Python for Java Pros

CS 540: Introduction to AI
Anthony Gitter

Slides created by Hobbes LeGault and Xiaojin Zhu (UW-Madison),
lightly edited by Anthony Gitter

A Crash Course in Python

1. Why are we doing this in Python?
2. Where do I write Python code? How do I run it?
    a. Online
    b. Offline
3. What are the big differences between Java and Python?
4. TAs are preparing more tutorials and background material

Why Python?

- Flexible styles: object-oriented, procedural, functional
- Interpreted language, good for exploratory analysis
    - read–eval–print loop (REPL)
- Vast collections of 3rd party pacakges

Why Python?

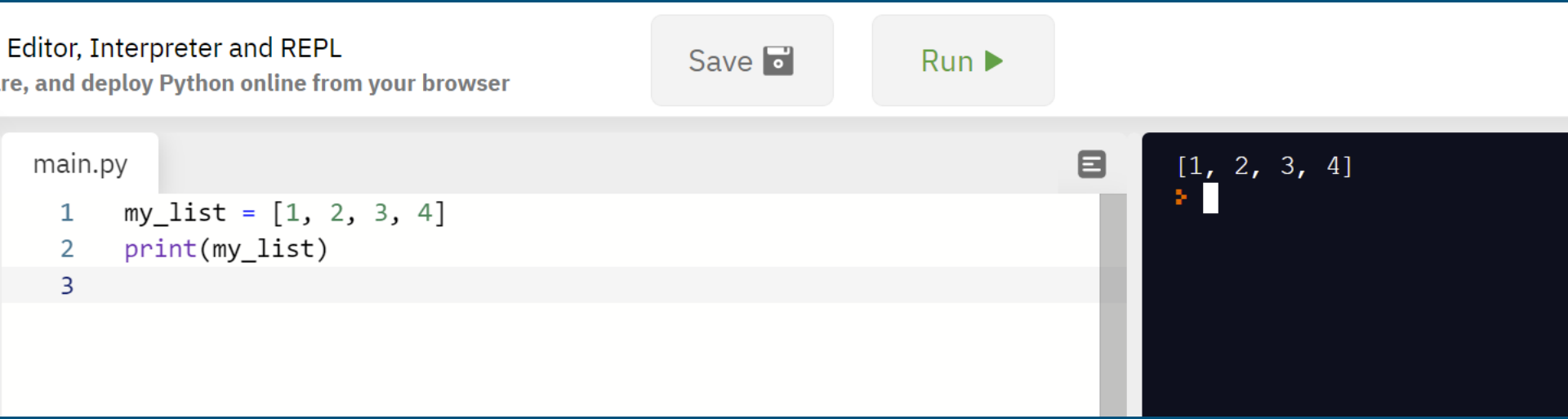
Better machine learning libraries!

Where Python?: Online

Not ideal in the long run, but sufficient for today. Difficult/impossible to customize, but easy to get up and running.

repl.it/languages/python3

Where Python?: Online



repl.it/languages/python3

Where Python?: Offline

Be real cool: vim/emacs + command line `python3`

IDEs:

- Anaconda/Spyder
- PyCharm
- Thonny
- Atom
- Eclipse + plugins if you *really* love eclipse for some reason

Many libraries have installers, but get to know `pip` (and `conda`)

Hello World: Key differences from Java

Let's translate the traditional first program to Python.

```
public class Hello {
    public static void main(String[] args) {
            // print to the console
            System.out.println("Hello, world");
    }
}
```

Hello World: Key differences from Java

```
public class Hello {
        public static void main(String[] args) {
                // print to the console
                System.out.println("Hello, world");
    }
}
```

Don't bother with a class unless you actually want to make an object

Functions don't need return types (or parameter types, for that matter)

Indentations matter, not { }. Begin functions with : and end by unindenting

```
def main(args):
        # print to the console
        print('Hello, world')
```

Strings can be " " or ' ', comments begin with #, and no semicolons needed

Python Control Flow

Conditionals and loops have the same indentation rules as functions.

```
if x > 5:
    # do something

for i in range(5):
    print(i)
```

Note: for loops in Python are really for-each loops, and need some iterable to iterate over (e.g. list, string, etc.)

Operators

Alas poor ++ operator, we knew ye well

```
x = 0
while x < 10:
    x += 1
```

Otherwise things pretty much work the same.

Comprehensions and generators

Create a new list by applying an operation to members of existing list

```
squares = [square**2 for square in range(5)]
print(squares)

> [0, 1, 4, 9, 16]
```

Generator is similar but does not store all items in memory

Reading files is easy

No Scanners, no BufferedReaders.

```
with open(filename, mode) as f:
    for line in f:
            print(line)
# closes automatically when you unindent
```

There are also libraries like pandas for reading formatted files like CSVs.

How to get Python libraries

To get access to any code beyond the basics: import

```
import math

x = 12 + 144 + 20 + 3 * math.srqt(4)
print(x / 7 + 5*11)
```

Specialized libraries (like the ones we'll be using for ML) will need to be installed before you can import them.

PYTHON PRACTICE

```
>>> filenums('nums.txt')
=> 23
```

1. Make a text file with some numbers in it (not code)
2. Write a program to read the file, sum the numbers, and print the sum to the screen
3. Challenge: put it in a function and get the filename as user input -> pass to function as argument, return total