CSCI-1100

Topics in Computing
University of Georgia

Lab XX

NAO Robot: An Introduction to Python

Purpose: The purpose of this lab is to…. {{FINISH}}

Introduction

Congratulations! You've been hired to work at Athena Technologies® - an industry leading robotics corporation. Athena Technologies® specializes in programming NAO Robots. You have been hired as a Python developer – someone who writes code using the programming language called Python.

NAO Robot

https://www.aldebaran.com/en/humanoid-robot/nao-robot
https://www.aldebaran.com/en/humanoid-robot/nao-robot-working

Python

https://www.python.org/
https://www.python.org/about/apps/

In this lab, we are going to learn the following concepts:

- Programs and Algorithms
- Variables and Data Structures
- Expressions
- Loops
- Conditional Statements
- Robotics

Part 0: Lab Introduction

Programs are what make computers useful. **A program is a finite sequence of instructions written in a particular language (called a programming language) that achieves some task**. We've seen many examples of programs like Windows, Mac OSX, email clients, web browsers, video games, text editors, and many others. Programs usually achieve one or multiple tasks. For example, an email client program will log you into an email server, send and receive email, and offer many other services like organizing your emails.

Before we can dive into programming, we have to take a step back and define an important concept in computer science called an **algorithm**. **An algorithm is a finite sequence of steps that achieves some task**. An algorithm is independent of any programming language, and it can be thought of as "a way to solve a problem" or "a set of directions to perform some task". The difference between an algorithm and a program can be subtle sometimes, but remember this: an algorithm is just a general solution to a problem or way to achieve a task whereas a program is a solution written in certain programming langue that solves a problem or achieves some task.

In this lab, we are going to program a NAO robot using Python! Athena Technologies® has been hired by an elementary school district to produce robots that can help out in Math classrooms! For now, all your robot has to do is compute whether a number from 1-100 is a prime number. A prime number is a number that is only divisible by 1 and itself. Some examples of prime numbers are 7, 13, and 23, and 97. Some numbers that are not prime are 33, 45, and 81 (to name a few). Of course once the robot computes this, it will need to speak its answer out loud for the students in the classroom to hear! Thus, by the end of the lab you will become a **programmer**, a person who translates an **algorithm** into a **program** that executes on a computer. Sounds easy, right? Let's get started!

Part 1: The algorithm for determining divisibility of a number

Before we jump right into programming our robot, we need to think about how we would solve this problem. Thus, we need to come up with an **algorithm** to follow. We can't program a robot to solve this problem unless we can think of how to do it ourselves!

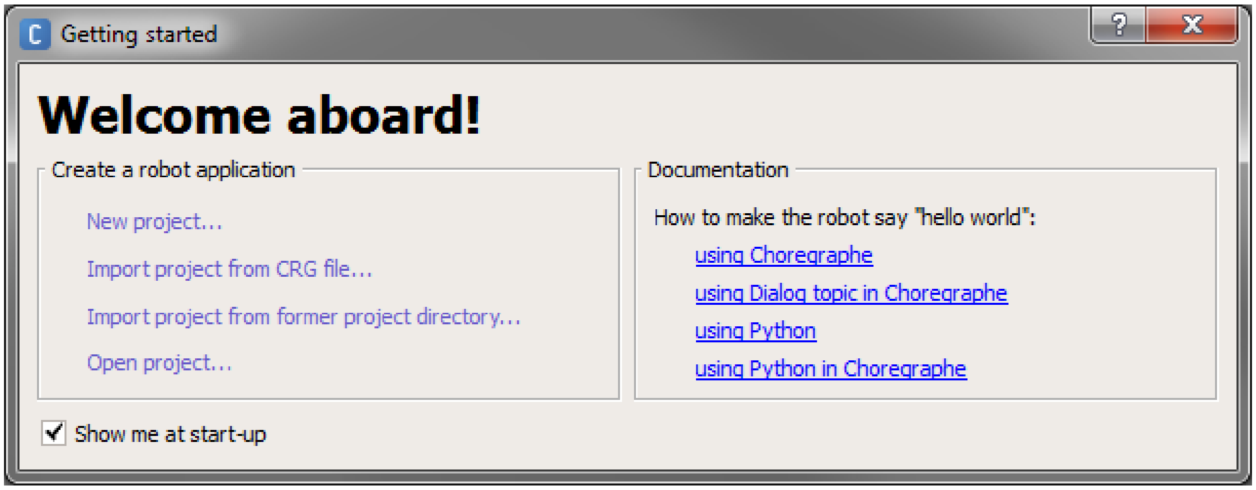
Here are the general steps for an algorithm that we will use for this lab:

1. Generate a random number to determine whether or not it is prime. Let's call this number "X".
2. We need to start looking at all of the integers from 2 until sqrt(X) and checking if any of those numbers will divide X with no remainder. Let's call these numbers "divisors".
    a. We start our search for divisors at 2 because we know that all prime numbers will be divisible by 1.
    b. We end at sqrt(X) because after we evaluate divisors up to sqrt(X) and we still haven't found a divisor that evenly divides X with no remainder, then a divisor after sqrt(X) can't possibly exist that will evenly divide X.
3. If our program reaches that final term and we still haven't found a divisor that will divide X with no remainder, then we know that X must be a prime number!
4. Lastly, we need our NAO Robot to speak out loud the number that X was, and whether or not it is prime!

Part 2a: NAO Robot's IDE: Choreographe

To start, we need to open up the software we will use to program our robot. This name of this software is Choreographe. On a Windows machine, either click the windows key on the keyboard or click start on the desktop. Search for "Choreographe" and open the program.

Upon opening Choreographe, we are greeted with the following pop-up box:



Getting started

Welcome aboard!

Create a robot application

New project...

Import project from CRG file...

Import project from former project directory...

Open project...

Documentation

How to make the robot say "hello world":

using Choregraphe

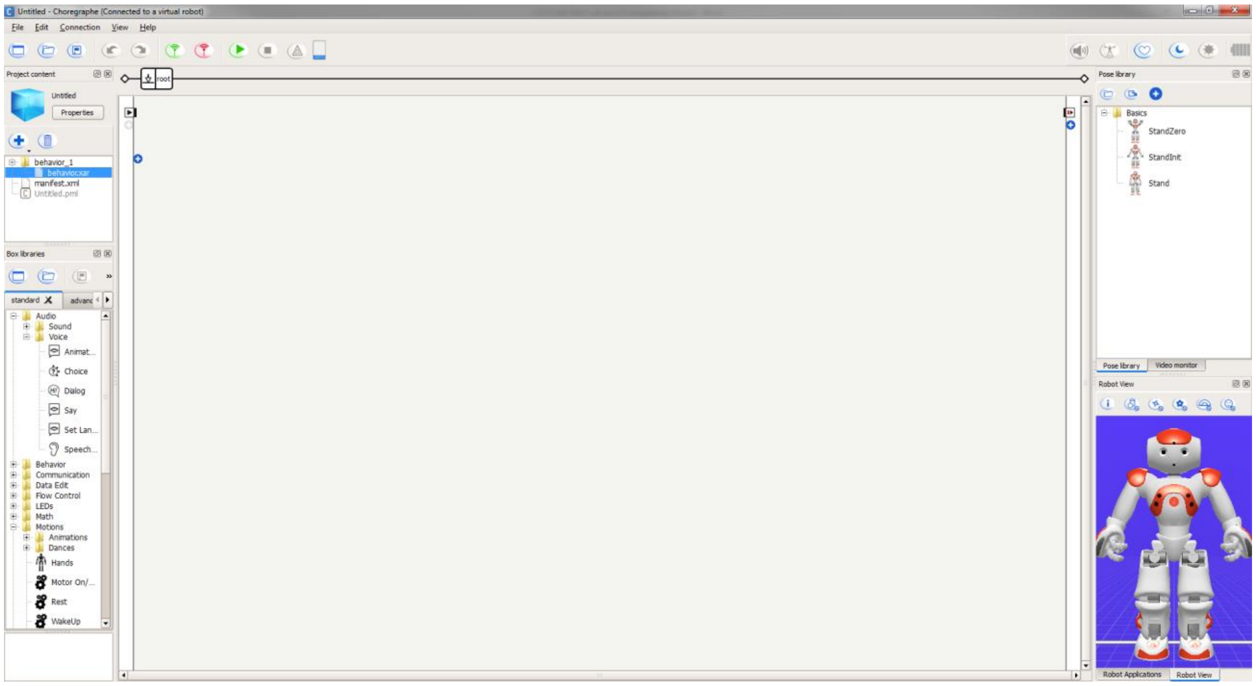
using Dialog topic in Choregraphe

using Python

using Python in Choregraphe

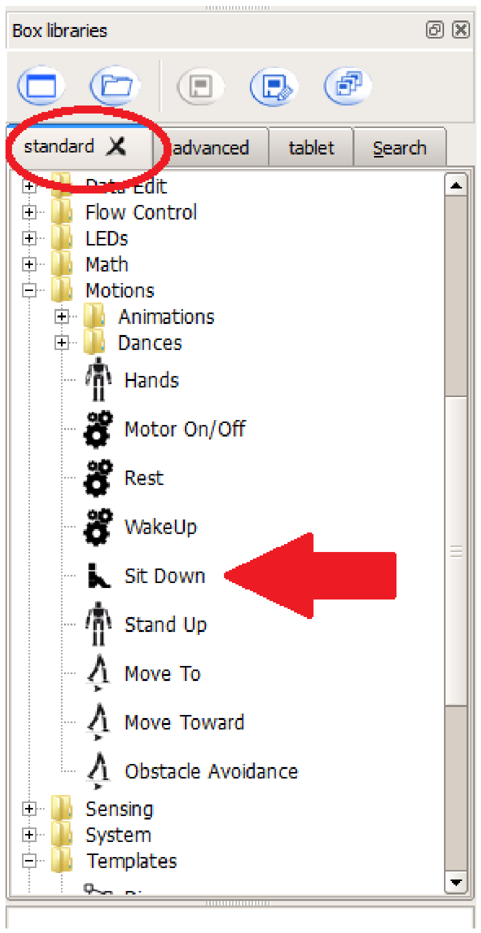
☑ Show me at start-up

We are going to want to click "New Project". Below is a screenshot of what you should see as a blank project. Choreographe is both a drag-and-drop environment as well as a text editor where you can manually write code. For this lab, we will be doing a little of both!

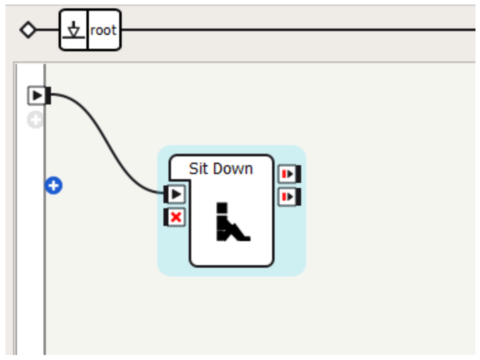


Now, on the bottom right hand side of the screen is the view of your robot that we'll be programming! Across the very top of the screen, you can see all of your options for managing projects, undoing/redoing changes, connecting to a physical robot to test your code on it in real life, and lastly you have the start/stop commands for running your project. Working down the left hand side of the screen, we see the overview of our project, and before that we have some sample pieces of code we can use as a starting point for our project. We can call this section that contains these pieces of already done code the "Box Libraries" section. We'll use this heavily throughout the lab.

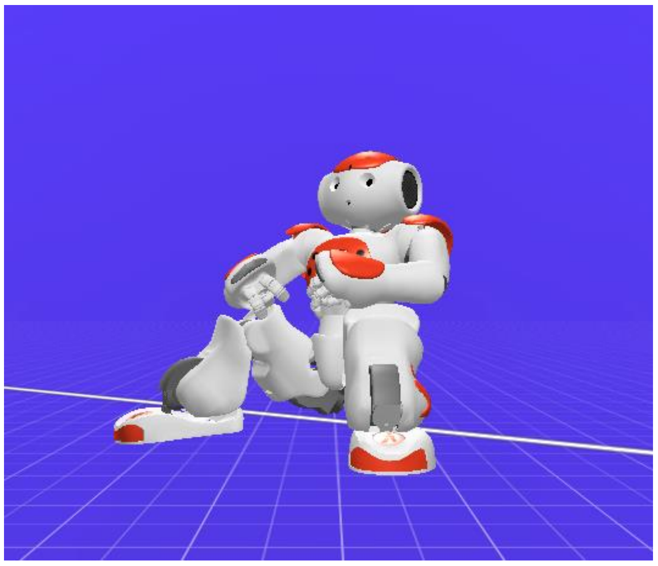
Let's begin by having our robot sit down. We can't expect our robot to do heavy math calculations while he is standing up, can we? To do this, scroll down in the Box Libraries section of Choreographe. Make sure you are on the standard tab, and keep scrolling until you see a folder called "Motions". Underneath that folder, we can see the "sit down" command. Below is a screenshot of what that looks like.



To use "Sit Down" in our project, click and drag it from the box libraries section into the main project area. Try and keep it somewhere in the top left of the project area. This will keep everything nice and neat. Now, the very next thing we want to do is gran the little black arrow attached to the top right side of the project area, and we want to drag that onto the matching black arrow of our “Sit Down” box. Below is a screenshot of what it looks like when you’ve completed this step.



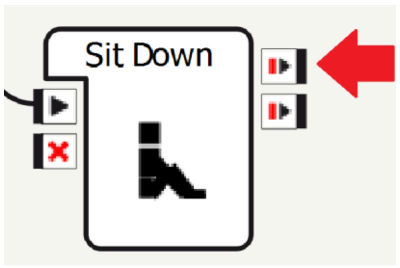
If you want to see our robot in action, click the green “Play” button in the toolbar across the top of Choreographe. You can watch your robot follow the command, and sit down.



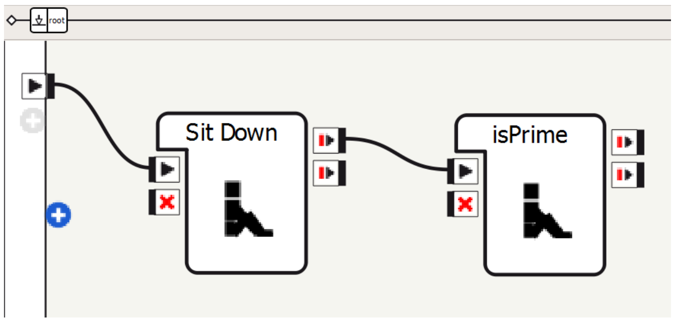
Part 2b: Finishing the Drag-And-Drop

Now that we know we can add actions for our robot to perform by dragging and dropping them from the Box Libraries section on the left hand side of the screen, we are going to add one last action for him to perform and we are going to edit and customize it to perform the math we need it to do!

Start by clicking and dragging another "Sit Down" from the Box Libraries section. Place it somewhere to the right of the first "Sit Down". The second one is called "Sit Down (1)" to differentiate it from the first one that we already made. Let's rename it! Right click on "Sit Down (1) and click on "Edit Box". Our robot is already sitting down, so we just need this box to perform our math for us. So, we are going to fully customize this action and make it our own. First, you should change the name of "Sit Down (1)" to "isPrime". Giving our actions useful names is really important when you work at a company like Athena Technologies®. Next, edit the description of our box. Instead of the description talking about sitting down, we want the description to say something along the lines of how it will compute whether or not a number is prime. Leaving useful information like this as you work is a form of "Commenting". Commenting is when programmers leave notes about their work so that when someone else looks at the project, they can see how things work and what they do. After editing the name and the description of our box, click "OK". The very last thing we want to do right now is connect our "Sit Down" box to our "isPrime" box. You do this by clicking and dragging one of the outputs from "Sit Down" and connecting it to the input of "isPrime". Below is a screenshot of an output that you'll want to connect into "isPrime".



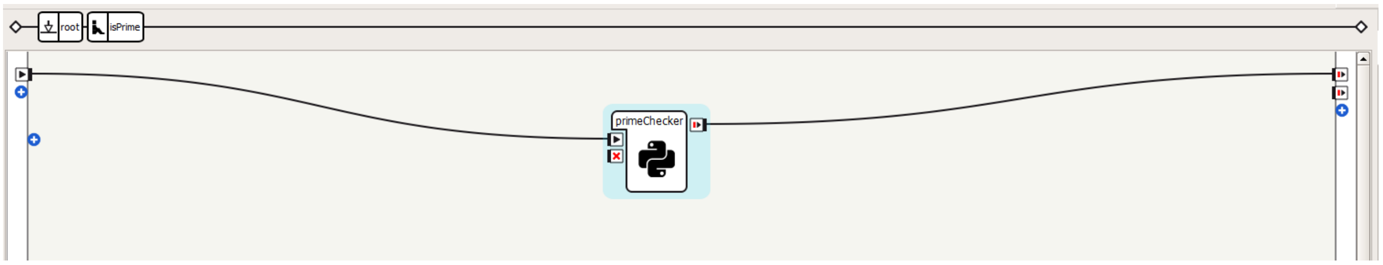
Once they are connected, we are all done with the drag and drop portion of the lab and can start programming in Python!



Part 3: Programming in Python

Now that we are done setting everything up in Choregraphe, it's time to get started writing code in Python. As stated before, Python is a programming language and we will use it to tell our robot how to compute whether or not a number is prime.

Start by double clicking on the isPrime box. This brings you down a level onto isPrime. In here, you can see everything that isPrime does. For now, it is filled with another box called "Goto Posture". This is leftover from the "Sit Down" command that our box used to be. You can see how input comes in from the left, goes into the "Goto Posture" box, leaves the box from an output, and then goes to the output on the right side of the screen area. For now, we want to completely replace this box. So, click on it ONCE to select it, and then click the "Delete" key on your keyboard. This will completely remove it. Next, we want to right click on the project area in the middle and select "Create New Box" and under that menu click "Python…". This will be the Python code that contains the instructions for checking to see if a number is prime. For now, we can call this new box "primeChecker", as this will be the name of the code that checks to see if a number is prime. You can also fill in the description with a short message describing that this block will check if an integer is prime. The last step to do for now is to connect up "primeChecker" to be just like how the "Goto Posture" box was connected before. This means that the output from the left side of the work area will go to the input on the left side of "primeChecker" and then the output of "primeChecker" will go to the output on the right hand side of the page. Below is a screenshot of what your lab should look like:



Notice across the top we have something similar to a road map of where we are in the project. Remember how we double clicked on isPrime? That is similar how to you'd click on a folder in your computer. It takes you inside that folder so you can view everything inside of it. The root is like level 0. There are no more levels above the root, which is why it appears first on the road map. If we click on "root" on the road map, we are taken up a level and back to where we were before. For now, let's stay inside isPrime so we can work on the Python.