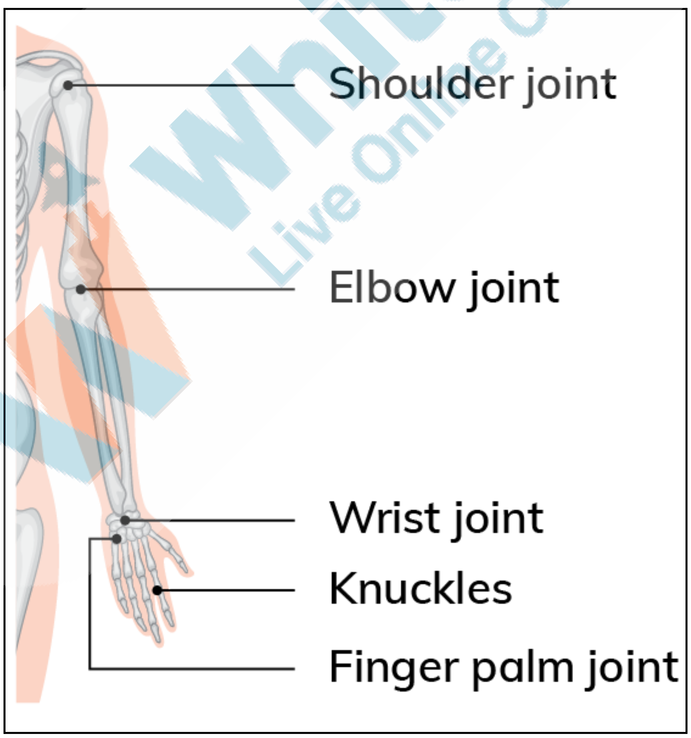
Topic	ROBOTIC ARM 1	
Class Description	Students will learn how to create a factory environment and then control a robotic arm using keyboard control.	
Class	PRO C288	
Class time	50 mins	
Goal	● Designing a factory environment. ● Importing ur5e bot. ● Controlling a robotic arm using keyboard controls.	
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	

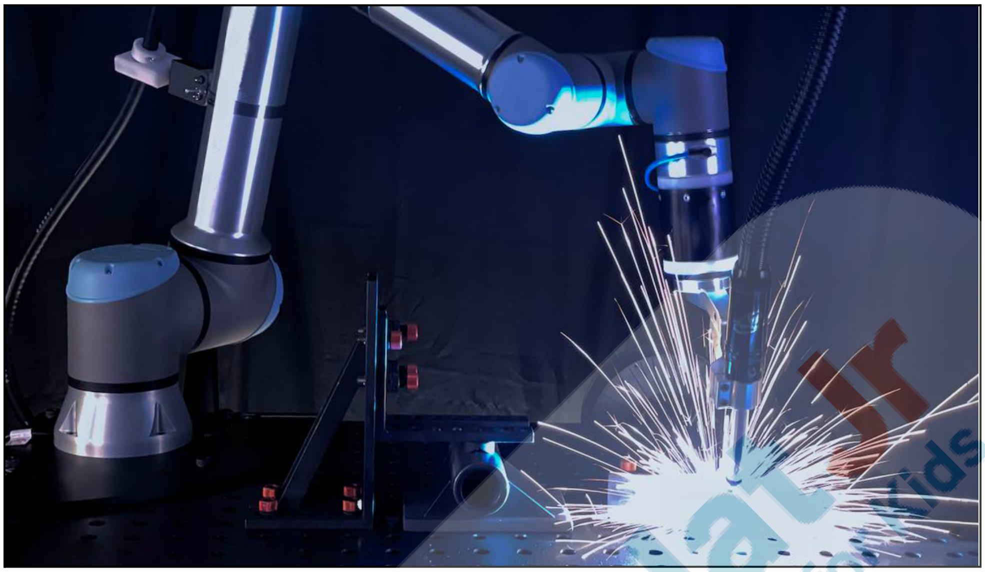
WARM-UP SESSION - 10 mins	
Teacher Action	Student Action

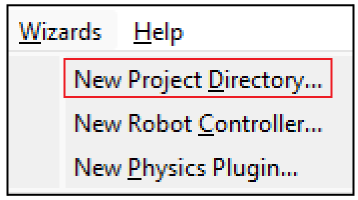
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show 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> ● **Designing the factory environment.** ● **Adding support structure for the robotic arm.**	
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.* Now that you don't have any questions from the previous classes, let's learn something new today.	**ESR :** Yes, we created a planet exploration robot.

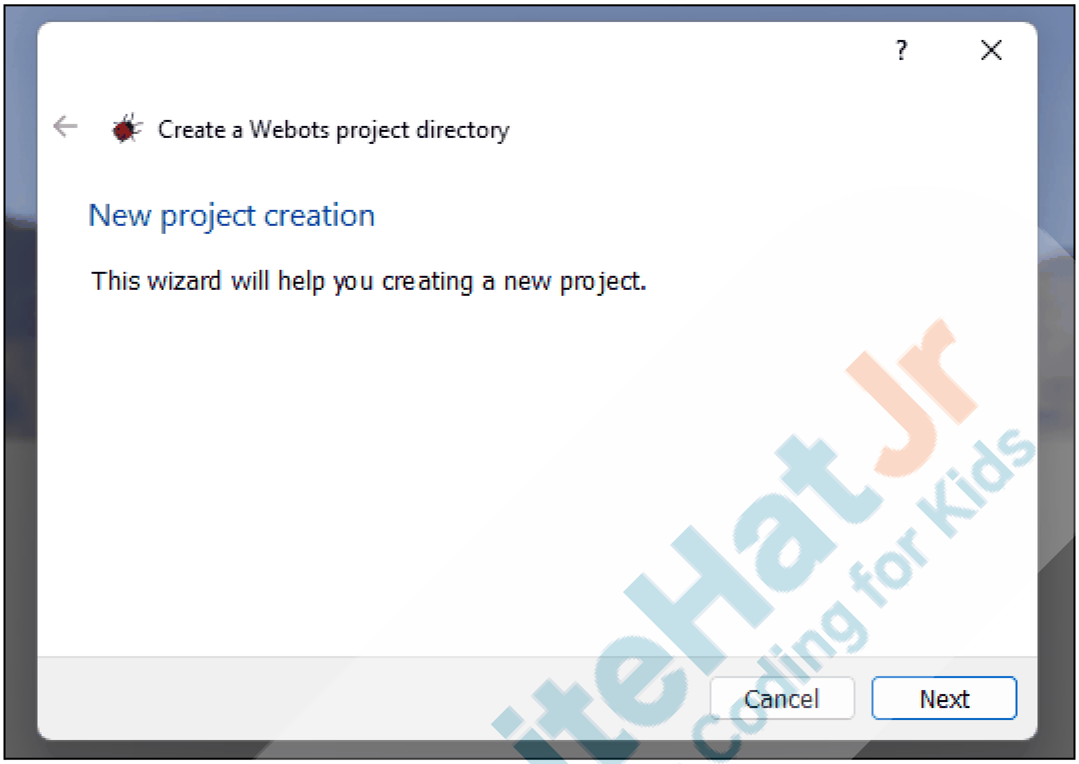
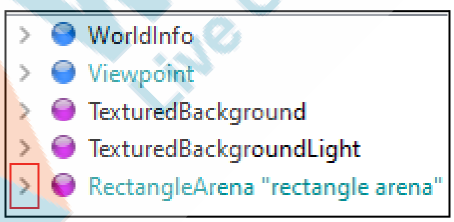
Let me give you a simple task. Can you pick up any object from your surroundings? It can be anything, a water bottle, charger etc. Great, can you tell me which organ of your body helped you to pick this object? Correct! Can you tell me what are the major organs or joints that make up your arm? Great, your arm is majorly made up of the following joints, a) **Shoulder joint.** b) **Elbow joint.** c) **Wrist joint.** d) **Finger palm joint.** e) **Knuckles.**	**ESR** : Student picks up anything from his surroundings. **ESR** : My arm. **ESR** : Varied

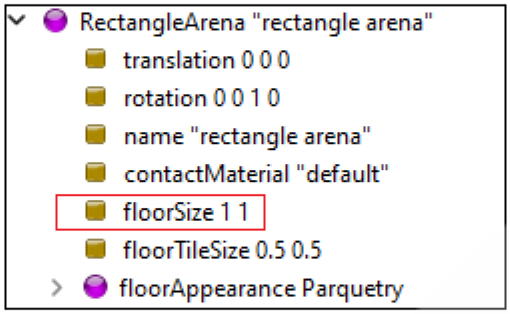


Now we know how that our arm helps us to **pick and drop** various objects in our day to day life, let me ask you one more question. Would you like to have an extra pair of arms? Yes, ofcourse, I mean who wouldn't. If you had 3 or maybe 4 arms, you would have picked up multiple things at once.	**ESR** : Varied
	
Inspired by the idea of an **'extra arm',** engineers designed one arm robots. These robots are extensively used in **industrial applications** to perform **repetitive tasks** such as **welding, material handling, drilling, painting etc.**	

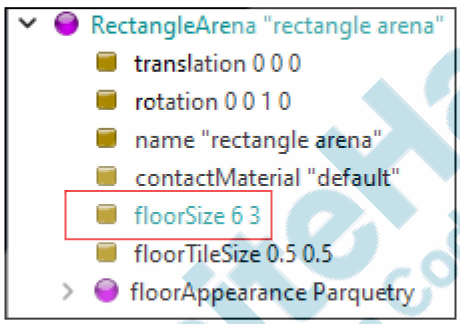
	
The robotic arm displayed in the above graphic is a **UR5e robotic arm**, created by **Universal robotics**. It is also available in the **webots** software. So let's learn how to use it and create an **OBJECT SORTING SYSTEM** using it, which will separate the **Rusty** or the **waste product** from the **Galvanized** or the **finished product.** Are you excited? To start with this project, open the **webots application** and **pause** the **last simulation** if it's already **running**. After that, from the top menu bar, click on the **Wizards** option.	**ESR** : Yes
	
A drop down menu will appear. Select the **New Project Directory** option.	

	
A dialog box will appear which will ask you the following questions, a) If you want to create a new project or not? ● Click on **Next**. b) It will ask you to either **choose** an **existing directory** or **create** a **new directory** so that you can **store** all the **assets** of your project in a single directory. ● Create a new directory and name it as **one arm robot.** c) Next it will ask to **give a name** and **choose the features** that you want in your project world. ● Write the name as **one arm robot.wbt** and check mark **Add a rectangular arena** into your project world. d) Finally, click on **finish.** Your new project world will be created.	

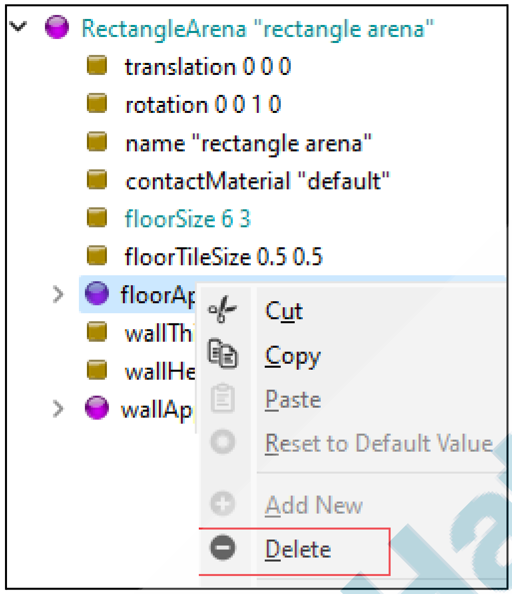
	
As soon as the project world is created, you will see the **rectangular arena** node by default, in the **scene tree.**	
	
Let's modify the **size** and **appearance** of the **rectangular arena**, so that it looks more like a **factory** floor. For that, first let's **expand** the **Rectangular arena** node, so that its properties are visible. You will see that the floor **size** is **1 m X 1m** by default.	



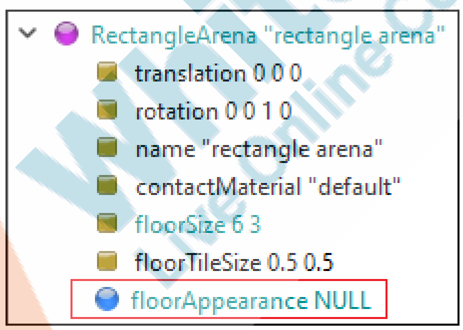
Let's change the floor size to **6 m** in **x direction** and **3 m** in **y direction**.



To **modify** the **appearance** of the floor, let's first delete the default appearance of it. For that, **right click** on the **floor appearance property** and then select **delete**.



Once you have deleted it, you will see the **floor Appearance property** as **Null**.

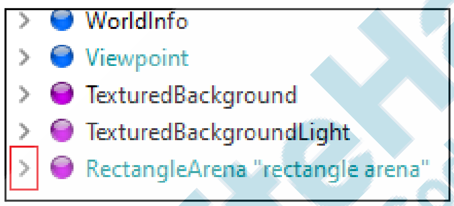


After that, let's **double click** on the **floor appearance property.** A window will appear which will ask you to add an appearance to the floor.

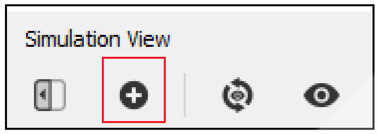
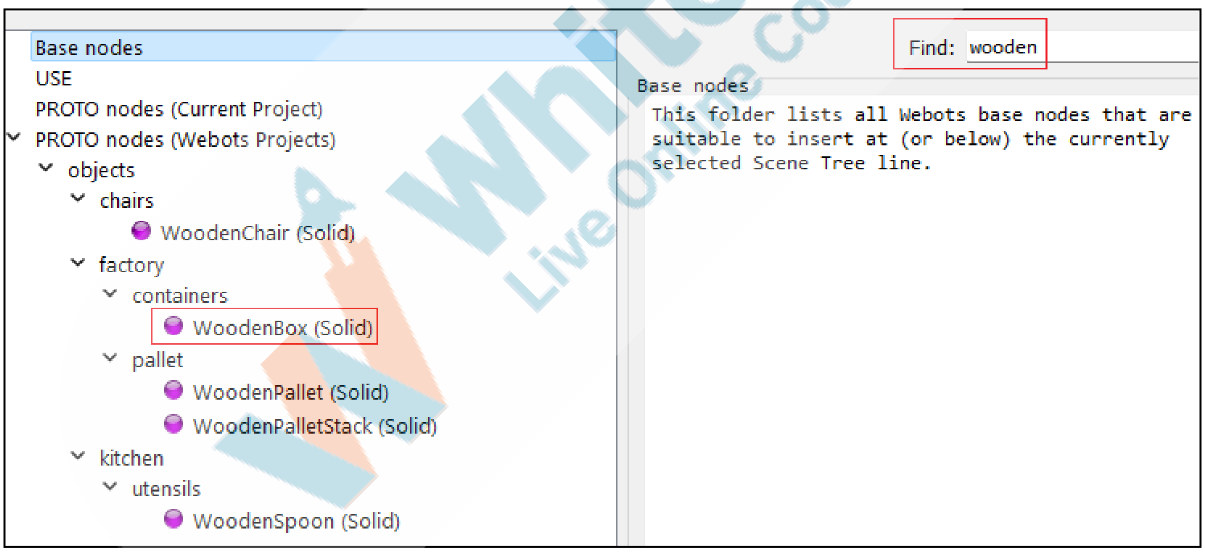
After that,

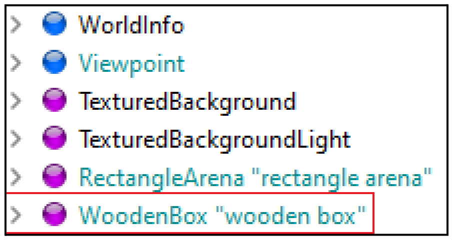
a) Expand the **PROTO nodes**.
b) Expand the **appearances**.
c) Scroll down and search for **OsbWood** appearance.
d) To add it, **double click** on it.

After that **close** the rectangular Arena node properties, by clicking on **'>' button.**



Your floor should look as shown in the graphic below.

Now, before adding a robotic arm, we need a supporting structure for it. let's add a **wooden box** for it. Click on the **Add object button** or the button with the **+ sign** on it.	
 	
A window will appear. Write the word **wooden** in the **Find textbox**. **Double click** on the **WoodenBox (Solid) node**, listed under proto nodes as, **PROTO nodes → objects → factory → containers → wooden box**	
 	
You will see a **wooden box** node added in the **scene tree**.	



The wooden box will appear in the center of the floor.

Click on the **save button** on the **simulation view window** to **save** your work till here.

Teacher Stops Screen Share

So now it’s your turn.

Please share your screen with me.

STUDENT-LED ACTIVITY 15 mins

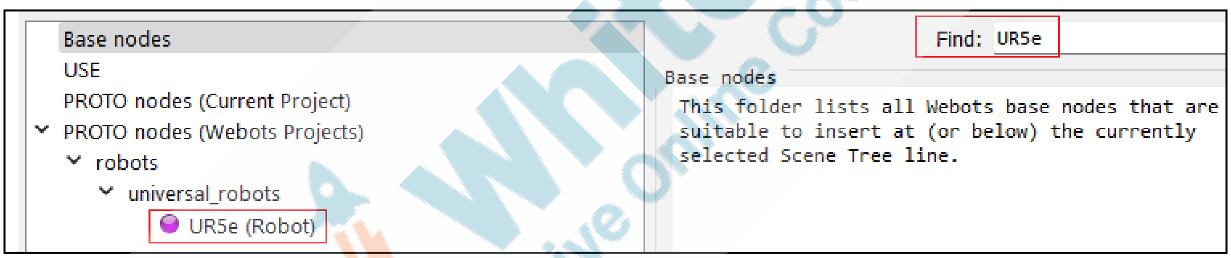
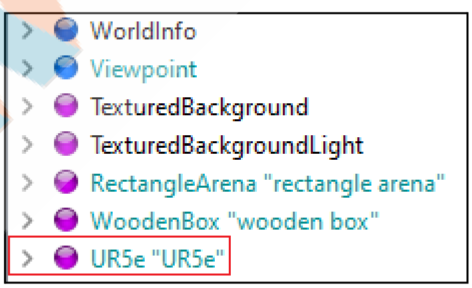
- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

Student Initiates Screen Share

ACTIVITY

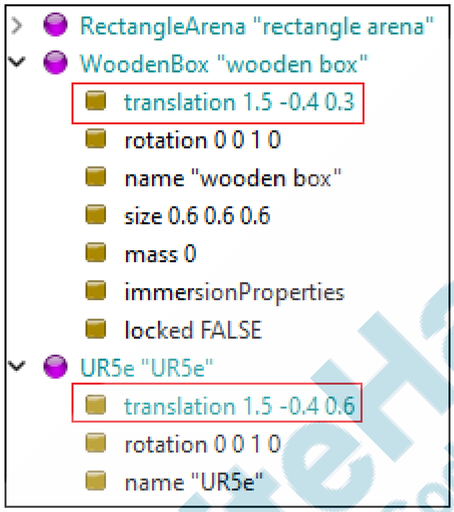
WhiteHat Jr
Live online Coding for Kids

- **Adding UR5e robotic arm.**
- **Creating a new controller.**
- **Controlling arm using keyboard control.**

Teacher Action	Student Action
Open the [student boilerplate link](), and download the **.wbt** file. Open the file in webots software. Now that we have a supporting structure for our robotic arm, let's add it. For that, click on the **+ sign** button and search for **UR5e (Universal robots 5e version)** node. It would be listed under proto nodes as, **PROTO nodes → robots → universal_robots → UR5e (Robot)**	
	
You will see a **UR5e** node added in the **scene tree**.	
	
Now we have both the nodes, let's position them appropriately. For that, **expand** both the **nodes** and set the **translation**	

as,

a) **1.5** in **x direction, -0.4** in **y direction, 0.3** in **z direction** for the **wooden box** and,
b) **1.5** in **x direction, -0.4** in **y direction, 0.6** in **z direction** for the **UR5e robotic arm.**

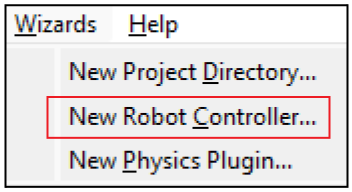


The **repositioned wooden box** and the **robotic arm** would look as shown in the graphic below.

Once we are done with the designing part, let's write some **code** so that we can move our arm with the help of the **keyboard keys**.

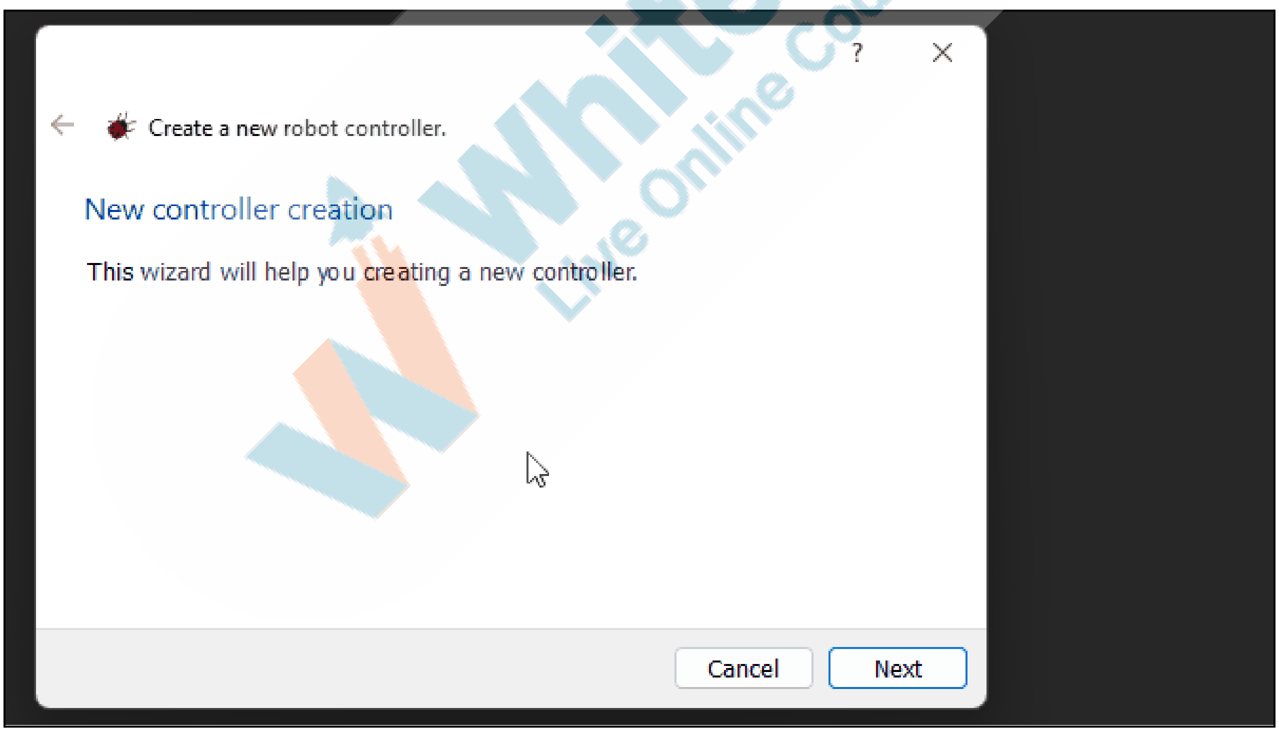
For that, let's create a **new python controller** as,

a) Click on the **Wizards** window on the top menu bar.
b) Select the **New Robot Controller** option.



A window will appear which will ask you,

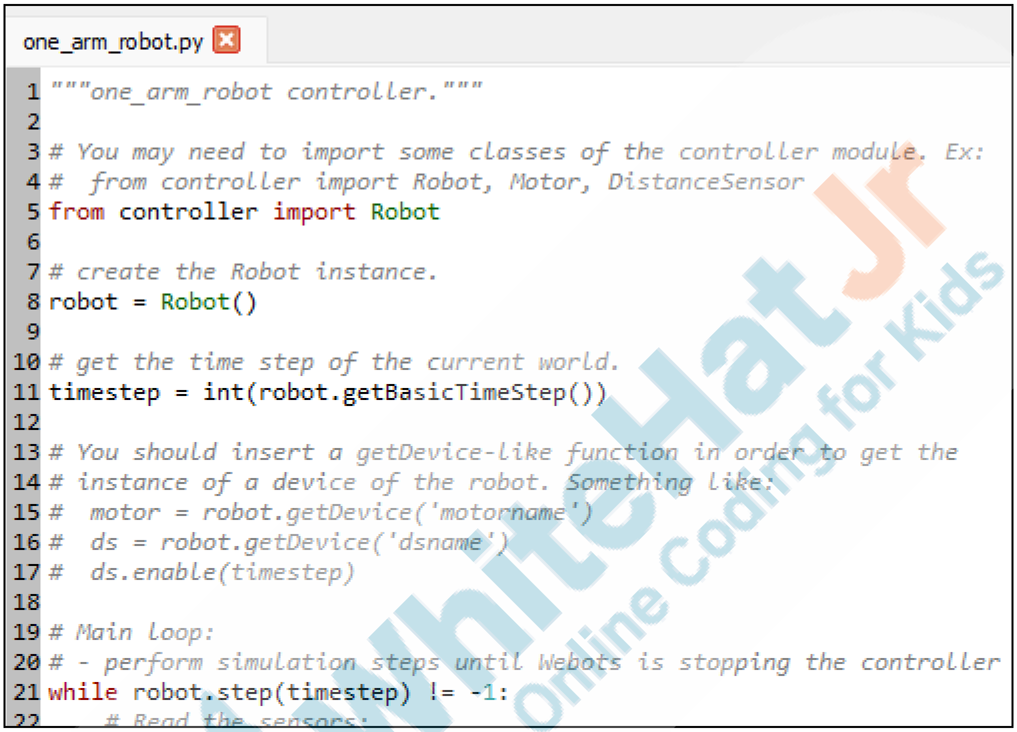
a) If you want to create a new controller or not.
   - Click on **Next**.
b) It will then ask you the language in which you want to write the code.
   - Select **python** and click on **Next**.
c) Finally, it will ask you to name your controller file.
   - Write the file name as **one_arm_robot** and click on **Next**. Although you can write any name you want.
d) Finally, click on **Finish**.



You will see that a **python file** named as **one arm**

robot.py will be opened with a standard template code in it.

Delete all the code from it as we will write our own code.



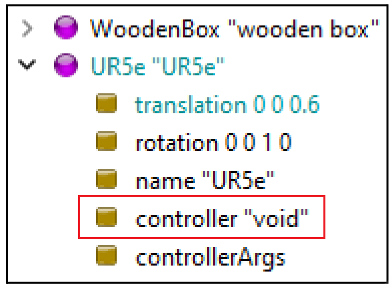
one_arm_robot.py

```
"""one_arm_robot controller."""

# You may need to import some classes of the controller module. Ex:
#  from controller import Robot, Motor, DistanceSensor
from controller import Robot

# create the Robot instance.
robot = Robot()

# get the time step of the current world.
timestep = int(robot.getBasicTimeStep())

# You should insert a getDevice-like function in order to get the
# instance of a device of the robot. Something like:
#  motor = robot.getDevice('motorname')
#  ds = robot.getDevice('dsname')
#  ds.enable(timestep)

# Main loop:
# - perform simulation steps until Webots is stopping the controller
while robot.step(timestep) != -1:
    # Read the sensors:
```

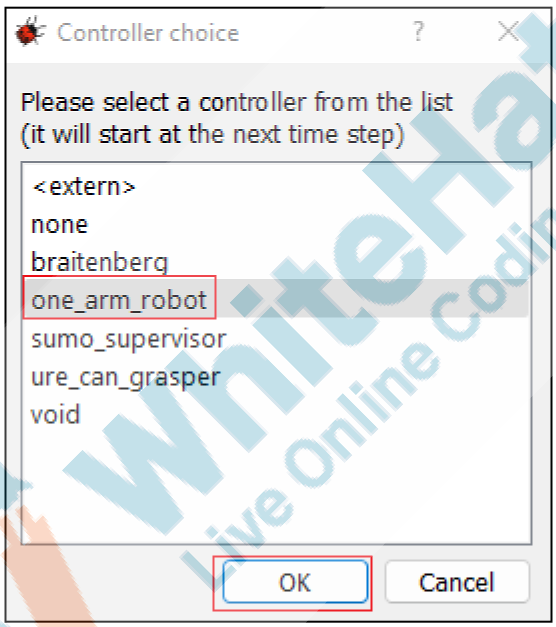
Before we start writing our code, let’s specify that our **Ur5e robot** will follow the code written in the **one_arm_robot.py** file.

For that,

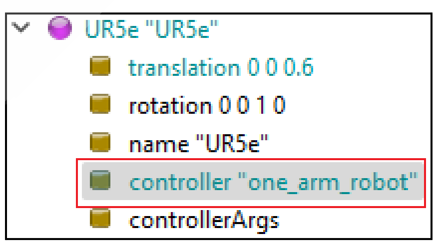
a) Expand the **UR5e** node.
b) You will see tha **controller** property. Initially it shows **void**, which means no controller is selected yet.
c) **Double** click on the **controller** property.



A list of available controllers will open up. Select **one_arm_robot** as your controller and then click on **OK**.



You will see that the controller named **one_arm_robot.py** will be selected.



Now let's start writing our code. First let's import the **Robot** and **Keyboard** class from the **controller** module as,

from controller import Robot, Keyboard	

```
from controller import Robot, Keyboard
```

After that let's create instances so that we can access the member functions of these classes as, **bot = Robot()** **keyboard = Keyboard()**	

```
bot = Robot()
keyboard = Keyboard()
```

Next, let's define the **controller timestep** as **64 ms**. This is the time **increment** executed at each iteration of the control loop of the controller.	

```
timestep = 64
```

The UR5e robotic arm has 6 joints named as, ● **shoulder_lift_joint** ● **shoulder_pan_joint** ● **elbow_joint** ● **wrist_1_joint** ● **wrist_2_joint** ● **wrist_3_joint** We will define **objects** for each of the **joints** using the **.getDevice()** method of the **robot** class using the following syntax as, **object_name = bot.getDevice('joint name')**	

```
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')
```

Let's **enable** or **initialize** the **keyboard** using the **.enable()** method, so that we can get inputs from the keyboard. It will take the **controller timestep** as an **argument**.

```
# enabling devices
keyboard.enable(timestep)
```

Also, let's create a **method** which will allow us to change the **position** for each of the **arm joints** as,

a) Create a method and name it as **move_bot()**
b) This method will take **6 arguments**, so that we set the positions for the 6 joints. Set all the arguments initially to **0**.
c) 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):

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

Call the **move_bot()** method, so that initially all the joints are at **0 position**.

```
move_bot()
```

Next, let's create our **controller loop**, where we can check for the **keyboard inputs** as,

a) Use the **.step()** method of the Robot class, as the condition of our **while** loop.
b) Within the loop, use the **.getKey()** method to get the **code** for the **key** which is pressed.
c) Print the code, so that we can use it to address the key.

while bot.step(timestep) != -1:
keypressed = keyboard.getKey()
print(keypressed)

*Note : **.step()** method returns **-1** only when webots tries to terminate the controller. This happens when we hit the reload button or try to quit webots.*

```
while bot.step(timestep)  !=  -1:
    keypressed = keyboard.getKey()
    print(keypressed)
```

Save this code and run the simulation. You will see,

a) **-1**, if **no** key is pressed.
b) **314**, if the **left arrow key** is pressed.

You will see different codes for different keys.

```
Console - All
-1
-1
-1
-1
-1
314
314
314
314
314
314
314
```

Try to find out the keycodes for the following keys and match it with the table given below,

- **Up arrow key**
- **Down arrow key**
- **Left arrow key**
- **Right arrow key**
- **w**
- **s**
- **a**
- **d**
- **1**
- **2**
- **3**
- **4**
- **5**
- **6**
- **7**
- **8**
- **9**
- **0**
- **-**
- **+**

Key	Keycode	Key	Keycode
Up arrow	315	3	51
Down arrow	317	4	52
Left arrow	314	5	53
Right arrow	316	6	54
w	87	7	55
s	83	8	56
a	65	9	57
d	68	0	48
1	49	-	45
2	50	+	61

Now let's create **6 variables** which will keep track of the joint positions as,

- **shoulder_lift_pos :** Keeps track of shoulder lift.
- **Shoulder_pan_pos :** Keeps track of shoulder pan.
- **elbow_pos** : Keeps track of elbow movement.
- **wrist_1_pos :** Keeps track of wrist movement along the Y axis.
- **wrist_2_pos :** Keeps track of wrist movement along the Z axis.
- **wrist_3_pos :** Keeps track of the hand movement along the Y axis.

Initially all the variables are set to **0**.

```
# variables to track join positions
shoulder_lift_pos = 0
shoulder_pan_pos = 0
elbow_pos = 0
wrist_1_pos = 0
wrist_2_pos = 0
wrist_3_pos = 0
```

Finally let's create some **conditional** statements, so that we can change the **variables** or the **position** of the **arm joints**, whenever **keyboard keys** are **pressed as,**

```
if keypressed == 317:   # down key is pressed
   shoulder_lift_pos += 0.01
elif keypressed == 315:   # up key is pressed
   shoulder_lift_pos -= 0.01

move_bot(shoulder_lift_pos)
```

The above code **increments** the **shoulder_lift_pos** variable by an amount of **0.01 m**, if the **down** key is pressed, and **decrements** it by **0.01m**, if the **up** arrow key is pressed.

Add other statements and write the code for other joints as well.

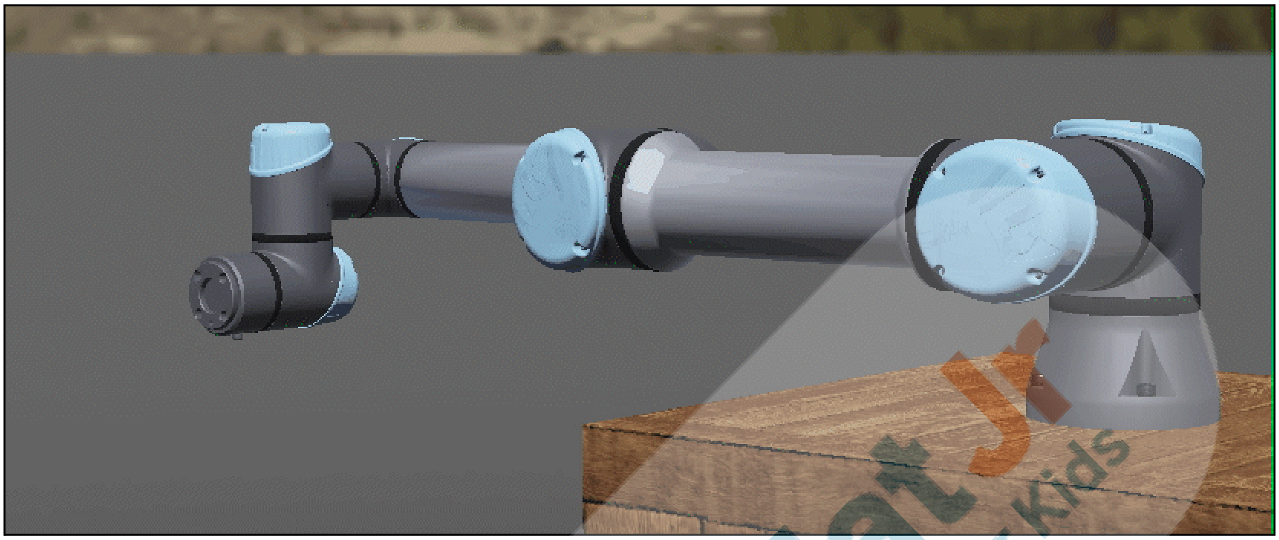
```python
while bot.step(timestep)  !=  -1:

    keypressed = keyboard.getKey()

    if keypressed  ==  317:          #  down key is pressed
        shoulder_lift_pos += 0.01
    elif keypressed  ==  315:        #  up key is pressed
        shoulder_lift_pos -= 0.01
    elif keypressed  ==  314:        #  left key is pressed
        shoulder_pan_pos += 0.01
    elif keypressed  ==  316:        #  right key is pressed
        shoulder_pan_pos -= 0.01
    elif keypressed  ==  87:        #  w key is pressed
        elbow_pos -= 0.01
    elif keypressed  ==  83:        #  s key is pressed
        elbow_pos += 0.01
    elif keypressed  ==  65:        #  a key is pressed
        wrist_1_pos += 0.01
    elif keypressed  ==  68:        #  d key is pressed
        wrist_1_pos -= 0.01
    elif keypressed  ==  49:        #  1 key is pressed
        wrist_2_pos += 0.01
    elif keypressed  ==  50:        #  2 key is pressed
        wrist_2_pos -= 0.01
    elif keypressed  ==  51:        #  3 key is pressed
        wrist_3_pos += 0.01
    elif keypressed  ==  52:        #  4 key is pressed
        wrist_3_pos -= 0.01
```

Finally, let's pass these variables as an argument to the **move_bot()** method to see the change in position of the UR5e bot.

Save your code and run the simulation.

```python
move_bot(shoulder_lift_pos, shoulder_pan_pos, elbow_pos,
         wrist_1_pos, wrist_2_pos, wrist_3_pos)
```

The final output for your code would look as shown in the gif below.



[Click here](#) to view the reference video.

Great, the UR5e robotic arm is working fine. But there is one challenge with it? Can you tell me what it is? The UR5e robot won't be able to hold or grip an object, because it does not have any **gripping hand** attached with it. In the next class we will learn how to attach a gripper hand with the **UR5e node**.	ESR : Varied

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)

<u>FEEDBACK</u>

- **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	**Student Action**
You get "hats-off" for your excellent work! In the next class, we will attach a gripper hand with your robotic arm so that you can pick and drop objects.	*Make sure you have given at least 2 hats-off during the class for:* 

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/b06b252c-0202-49ed-ae72-0c18e27fb597.pdf
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C288-Project-Solution
Teacher Reference 3	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/a230ed00-8385-4a35-986b-5663b806b642.pdf
Teacher Reference 4	Reference code	https://github.com/procodingclass/PRO-C288-Reference-Code.git
Teacher Reference 5	Final output gif	https://s3-whjr-curriculum-uploads.whjr.online/f9253648-2d06-4362-96ab-fb07d3641f95.gif
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C288-Student-Boilerplate.git