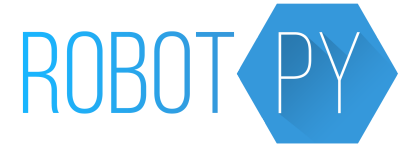


Robot programming & simulation using Python

Dustin Spicuzza
September 10, 2016

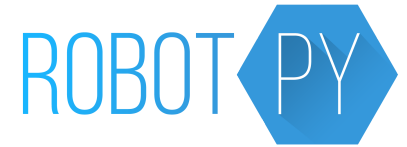
NE FIRST University Day

Agenda



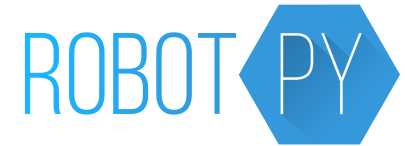
- Why Python?
- Quick Simulation Demo
- Quick Python Primer
- Putting it all together
- RoboRIO Demo

Intended Audience



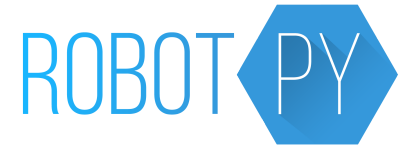
- Some familiarity with WPILib in C++/Java
- For maximum benefit, you should have some programming experience
 - If you don't, that's ok too

Who Am I?



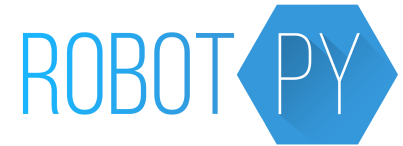
- Software engineer at BBN Technologies
- Mentoring FRC since 2009
- Co-maintainer of RobotPy since 2010
- My contributions have helped win awards for my teams:
 - 2012 Boston Regional; Innovation in Control
 - 2013 Boston Regional; Innovation in Control
 - 2014 Virginia Regional; Industrial Design
 - 2015 Greater DC Regional; Innovation in Control
 - 2016 Chesapeake Champs; Innovation in Control

What is Python/RobotPy?



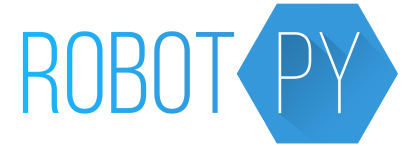
- Python is a widely used general purpose, high-level programming language
 - It's pretty sweet
- RobotPy is the project that packages python for the cRio and RoboRIO
 - And maintains various other FRC related python packages

Why Python?



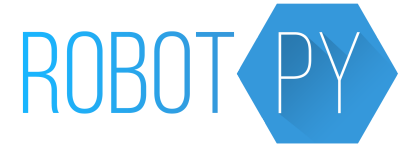
- Simple language syntax
 - Fewer brackets and semicolons
 - Indentation is mandatory
 - Very understandable and readable
- Designed for Rapid Development
 - Quick iteration, no compilation
 - Dynamic typing
- Cross platform
 - Windows, OSX, and Linux

Why RobotPy + Python?



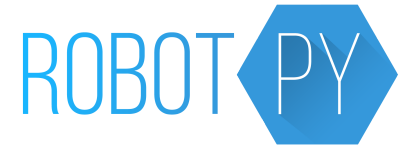
- Mature, proven codebase on high performance robots
 - RobotPy project started in 2010
- Supported for FRC teams by FRC teams
 - Quick bugfixing, because we're using this code too
 - Have you ever submitted a bug report to WPILib? They don't respond quickly*
- Good documentation

Why RobotPy + Python?



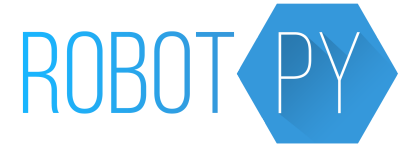
- Superior support for running your code without a robot
 - Integrated unit testing (with builtin tests)
 - Zero-configuration low fidelity simulator

Why Not Python?



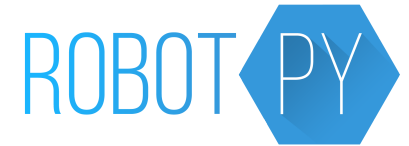
- Not officially supported by FIRST
- Community isn't as big as official languages
 - 33 teams used Python in 2016
- Support at competitions is low
 - However, because WPILib is the same, other teams can help you with problems that aren't specific to python
 - Most problems are WPILib problems, not python problems

Why Not Python?



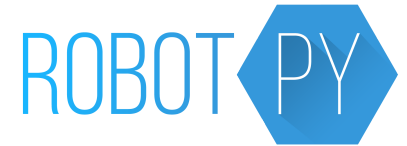
- Spacing matters
 - Indentation is syntax
- Requires that you actually test your code
 - Syntax error or mistyping something could crash your code
 - You DO test your code, right?

Quick Demo



- Team 1418's robot code for 2015
 - Same code runs on the Robot & on your PC
 - Running as simulation with pyfrc
 - Supports connection to UI on driver station laptop
 - Innovation in Control award @ DC Regional

How does this work?



- Dashboard is HTML/Javascript
- Robot code is written in python
- Pure python implementation of WPILib
- Communications via pynetworktables
 - Compatible with SmartDashboard/SFX
- Simulation engine is pyfrc

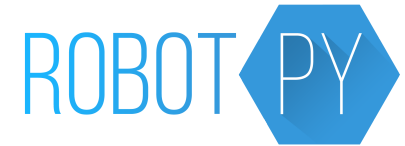
- A python library designed to make developing robot code easier

- Easy to use testing framework you can use to test your robot code at home
- Tool to upload code to the robot
 - Integrated with testing framework to prevent uploading bad code
- Low fidelity robot simulator

Python syntax compared to Java

PYTHON PRIMER

Python primer: comments



Python

```
# This is a comment
```

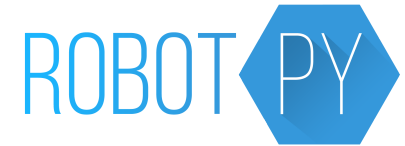
```
'''This is sometimes  
used as a multiline  
comment'''
```

Java

```
// This is a comment
```

```
/* This is a multiline  
comment */
```


Python primer: variables



Python

```
a = 13
b = False

c = 'foo'
c = "foo"

d = None

e = Bar(c)
```

Java

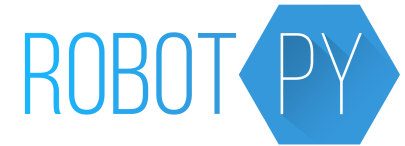
```
int a = 13;
boolean b = false;

String c = "foo";

Object d = null;

Bar e = new Bar(c);
```

Python primer: operands



Python

```
a and b
```

```
x == 3 and y == 7
```

```
a == x or b == y
```

```
x == "string"
```

```
o is None
```

Java

```
a && b
```

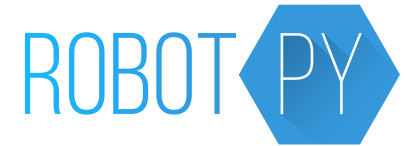
```
x == 3 && y == 7
```

```
a == x || b == y
```

```
x.equals("string")
```

```
o == null
```

Python primer: functions



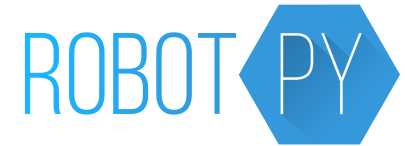
Python

```
def hello():  
    print("Hi")  
  
def add2(a):  
    return a + 2  
  
# Call the function  
hello()  
r = add2(2)
```

Java

```
private void hello(){  
    System.out.println("Hi");  
}  
  
public int add2(int a){  
    return a + 2;  
}  
  
// Call the function  
hello();  
int r = add2(2);
```

Python primer: if/else



Python

```
if a:
    do_something()

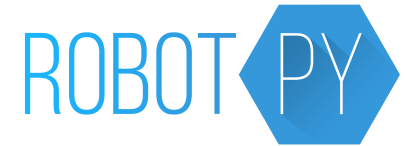
elif b == 4:
    doit(b)

else:
    done = False
```

Java

```
if (a == true) {
    do_something();
} else if (b == 4) {
    doit(b);
} else {
    done = false;
}
```

Python primer: objects



Python

```
class Obj:

    def __init__(self):
        super().__init__()
        self.x = 2

    def getX(self):
        return self.x
```

Java

```
public class Obj {
    int x = 1;

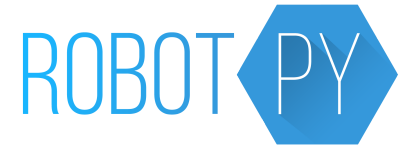
    public Obj(){
        super();
        x = 2;
    }

    public int getX(){
        return x;
    }
}
```

Putting it all together

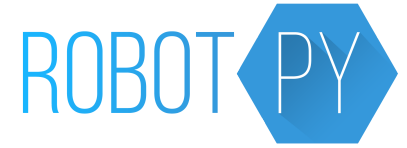
REAL ROBOT CODE

Real robot code



- python uses WPILib
 - Robot code isn't that different from C++/Java, same principles still apply
- Let's create a robot that changes a solenoid based on a joystick trigger

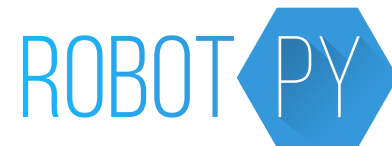
Real robot code: robot.py



- Create a file called robot.py
- First, import WPILib so you can use it

```
import wpilib
```

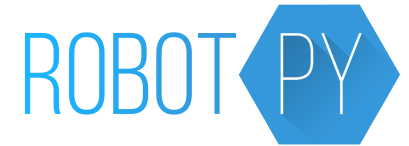

Real robot code: MyRobot



- Next, need to define a robot object
 - We'll use IterativeRobot for simplicity

```
class MyRobot(wpilib.IterativeRobot):  
  
    def robotInit(self):  
        '''Called at startup'''
```

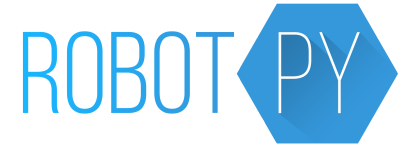
Real robot code: MyRobot



- Create a joystick and some devices in the robotInit function

```
# Note this is still indented..  
self.joystick = wpilib.Joystick(1)  
  
self.dio = wpilib.DigitalInput(1)  
  
self.solenoid1 = wpilib.Solenoid(1)  
self.solenoid4 = wpilib.Solenoid(1)
```

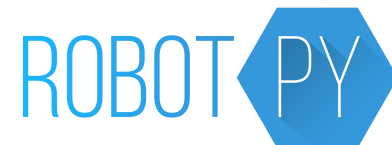
Real robot code: MyRobot



- Next, define your teleoperated code

```
def teleopPeriodic(self):  
    # control solenoid 1 via joystick trigger  
    self.solenoid1.set(self.joystick.getTrigger())  
  
    # control solenoid 4 via digital input  
    self.solenoid4.set(self.dio.get())
```

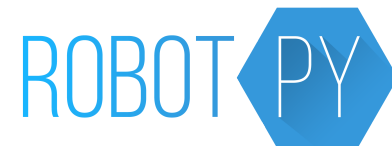
Real robot code: the end



- Finally, robotpy needs some stuff to run your code correctly

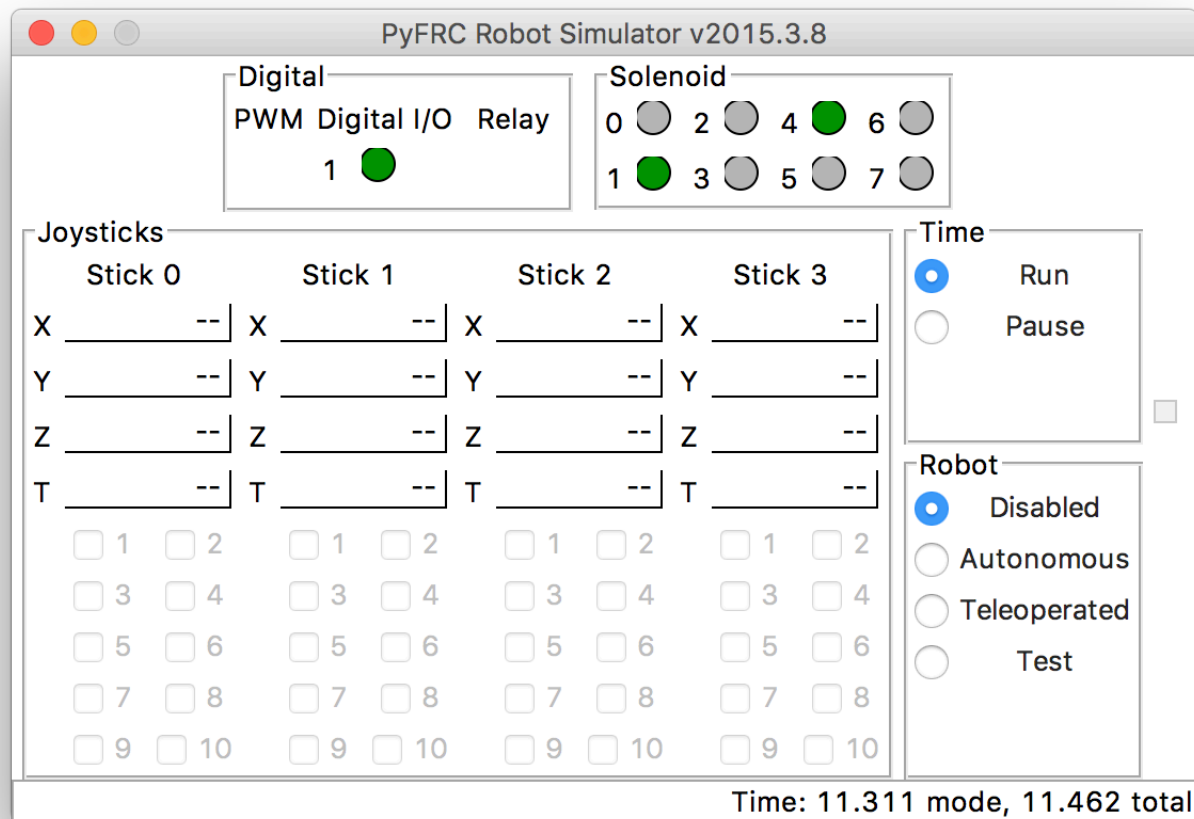
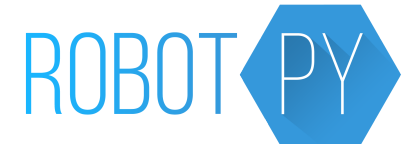
```
if __name__ == '__main__':  
    wpilib.run(MyRobot)
```

Real Robot Code: Running it



- Use python3 to execute robot.py, and pass it the 'sim' argument
- Windows: Open cmd, and...
 - cd path\to\robot
 - py -3 robot.py sim
- OSX: Open Terminal, and...
 - cd path/to/robot
 - python3 robot.py sim

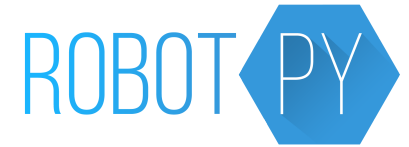
The result



Putting it all together

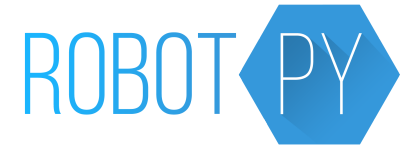
DEMO

Demo



- A small snippet of code that:
 - Runs on a RoboRIO
 - Runs in simulation
 - Connects to a webpage

Demo

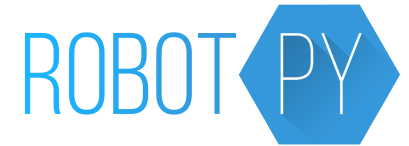


- This is the robot code that does the work

```
sd = wpilib.SmartDashboard
sensor = wpilib.AnalogInput(2)
...

sd.putDouble('sensor', sensor.getVoltage())
```

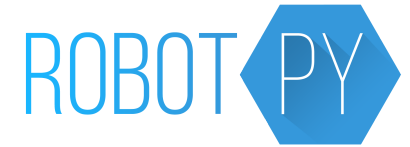
Demo



- Let's make a webpage!
 - Javascript to interact with pynetworktables2js

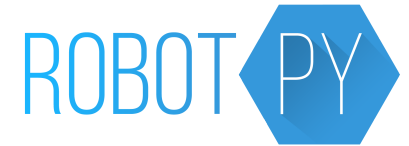
```
var nt = NetworkTables;  
var key= '/SmartDashboard/sensorDegrees'  
  
nt.addKeyListener(key, onValueChanged, true);  
  
function onValueChanged(key, value, isNew) {  
    // do something with it  
}
```

Encoder demo



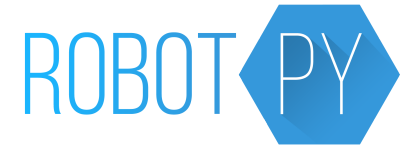
- Let's run this code!

Want More?



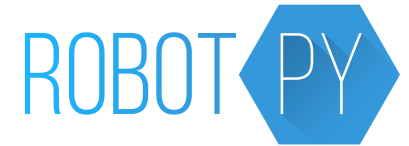
- pyfrc has 'physics' support
 - Allows you to adjust various robot objects on the fly to make the simulation more useful
 - There might be a web-based simulator in 2017
- There are more pyfrc samples on github
 - <https://github.com/robotpy/pyfrc/tree/master/samples>

Want More?



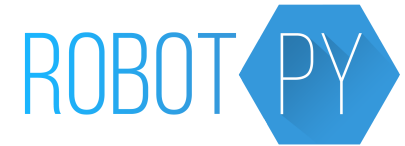
- This presentation + code available online
 - <https://github.com/virtuald/frc-robotpy-workshop>
 - <https://github.com/frc2423/encoder-demo>

Learn RobotPy



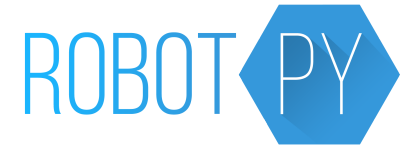
- Anatomy of a robot
 - <http://robotpy.readthedocs.io/en/latest/guide/anatomy.html>
- pyrobottraining
 - <https://github.com/robotpy/pyrobottraining>

Learn Python



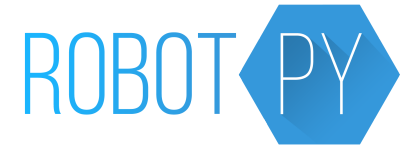
- Code Academy
 - <http://www.codecademy.com/tracks/python>
- Wikibooks Python Tutorial
 - https://en.wikibooks.org/wiki/Non-Programmer's_Tutorial_for_Python_3
- Official Python Tutorial
 - <https://docs.python.org/3.5/tutorial/>

Resources



- RobotPy mailing list
 - <https://groups.google.com/forum/#!forum/robotpy>
- ChiefDelphi Forums
 - <http://www.chiefdelphi.com/forums/forumdisplay.php?f=187>

Code Links



- RobotPy github site
 - <https://github.com/robotpy>
- pyfrc documentation
 - <http://pyfrc.readthedocs.io/>
- Team 1418's 2015 code
 - <https://github.com/frc1418/2015>

One more thing...

- Publicly editable repository of information related to FIRST Robotics
 - Technical topics
 - Non-technical
 - Team pages
- Add content to your team's page!

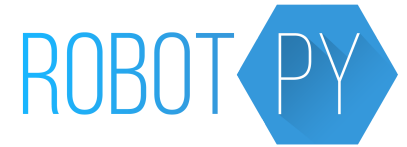


<https://firstwiki.github.io>

Questions

???

Special Thanks



- Team 1418 Programming Team
- Tim Winters for python tutorial help