

Principles of Robot Autonomy I

Section Logistics

Sections

- Modeled after sections in CS 106A/B/X/L
- Provide hands-on experience for commonly-used tools in robotics
 - AKA tools you'll be using for your homework and final projects
- Taking feedback from previous years to heart

Section Logistics

- First 15-30 minutes will be a presentation about the aims of the section, references, and a description of the hands-on activity you'll be doing
- Rest of the time (1.5+ hours) will be for you and a partner (your tablemate) to complete the hands-on activity
- You submit your results on Gradescope when you're done

Do I have to stay the whole time?

- Once you complete the activity and submit your results, you can leave

Do I have to arrive on time?

- Yes
- ... unless you have an overlapping class conflict. In that case, you should still arrive ASAP and make a group with someone else that is arriving similarly late
 - If you're *the only one* that arrives late, then you can join an existing group
- Section slides and the activity handout will be posted online, so you can still catch up
- However, we *will not* stay after hours

Questions about Section Logistics?

Principles of Robot Autonomy I

Section 1: Introduction to Python2.7, Git, and Debugging

Aims

- Learn how to use Git for version control
- Start working with Python 2.7 and some of its most common packages
- Tips and tricks for debugging

OS Setup

- Only need a local Python 2.7 installation for the first two homeworks
 - Later, we'll be providing a server for running ROS remotely and rendering the visualization on your laptop
- You have three options:
 1. Install Python and run your scripts locally on Windows/MacOS/Linux
 2. Set up a dual boot with Ubuntu 16.04
 3. Set up a virtual machine with Ubuntu 16.04

Git

- Popular source code version control system
- You probably already use it!
 - Github, BitBucket, etc. all support Git
- Replaces the days of
 - Important_doc.docx
 - Important_doc_v2.docx
 - Important_doc_final.docx
 - Important_doc_final2.docx
 - Important_doc_final2_USE_THIS_ONE.docx

Git

- Strongly recommend getting used to using Git on your homework
 - If you have a Github account, you can fork the homework repo and clone that out

Python 2.7

- We assume you already have some programming experience at the level of CS 106A
- As a result, rather than providing a full-blown tutorial about Python, we'll direct you to last year's Python + NumPy tutorial (hands-on!)
- It can be found online at:
http://asl.stanford.edu/aa274_win1819/pdfs/recitation/Tut3_NumPy.pdf

Debugging

- pdb

```
1 #!/usr/bin/env python
2
3 #This script will introduce us to the basics of using Python
4
5 #Imports
6 import pdb
7 import math
8
9 #Variables
10 initial_integer = 1
11 initial_float = 1.0
12 pdb.set_trace()
13 cast_int_to_float = float(initial_integer)
14 cast_float_to_string = str(initial_float)
15
16 #Printing
17 print("Hello world!")
18 name = "Your Name"
19 print("Hello my name is {}".format(name))
20
21 #Functions
22 print("\nFunctions:")
23 y = lambda x: math.sin(x)
24
25 def sin(x):
basics.py 9,10 Top
```

```
(base) acauligi@Abhisheks-MBP:~/Desktop$ python basics.py
> /Users/acauligi/Desktop/basics.py(13)<module>()
-> cast_int_to_float = float(initial_integer)
(Pdb) initial_integer
1
(Pdb) initial_integer + initial_float
2.0
(Pdb) cast_int_to_float
*** NameError: name 'cast_int_to_float' is not defined
(Pdb)
```

Debugging

- More generally the debugging process is:
 - Google it
 - Ask a peer in the class
 - Google it
 - If all else fails, then post on Piazza or at office hours

Section 1

- Focuses on Python and common use-cases for it in this course
- We'll ask you to perform a few basic mathematical operations and plot the results