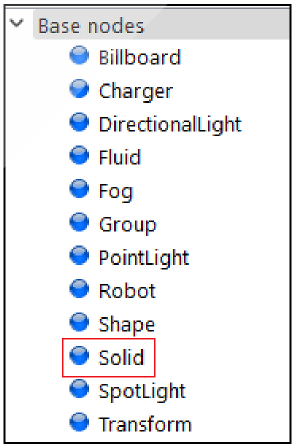
```
ConveyorBelt "conveyor belt"
    translation 0 0.4 0
    rotation 0 0 1 0
    name "conveyor belt"
    size 5 0.5 0.5
  > appearance CorrugatedMetal
    borderThickness 0.03
    borderHeight 0.01
    speed 0.1
    acceleration -1
    timer 0
```

The **conveyor belt, wooden box and UR5e** arm robot arrangement will look as shown in the graphic.

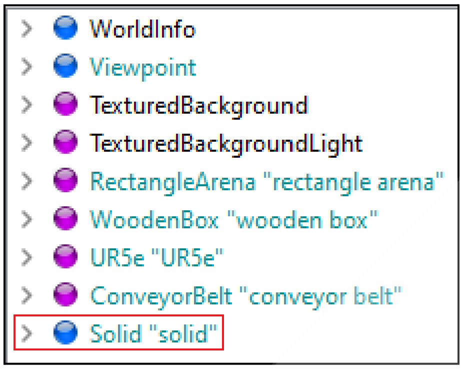
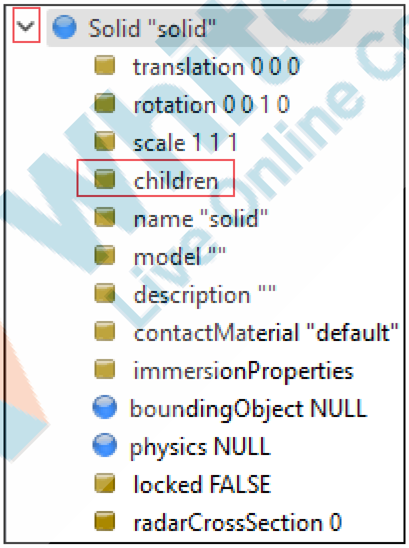


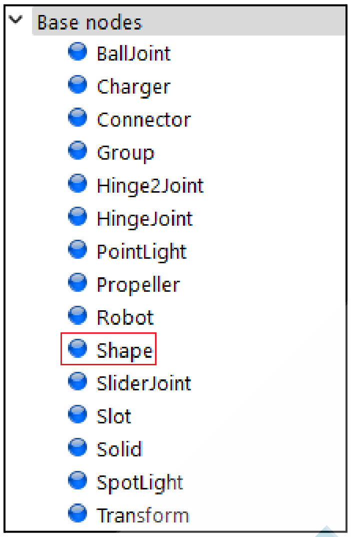
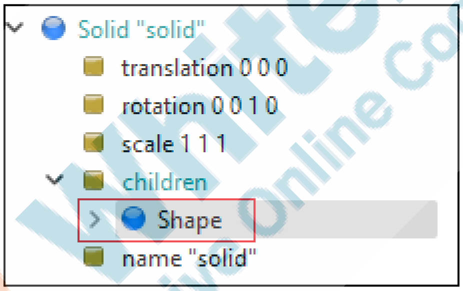
Once we have our conveyor belt, it’s time to keep some **solid objects** over it. For that,

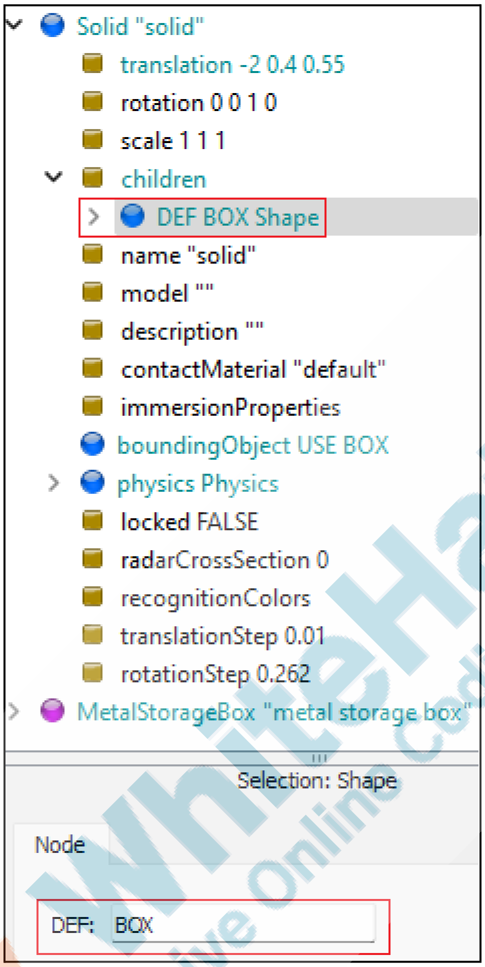
a) let’s click on the **add object** button or the button with the **+ sign**.
b) A window will open.
c) **Expand** the base nodes.
d) **Double click** on the **Solid** node.



Base nodes
- Billboard
- Charger
- DirectionalLight
- Fluid
- Fog
- Group
- PointLight
- Robot
- Shape
- Solid
- SpotLight
- Transform

You will see that a **Solid** node is added in the **scene tree**.	
 WorldInfo Viewpoint TexturedBackground TexturedBackgroundLight RectangleArena "rectangle arena" WoodenBox "wooden box" UR5e "UR5e" ConveyorBelt "conveyor belt" Solid "solid"	
Let's give a shape to our solid node. For that, a) Expand the solid node so that all its properties are visible. b) Double click on the children property.	
 Solid "solid" translation 0 0 0 rotation 0 0 1 0 scale 1 1 1 children name "solid" model "" description "" contactMaterial "default" immersionProperties boundingObject NULL physics NULL locked FALSE radarCrossSection 0	
A window will open. **Expand** the base nodes and **double click** on the **Shape** node.	

 Base nodes BallJoint Charger Connector Group Hinge2Joint HingeJoint PointLight Propeller Robot Shape SliderJoint Slot Solid SpotLight Transform	
You will see that a **Shape** node is added as a child of the **Solid** node.	
Solid "solid" translation 0 0 0 rotation 0 0 1 0 scale 1 1 1 children Shape name "solid"	
Next, let’s give a proper name to the **shape** node, so that it can be used later. For that, write the name **BOX**, in the **DEF** textbox and hit enter.	



Solid "solid"
translation -2 0.4 0.55
rotation 0 0 1 0
scale 1 1 1
children
DEF BOX Shape
name "solid"
model ""
description ""
contactMaterial "default"
immersionProperties
boundingObject USE BOX
physics Physics
locked FALSE
radarCrossSection 0
recognitionColors
translationStep 0.01
rotationStep 0.262
MetalStorageBox "metal storage box"

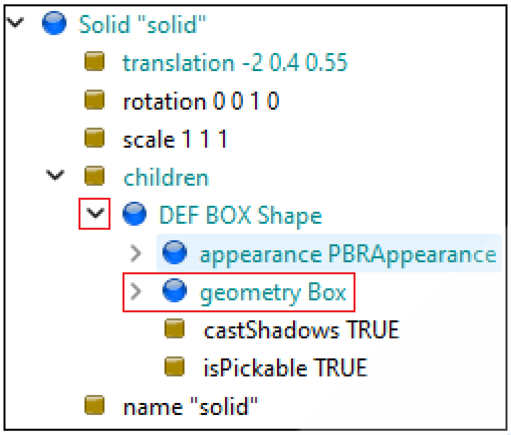
Selection: Shape

Node

DEF: BOX

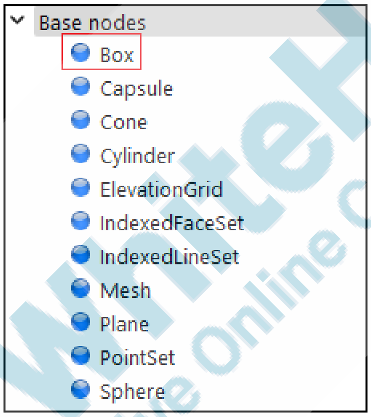
Next, let’s give our **solid** node a proper **shape**,

a) **Expand** the **shape** node.
b) **Double click** on the **geometry** property.



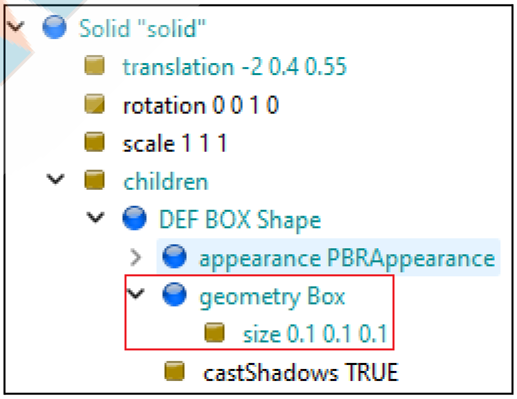
```
Solid "solid"
    translation -2 0.4 0.55
    rotation 0 0 1 0
    scale 1 1 1
    children
        DEF BOX Shape
            appearance PBRAppearance
            geometry Box
            castShadows TRUE
            isPickable TRUE
    name "solid"
```

A window will appear. **Double click** on the **box** shape.



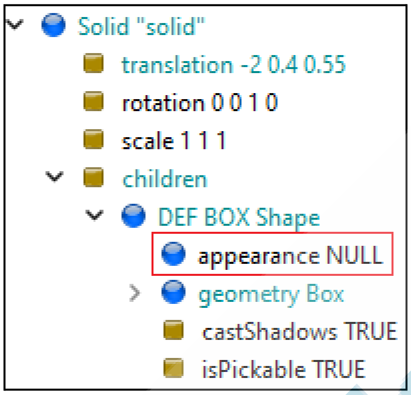
```
Base nodes
    Box
    Capsule
    Cone
    Cylinder
    ElevationGrid
    IndexedFaceSet
    IndexedLineSet
    Mesh
    Plane
    PointSet
    Sphere
```

You will see that a **box** node will be added. **Expand** it to change the size if you want. For now, **0.1 m** in **x, y** and **z direction** is **fine.**



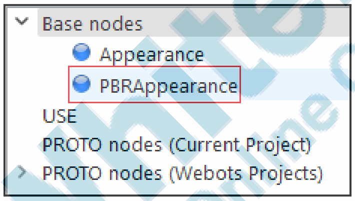
```
Solid "solid"
    translation -2 0.4 0.55
    rotation 0 0 1 0
    scale 1 1 1
    children
        DEF BOX Shape
            appearance PBRAppearance
            geometry Box
                size 0.1 0.1 0.1
            castShadows TRUE
```

Now once our box has a certain shape, it's time to give it an **appearance**. For that, **double click** on the **appearance** property.



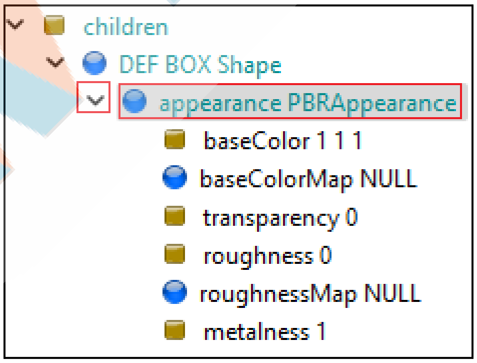
```
Solid "solid"
    translation -2 0.4 0.55
    rotation 0 0 1 0
    scale 1 1 1
    children
        DEF BOX Shape
            appearance NULL
            geometry Box
            castShadows TRUE
            isPickable TRUE
```

A window will appear. Expand **Base nodes** and select **PBR Appearance**.



```
Base nodes
    Appearance
    PBRAppearance
USE
PROTO nodes (Current Project)
PROTO nodes (Webots Projects)
```

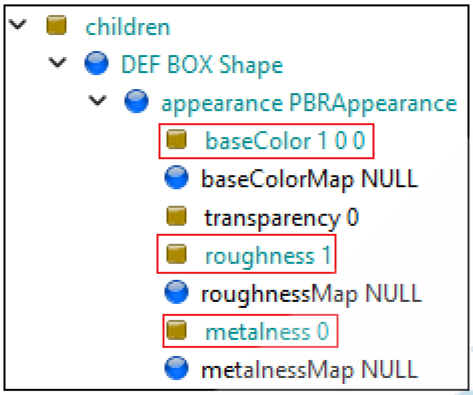
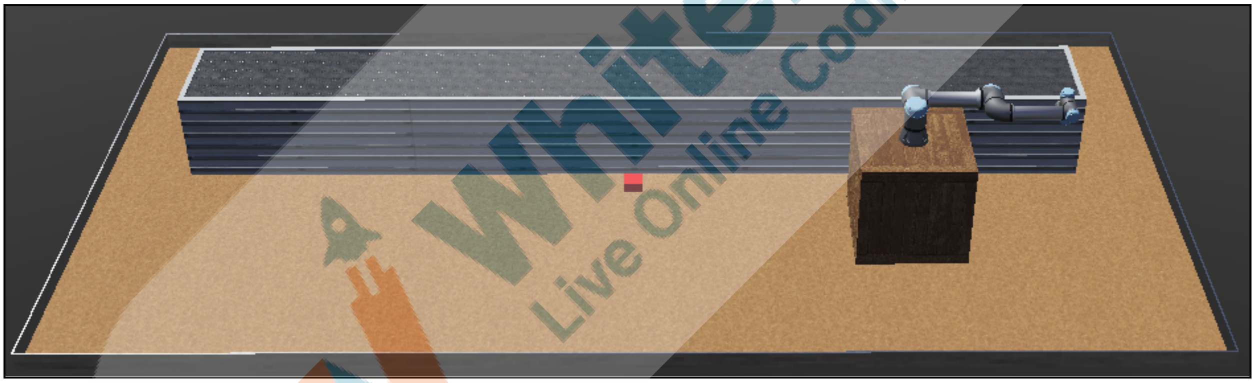
PBR Appearance node will be added to the **appearance property** of the **shape** node. **Expand** it.

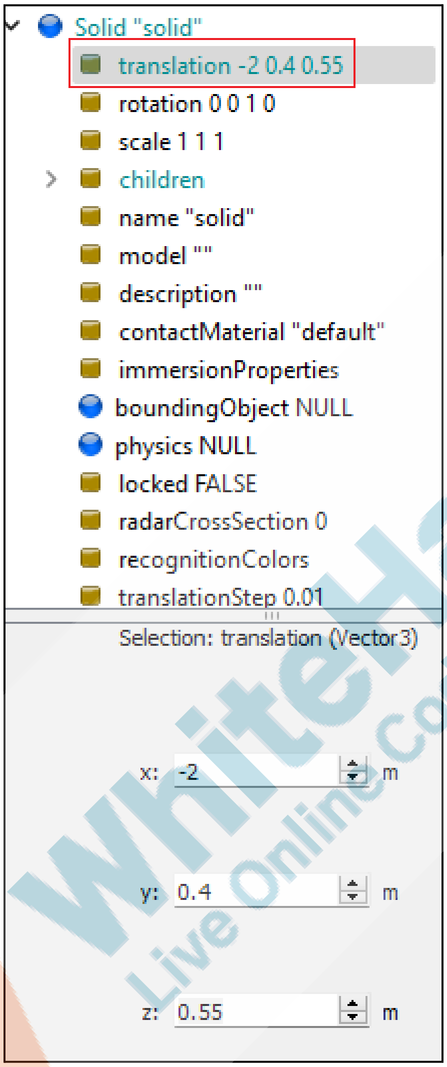


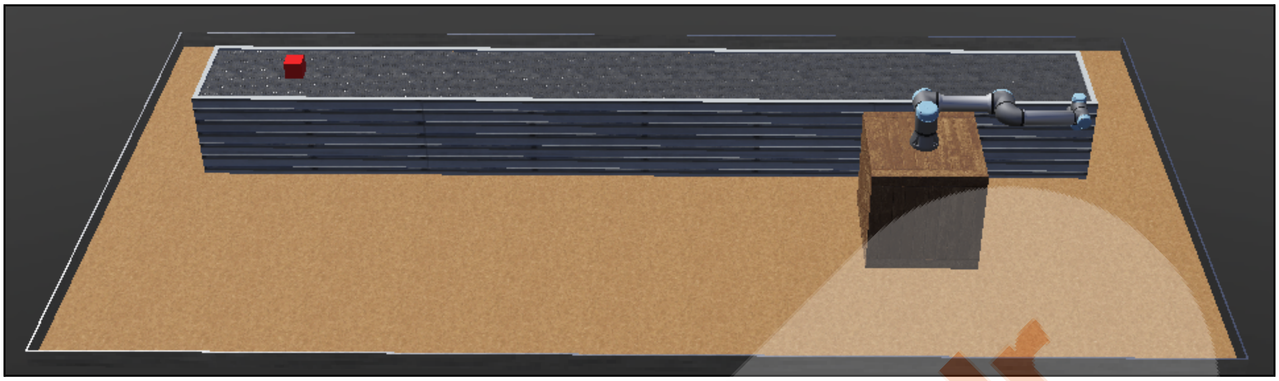
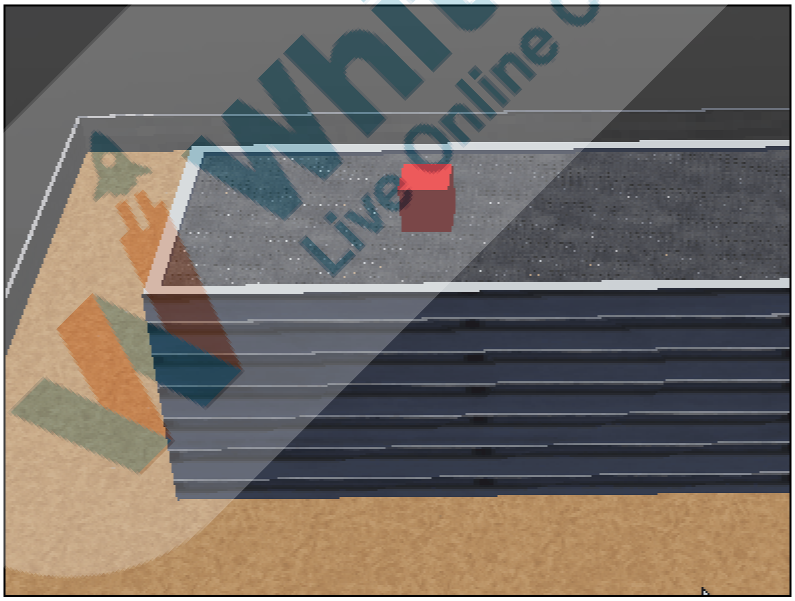
```
children
    DEF BOX Shape
        appearance PBRAppearance
            baseColor 1 1 1
            baseColorMap NULL
            transparency 0
            roughness 0
            roughnessMap NULL
            metalness 1
```

Let’s give it a **red** color. For that,

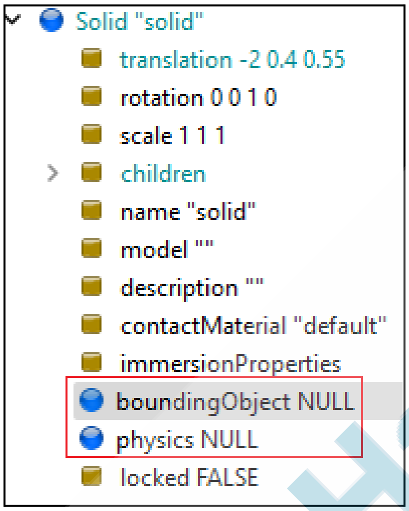
a) Change the **base color** to **1** in **red**, **0** in **green** and

0 in **blue**. b) Change the **roughness** to **1** and **metalness** to **0**.	
 children DEF BOX Shape appearance PBRAppearance baseColor 1 0 0 baseColorMap NULL transparency 0 roughness 1 roughnessMap NULL metalness 0 metalnessMap NULL	
Your environment will look as shown in the graphic below. Clearly we have to change the box position so that it is over the conveyor.	
	
To change the box position, change the **translation** property of the **solid** node to **-2** in **x** direction, **0.4** in **y** direction and **0.55** in **z** direction.	

 Solid "solid" translation -2 0.4 0.55 rotation 0 0 1 0 scale 1 1 1 children name "solid" model "" description "" contactMaterial "default" immersionProperties boundingObject NULL physics NULL locked FALSE radarCrossSection 0 recognitionColors translationStep 0.01 Selection: translation (Vector3) x: -2 m y: 0.4 m z: 0.55 m	
Your solid red box will appear over the conveyor.	

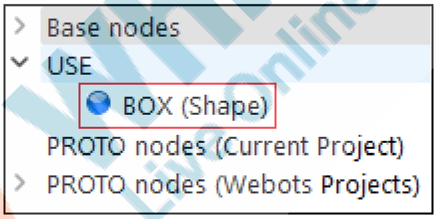
	
Save your environment and start the simulation. You will see that your conveyor is moving but your box isn't moving with the conveyor. Can you tell me the reason for this? If we don't define the **physics node** for any object, it would be considered as a **static body** and it won't **move**.	ESR : Varied
	
Before defining the **physics node** for the **solid body**, we have to define the **bounding object** for it. For that let's **expand** the solid node. You will see **bounding objects** and **physics properties** as **NULL**.	

To define the bounding object, **double click** on it.



Solid "solid"
- translation -2 0.4 0.55
- rotation 0 0 1 0
- scale 1 1 1
- children
- name "solid"
- model ""
- description ""
- contactMaterial "default"
- immersionProperties
- boundingObject NULL
- physics NULL
- locked FALSE

A window will appear. Expand the **USE** nodes and select the **BOX (shape).** A bounding box with the **same dimensions**, as of our box shape node, will be created.

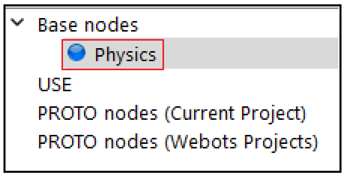


Base nodes
USE
- BOX (Shape)

PROTO nodes (Current Project)
PROTO nodes (Webots Projects)

To select the physics for the Solid node

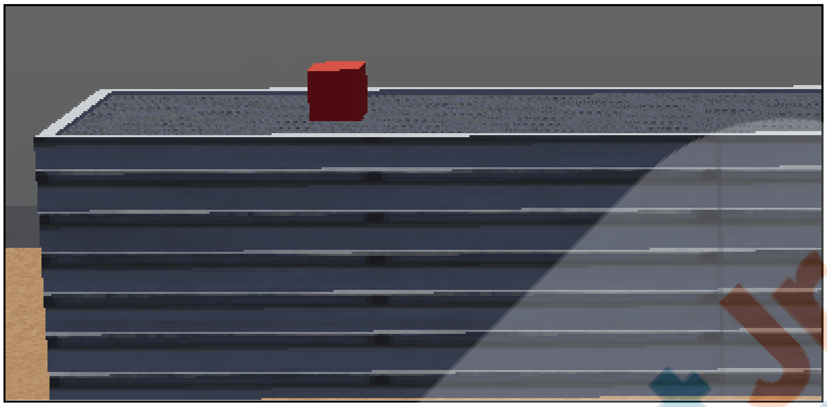
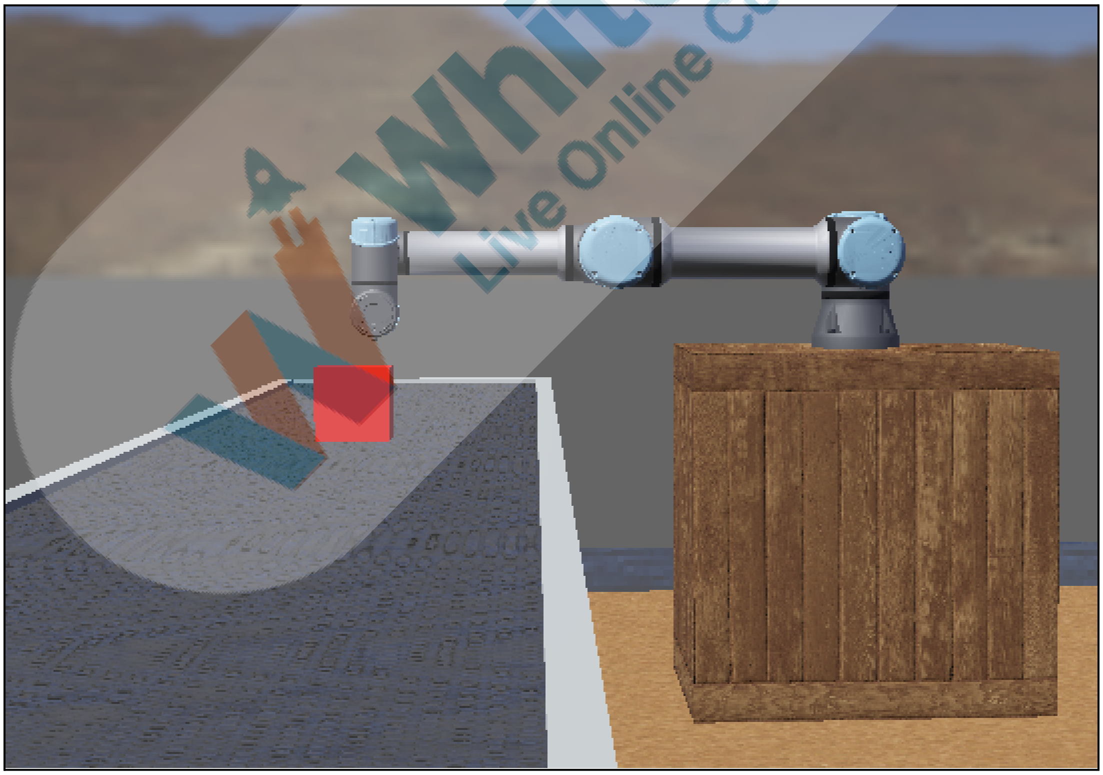
a) **Double click** on the **physics** property.
b) **Expand** the base nodes.
c) Select **Physics**.

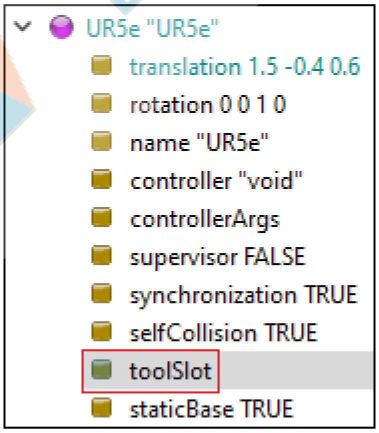


Base nodes
- Physics

USE
PROTO nodes (Current Project)
PROTO nodes (Webots Projects)

Save your environment and run the simulation again.

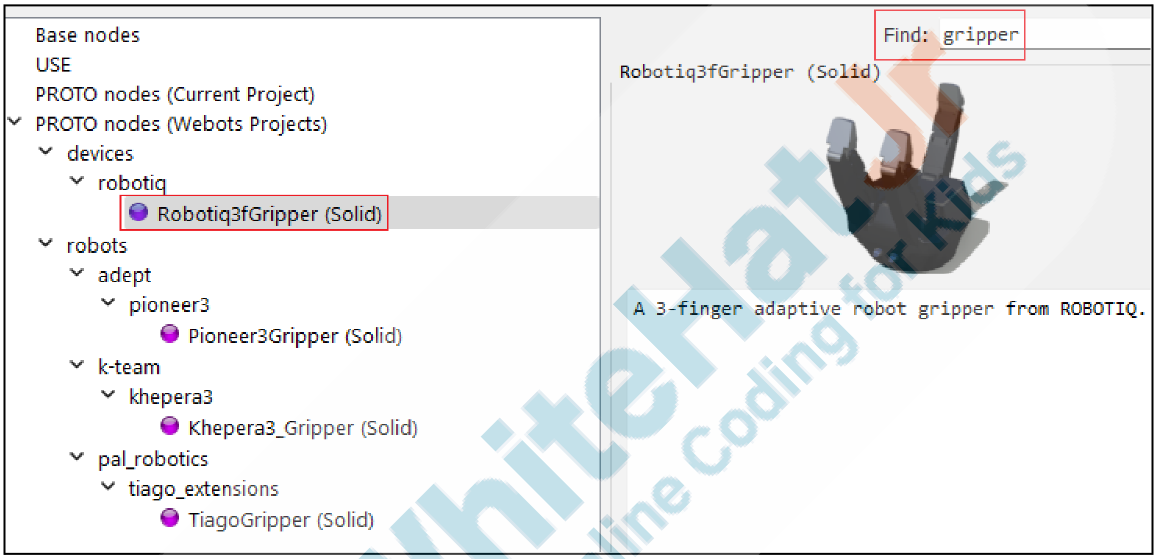
Great, the solid body is moving over the conveyor.	
	
Now there is one more challenge. Can you tell me what it is? That’s correct. We need a hand to pick and drop things.	ESR : The robotic arm doesn’t have a hand to pick up the solid object.
	

Teacher Stops Screen Share	
So now it’s your turn. Please share your screen with me.	
STUDENT-LED ACTIVITY 15 mins	
● **Ask the student to press the ESC key to come back to the panel.** ● **Guide the student to start Screen Share.** ● **The teacher gets into Full Screen.**	
Student Initiates Screen Share	
ACTIVITY ● **Attaching a gripper hand with the UR5e node.** ● **Controlling the hand and arm assembly using keyboard controls.**	
Teacher Action	**Student Action**
To attach a gripper hand with your robotic arm, let’s first download all the files from the student boilerplate link and open them in the webots software. Once the files are opened, **expand** the **UR5e node** and **double click** on the **tool slot** property.	
 UR5e "UR5e" translation 1.5 -0.4 0.6 rotation 0 0 1 0 name "UR5e" controller "void" controllerArgs supervisor FALSE synchronization TRUE selfCollision TRUE toolSlot staticBase TRUE	
A window will appear. To add the gripper hand,	

c) Search the word ‘**gripper**’ in the **Find** textbox.

d) Double click on the **Robotiq3fGripper (Solid)** node listed under **PROTO** nodes as,

PROTO nodes → devices → robotiq → Robotiq3fGripper (Solid)



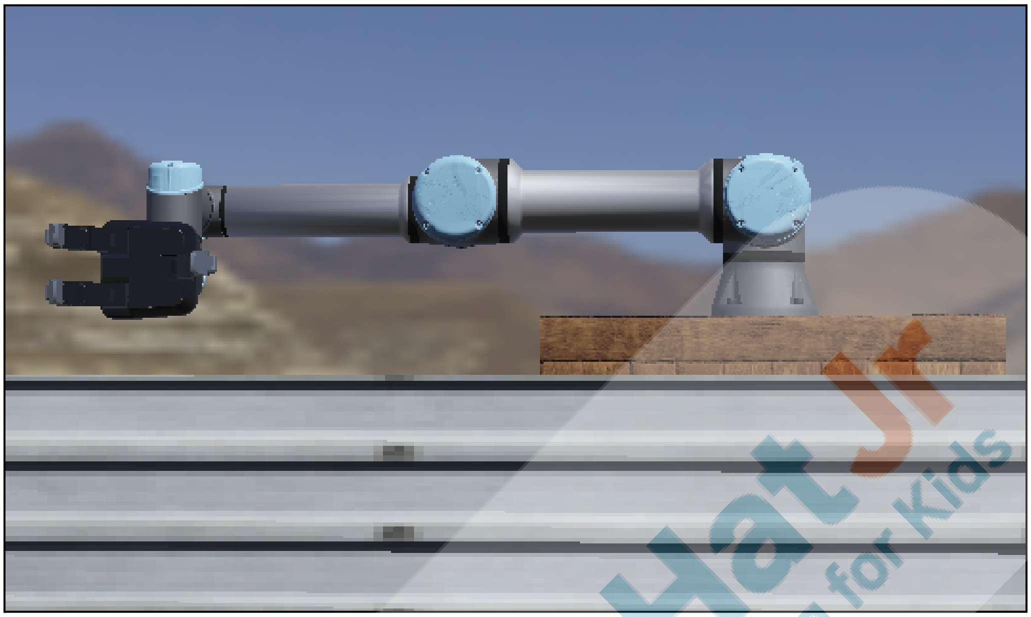
```
Base nodes
USE
PROTO nodes (Current Project)
PROTO nodes (Webots Projects)
  devices
    robotiq
        Robotiq3fGripper (Solid)
  robots
    adept
      pioneer3
          Pioneer3Gripper (Solid)
    k-team
      khepera3
          Khepera3_Gripper (Solid)
    pal_robotics
      tiago_extensions
          TiagoGripper (Solid)
```

Find: gripper

Robotiq3fGripper (Solid)

A 3-finger adaptive robot gripper from ROBOTIQ.

You will see that a **gripper** hand will get attached to the **UR5e robotic arm**.



Once the gripper hand is attached, we can control it using appropriate code. The **robotiq 3f gripper** hand has **3 fingers** and **11 joints** as,

a) **Finger 1 joints :**

- **palm_finger_1_joint**
- **finger_1_joint_1**
- **finger_1_joint_2**
- **finger_1_joint_3**

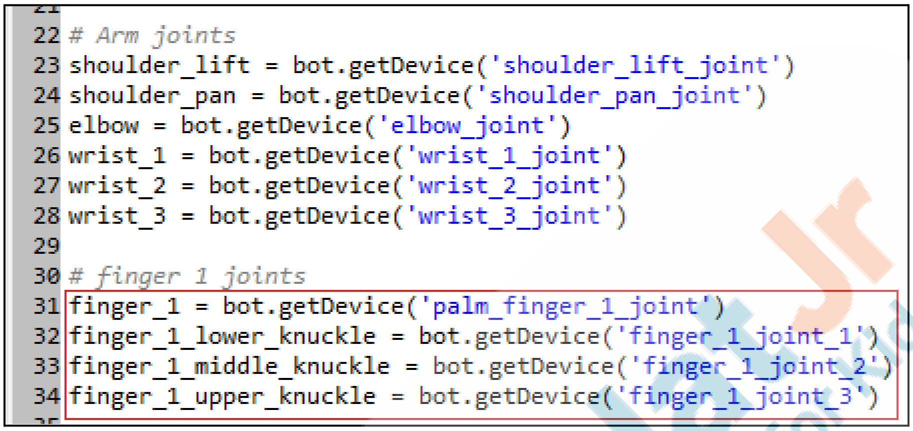
b) **Finger 2 joints :**

- **palm_finger_2_joint**
- **finger_2_joint_1**
- **finger_2_joint_2**
- **finger_2_joint_3**

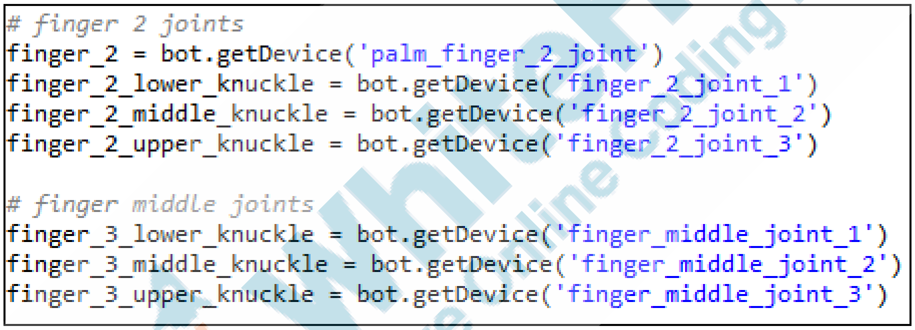
c) **Finger 3 joints :**

- **finger_middle_joint_1**
- **finger_middle_joint_2**
- **finger_middle_joint_3**

Let’s create **instances** for all the **11** joints, so that you can change their positions.



```
# Arm joints
shoulder_lift = bot.getDevice('shoulder_lift_joint')
shoulder_pan = bot.getDevice('shoulder_pan_joint')
elbow = bot.getDevice('elbow_joint')
wrist_1 = bot.getDevice('wrist_1_joint')
wrist_2 = bot.getDevice('wrist_2_joint')
wrist_3 = bot.getDevice('wrist_3_joint')

# finger 1 joints
finger_1 = bot.getDevice('palm_finger_1_joint')
finger_1_lower_knuckle = bot.getDevice('finger_1_joint_1')
finger_1_middle_knuckle = bot.getDevice('finger_1_joint_2')
finger_1_upper_knuckle = bot.getDevice('finger_1_joint_3')
```



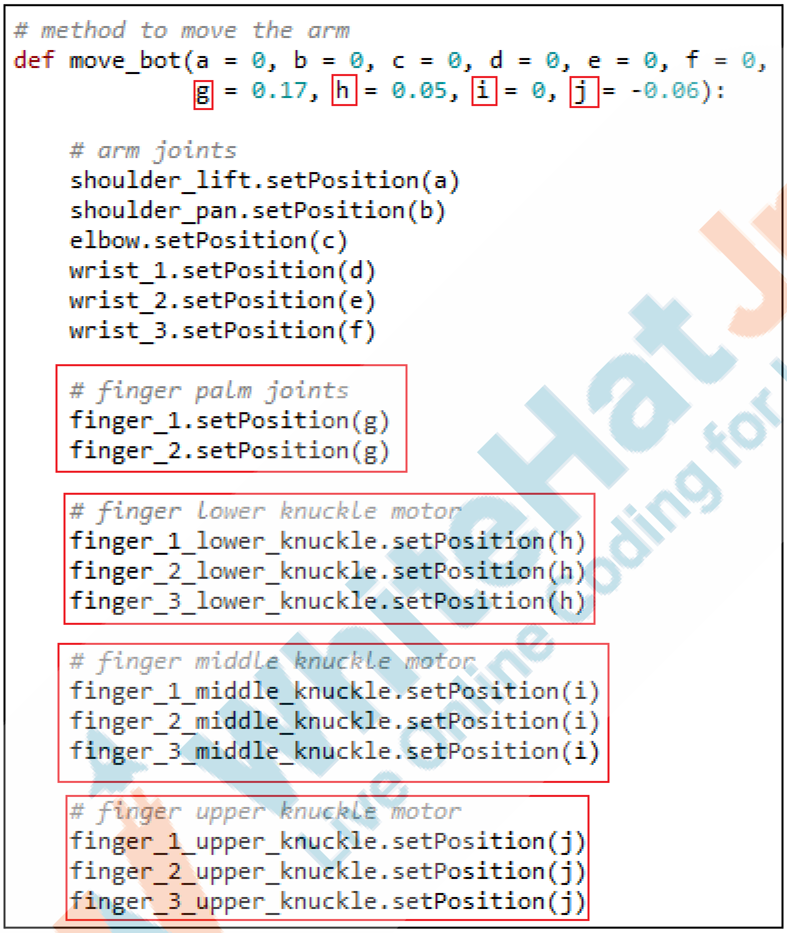
```
# finger 2 joints
finger_2 = bot.getDevice('palm_finger_2_joint')
finger_2_lower_knuckle = bot.getDevice('finger_2_joint_1')
finger_2_middle_knuckle = bot.getDevice('finger_2_joint_2')
finger_2_upper_knuckle = bot.getDevice('finger_2_joint_3')

# finger middle joints
finger_3_lower_knuckle = bot.getDevice('finger_middle_joint_1')
finger_3_middle_knuckle = bot.getDevice('finger_middle_joint_2')
finger_3_upper_knuckle = bot.getDevice('finger_middle_joint_3')
```

After that, let’s update the **move_bot()** method will allow us to change the **position** for each of the **arm and gripper joints** as,

a) This method will take **10 arguments**, so that we set the positions for the **6 arm joints** and **4 types of gripper joints**.
b) Set all the initial positions of all the arm joints to **0**.
c) Set the initial position of **finger-palm type joints** to **0.17**.
d) Set the initial position of the **lower knuckle type joints** to **0.05**.
e) Set the initial position of the **middle knuckle type joints** to **0**.
f) Set the initial position of the **upper knuckle type**

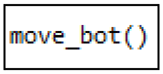
joints to **-0.06**.

g) Use **.setPosition()** method to set the position of the arm joints as, **object.setPosition(argument)**



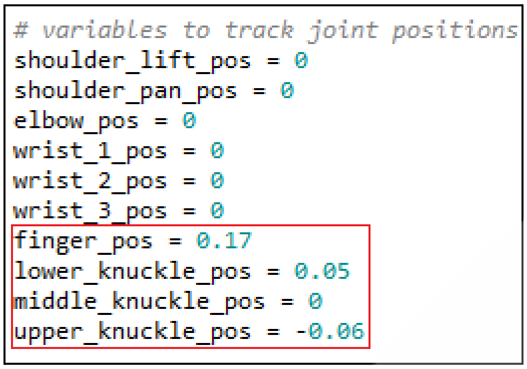
```
# method to move the arm
def move_bot(a = 0, b = 0, c = 0, d = 0, e = 0, f = 0,
             g = 0.17, h = 0.05, i = 0, j = -0.06):

    # arm joints
    shoulder_lift.setPosition(a)
    shoulder_pan.setPosition(b)
    elbow.setPosition(c)
    wrist_1.setPosition(d)
    wrist_2.setPosition(e)
    wrist_3.setPosition(f)

    # finger palm joints
    finger_1.setPosition(g)
    finger_2.setPosition(g)

    # finger lower knuckle motor
    finger_1_lower_knuckle.setPosition(h)
    finger_2_lower_knuckle.setPosition(h)
    finger_3_lower_knuckle.setPosition(h)

    # finger middle knuckle motor
    finger_1_middle_knuckle.setPosition(i)
    finger_2_middle_knuckle.setPosition(i)
    finger_3_middle_knuckle.setPosition(i)

    # finger upper knuckle motor
    finger_1_upper_knuckle.setPosition(j)
    finger_2_upper_knuckle.setPosition(j)
    finger_3_upper_knuckle.setPosition(j)
```

Call the **move_bot()** method, so that all the joints are at their default positions.



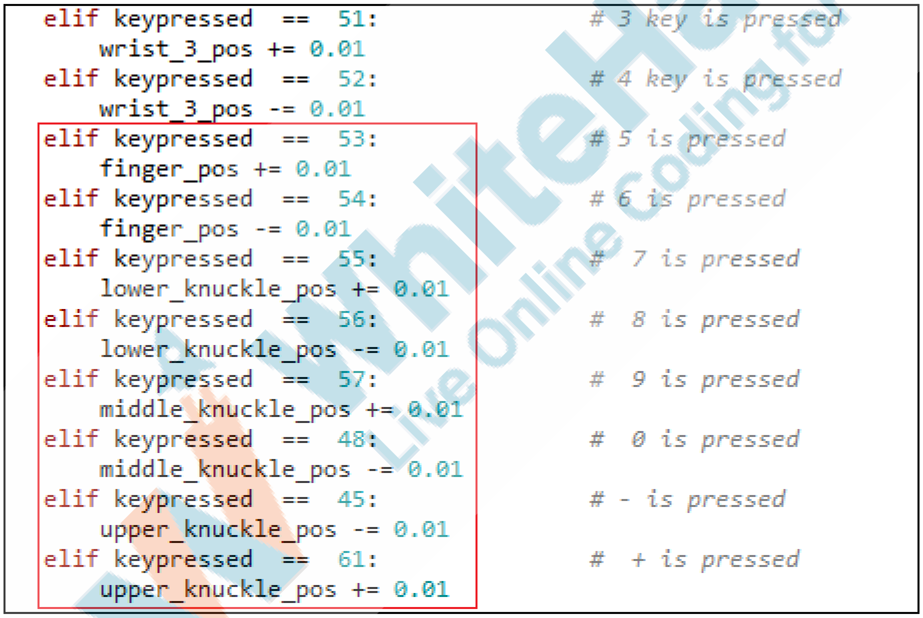
```
move_bot()
```

Let's add some new **variables** which will help us to track the **position** for the **gripper joints**.



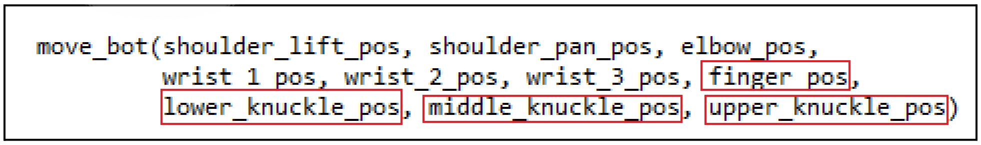
```
# variables to track joint positions
shoulder_lift_pos = 0
shoulder_pan_pos = 0
elbow_pos = 0
wrist_1_pos = 0
wrist_2_pos = 0
wrist_3_pos = 0
finger_pos = 0.17
lower_knuckle_pos = 0.05
middle_knuckle_pos = 0
upper_knuckle_pos = -0.06
```

Let’s add some more **conditional** statements, so that we can change the **variables** or the **position** of the **gripper joints**, whenever **keyboard keys** are **pressed**.



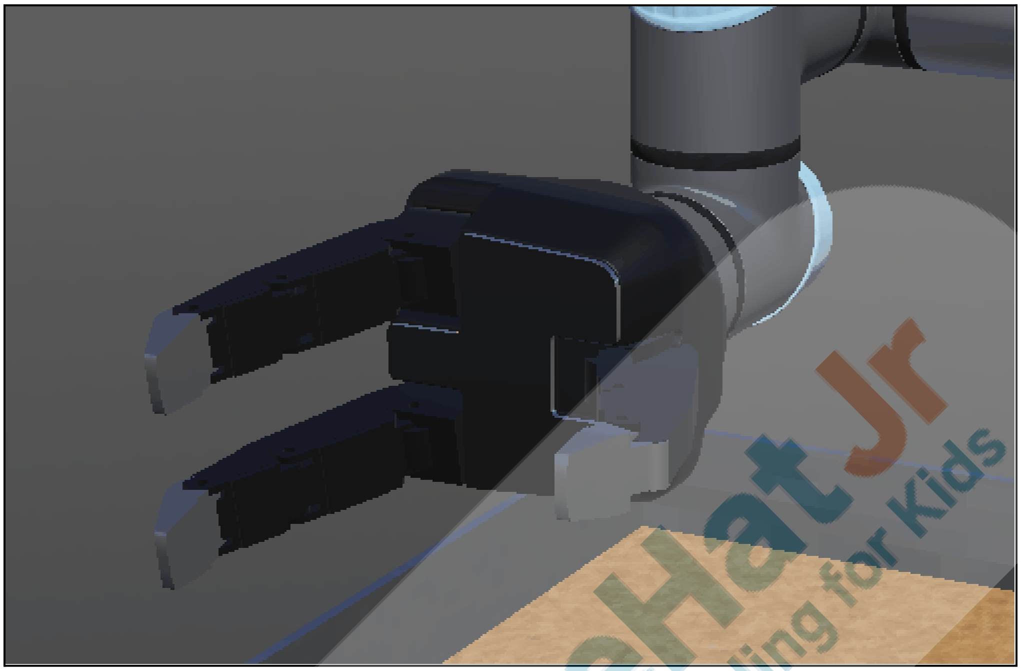
```
elif keypressed  ==  51:                      # 3 key is pressed
    wrist_3_pos += 0.01
elif keypressed  ==  52:                      # 4 key is pressed
    wrist_3_pos -= 0.01
elif keypressed  ==  53:                      # 5 is pressed
    finger_pos += 0.01
elif keypressed  ==  54:                      # 6 is pressed
    finger_pos -= 0.01
elif keypressed  ==  55:                      #  7 is pressed
    lower_knuckle_pos += 0.01
elif keypressed  ==  56:                      #  8 is pressed
    lower_knuckle_pos -= 0.01
elif keypressed  ==  57:                      #  9 is pressed
    middle_knuckle_pos += 0.01
elif keypressed  ==  48:                      #  0 is pressed
    middle_knuckle_pos -= 0.01
elif keypressed  ==  45:                      # - is pressed
    upper_knuckle_pos -= 0.01
elif keypressed  ==  61:                      #  + is pressed
    upper_knuckle_pos += 0.01
```

Finally let’s pass all these variables into the **move_bot()** method.

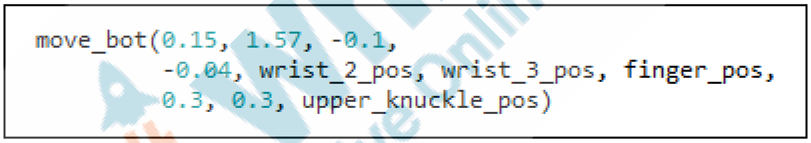


```
move_bot(shoulder_lift_pos, shoulder_pan_pos, elbow_pos,
         wrist_1_pos, wrist_2_pos, wrist_3_pos, finger_pos,
         lower_knuckle_pos, middle_knuckle_pos, upper_knuckle_pos)
```

Save your code and run the simulation.

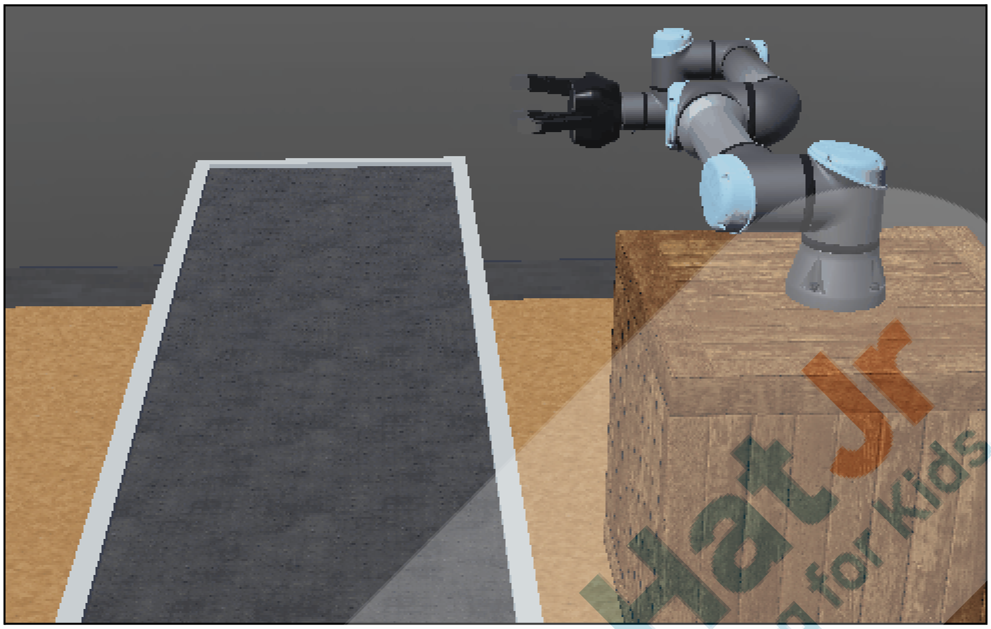


Let’s give specific values to the position variables, so that we can move our arm over the conveyor.



```
move_bot(0.15, 1.57, -0.1,
         -0.04, wrist_2_pos, wrist_3_pos, finger_pos,
         0.3, 0.3, upper_knuckle_pos)
```

Save the environment and run the simulation again.



Click here to view the reference video.

You will see that the gripper fingers are closed and it is not able to hold the box properly. Can you tell me the reason for this? Exactly. To solve this problem we will attach a distance sensor with our robot in the next class.	ESR : Yes, the arm doesn't know when the box will reach ,which means it doesn't know when to grip.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 mins

Activity details

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	
You get “hats-off” for your excellent work! In the next class, we will add a distance sensor to the gripper hand so that it can detect the objects that are in front of it.	

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks ✖ End Class

ACTIVITY LINKS

Activity Name	Description	Links
Teacher Activity 1	Teacher boilerplate	https://github.com/procodingclass/PRO-C289-Teacher-Boilerplate.git

Teacher Activity 2	Reference code	https://github.com/procodingclass/PRO-C289-Reference-Code.git
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/f41ae04f-c16f-400d-b7d0-e1d74b9495a6.pdf
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C289-Project-Solution
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/4754ddfa-cc00-450e-b372-9f1b7bf596da.pdf
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C289-Student-Boilerplate.git
Teacher reference 5	Final output gif	https://s3-whjr-curriculum-uploads.whjr.online/86e8d2fa-c600-470f-829a-98da74922e2a.gif