Intro to Python

Introductions

Sri Kanajan

Senior Data Scientist, 16 years of experience in various engineering and data science roles. Instructor for part time data science class for last 2 years.

Objectives for Class

- Build strong foundation of Python
- Remove barriers/frustration
- Enable you to approach more advanced topics
  - Data Analytics
  - General scripting and automation

HAVE FUN!

Class Introductions

- Name
- Expectations for this course
- How do you plan to apply skills your Python skills?
- Fun Fact

Course Structure

- Lectures on topics
  - Interaction is good
  - Feel free to ask questions
  - If there’s not enough time to cover questions, we’ll put it in a parking lot for after class
- Hands on exercises
  - Pair programming
  - Mix up partners

Why Python?

- Readability
- Flexibility
  - Dynamic typing
  - Supports multiple programming paradigm (Object oriented, Functional, Procedural)
- Entire ecosystem for interacting with web

Libraries of Tools for Data Analysis

What is Anaconda?

- Distribution of Python and commonly used libraries of tools
- Easier than individually installing many libraries
- Ensures the versions of each library are compatible with each other

How to Interact with Python

There are several ways to interact with python

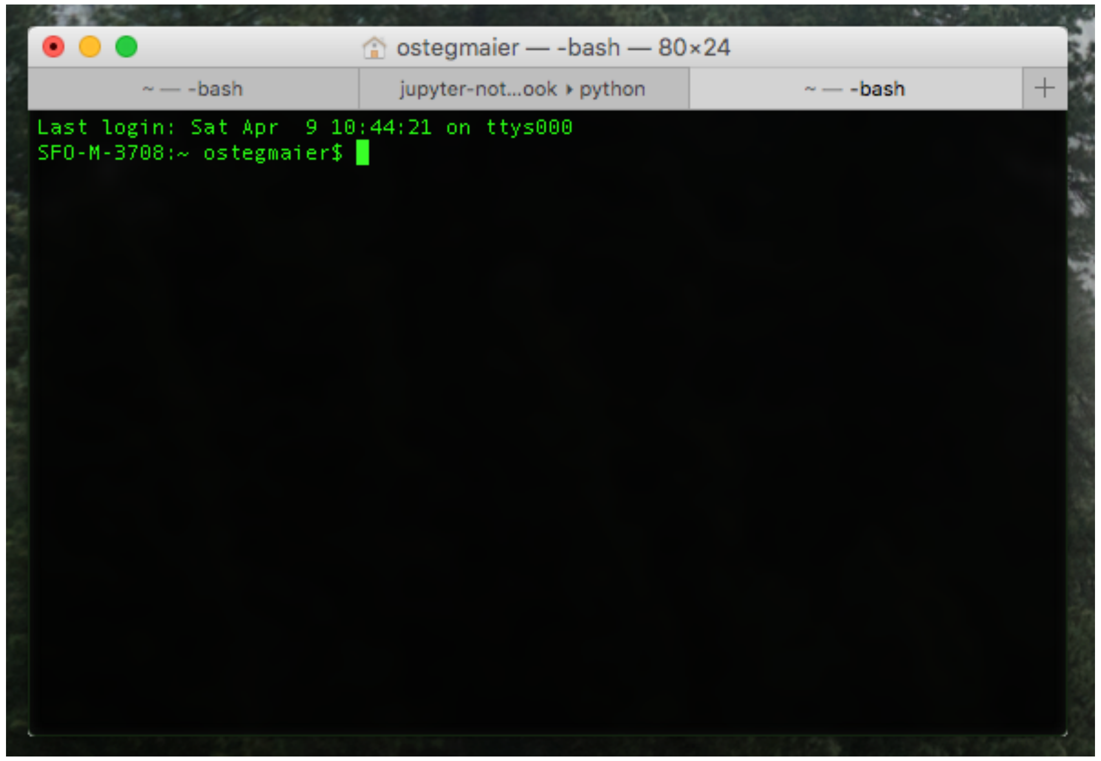
- Python Command Line
- Operating System Command Line
- iPython
- iPython Notebook

  ** Sometimes called jupyter

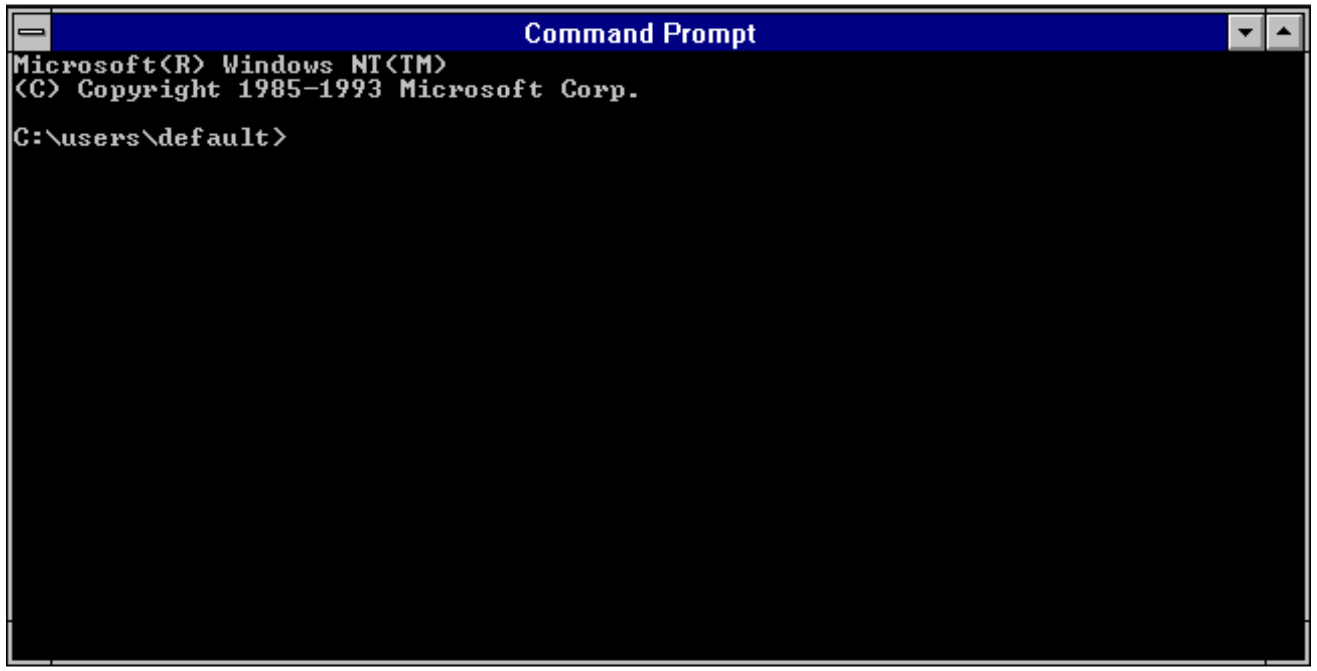
We’ll be using iPython Notebooks

Set Up for iPython Notebooks

- Download the dropbox folder to your desktop
- Open your terminal/command prompt and launch ipython notebook



```
Last login: Sat Apr  9 10:44:21 on ttys000
SFO-M-3708:~ ostegmaier$
```

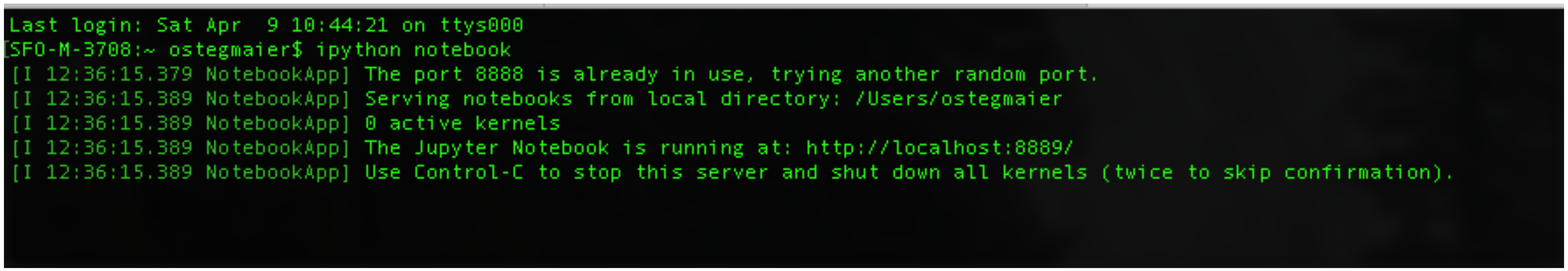


```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1993 Microsoft Corp.

C:\users\default>
```

Set Up for iPython Notebooks

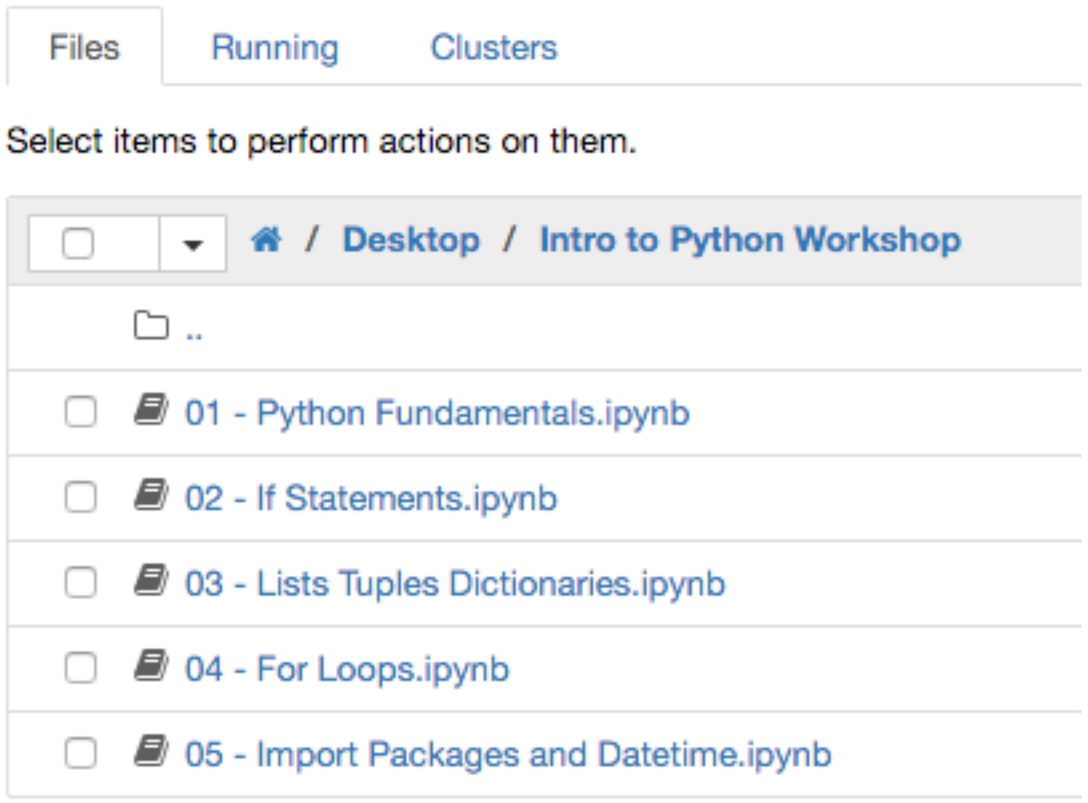
- launch ipython notebook by typing
- “ipython notebook”



```
Last login: Sat Apr  9 10:44:21 on ttys000
[SFO-M-3708:~ ostegmaier$ ipython notebook
[I 12:36:15.379 NotebookApp] The port 8888 is already in use, trying another random port.
[I 12:36:15.389 NotebookApp] Serving notebooks from local directory: /Users/ostegmaier
[I 12:36:15.389 NotebookApp] 0 active kernels
[I 12:36:15.389 NotebookApp] The Jupyter Notebook is running at: http://localhost:8889/
[I 12:36:15.389 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

Set Up for iPython Notebooks

- This will launch a web browser
- Navigate to your desktop in the browser window
- Open the Python Fundamentals.ipynb File



Select items to perform actions on them.

/ Desktop / Intro to Python Workshop

..

01 - Python Fundamentals.ipynb

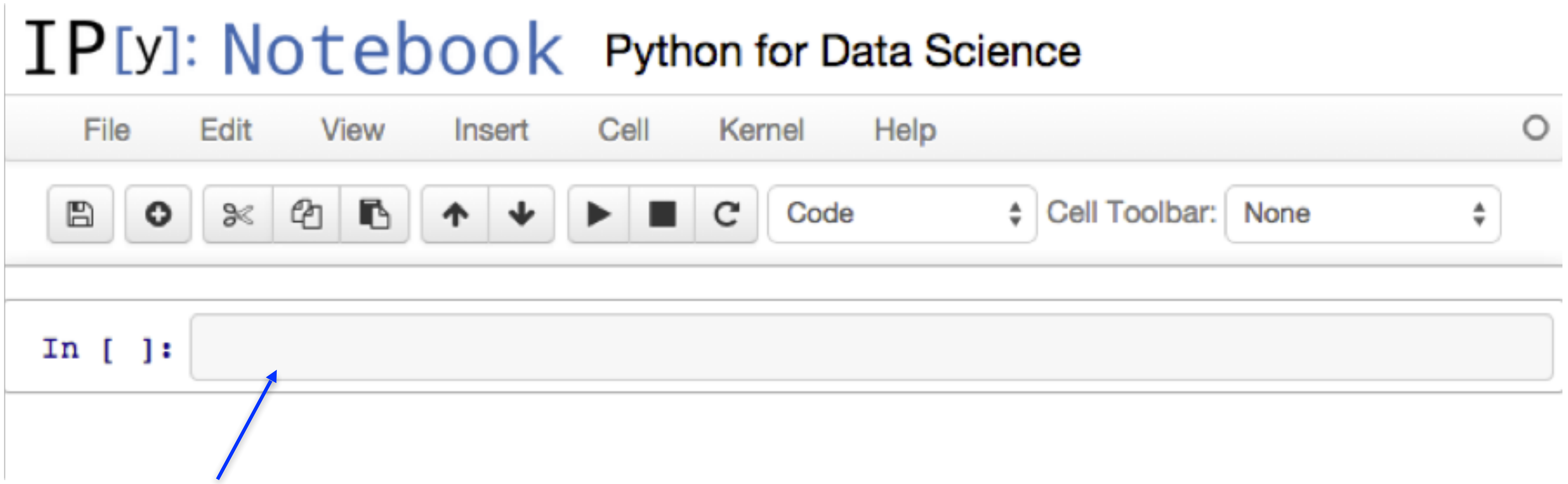
02 - If Statements.ipynb

03 - Lists Tuples Dictionaries.ipynb

04 - For Loops.ipynb

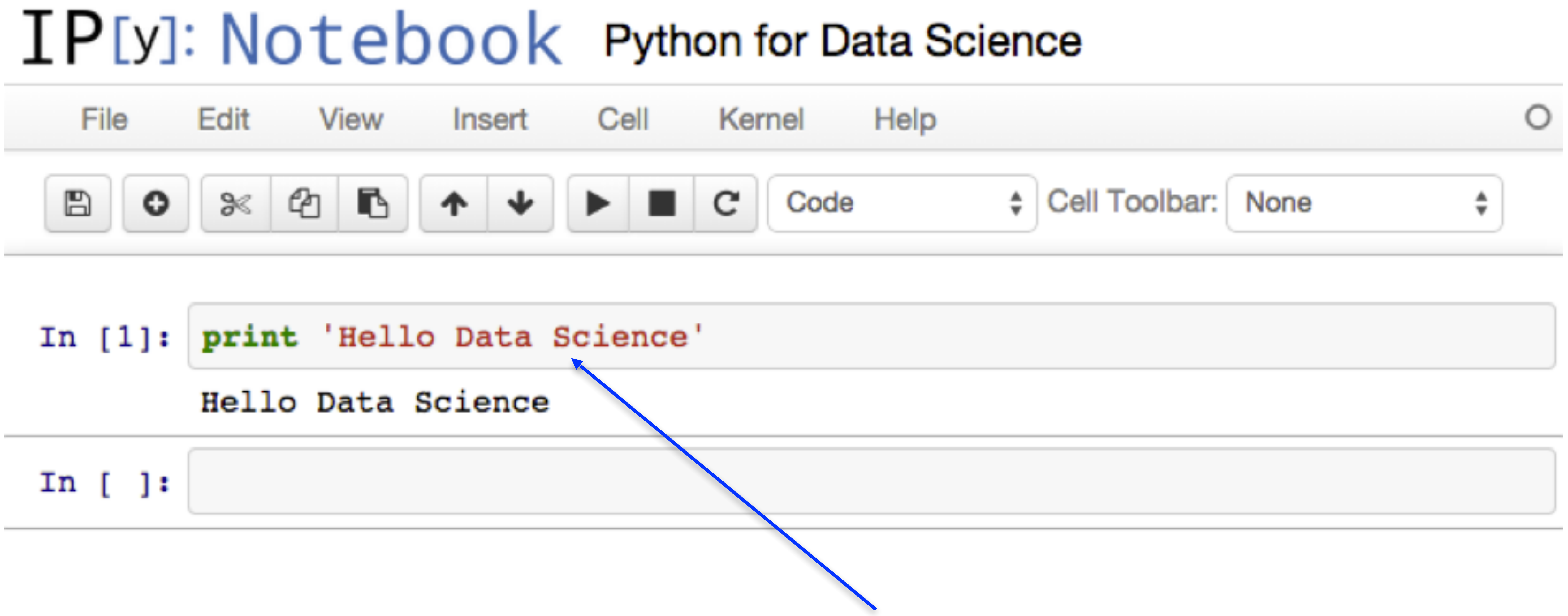
05 - Import Packages and Datetime.ipynb

iPython Notebook Basics



Enter code here

iPython Notebook Basics



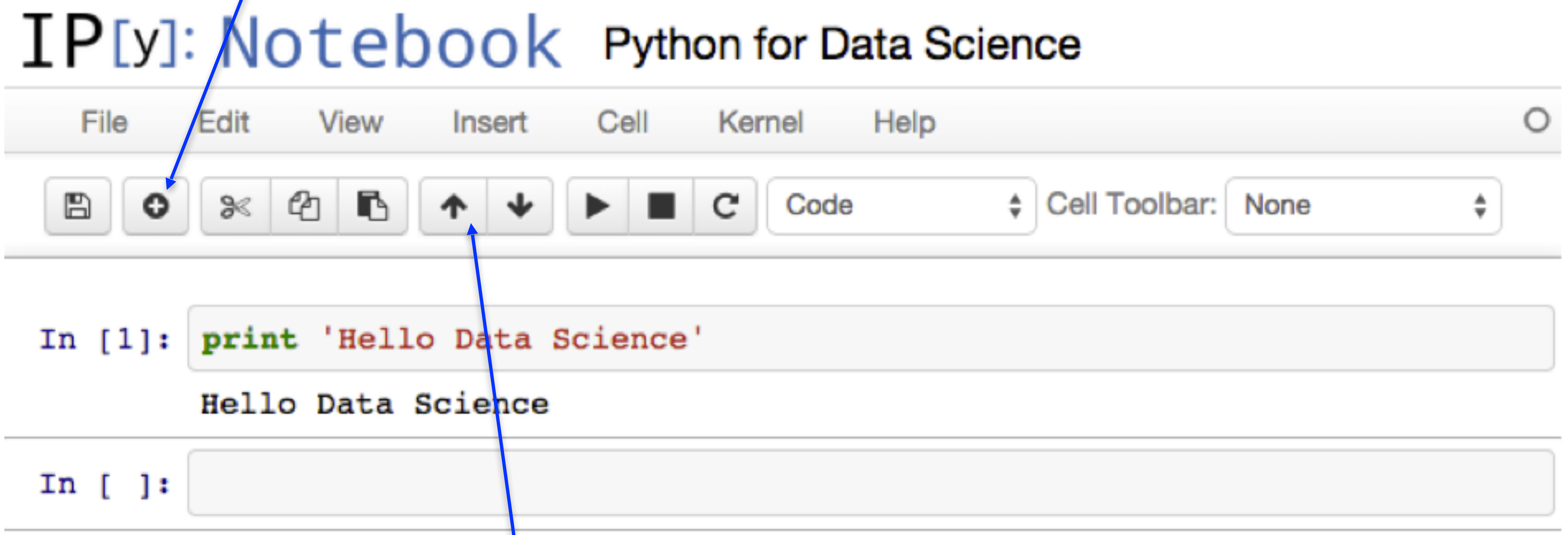
```
In [1]: print 'Hello Data Science'
        Hello Data Science

In [ ]:
```

Shift + Enter runs code and returns results

iPython Notebook Basics

Add more code blocks

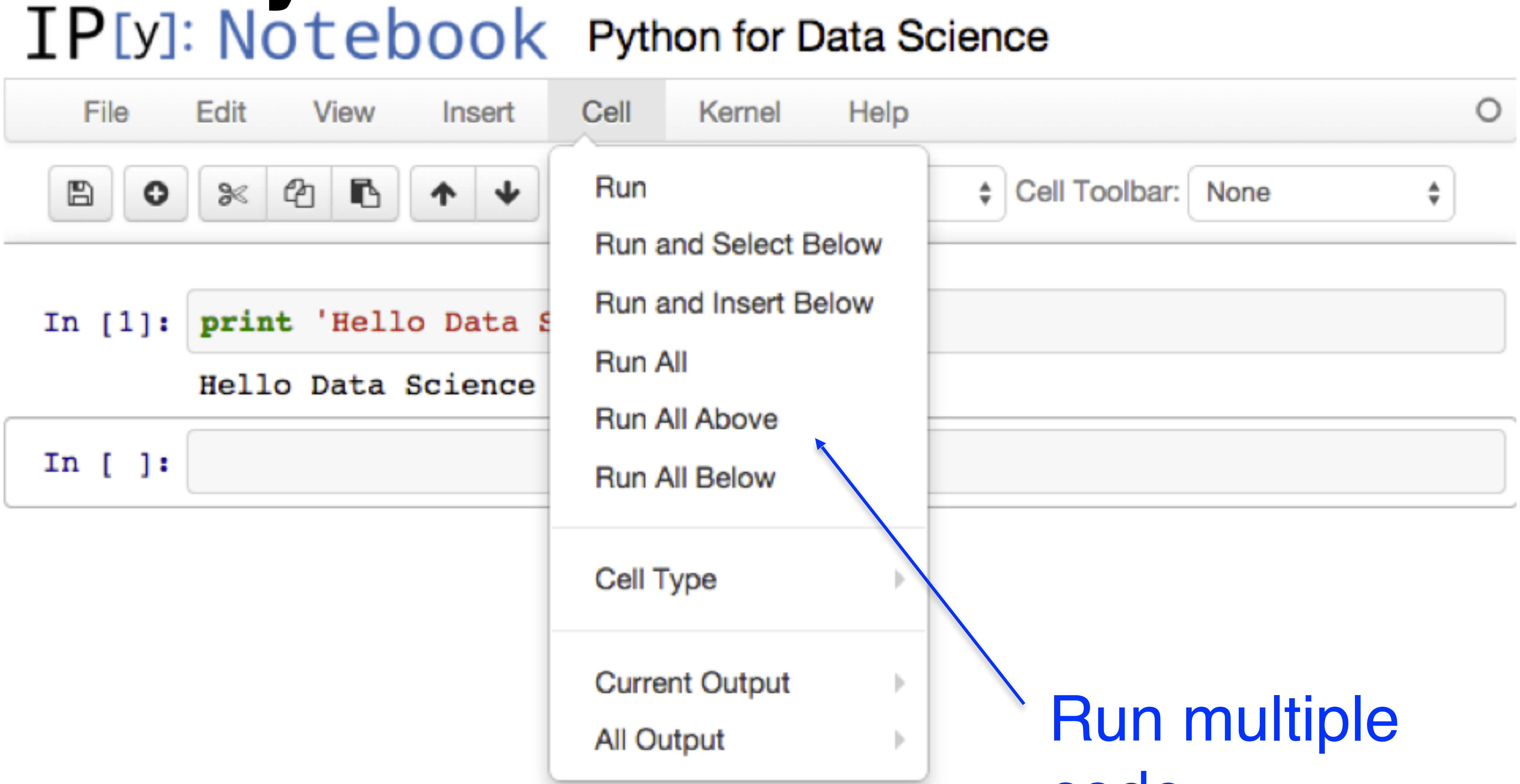


```
In [1]: print 'Hello Data Science'
        Hello Data Science

In [ ]:
```

Re-order code blocks

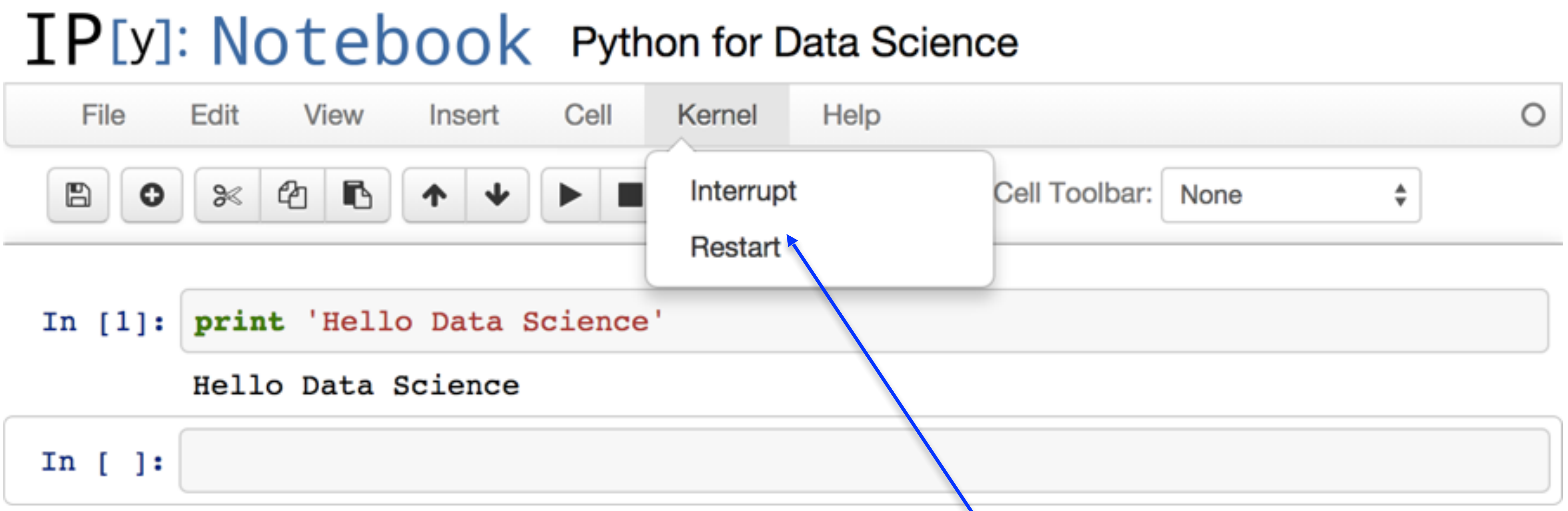
iPython Notebook Basics



```
In [1]: print 'Hello Data S
        Hello Data Science
In [ ]:
```

Run multiple code blocks

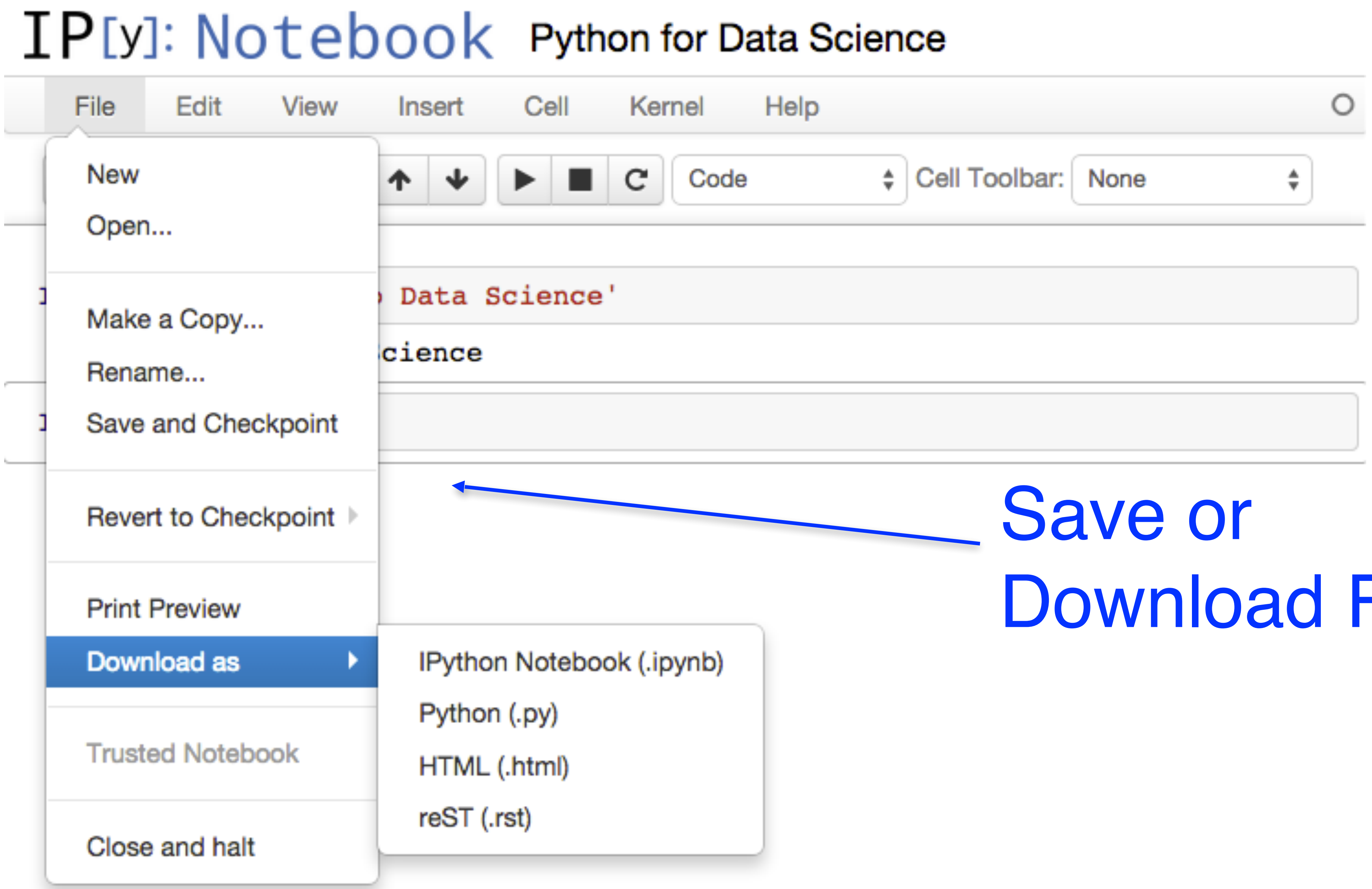
iPython Notebook Basics



```
In [1]: print 'Hello Data Science'
Hello Data Science
In [ ]:
```

Restart Notebook to Clear Memory

iPython Notebook Basics



```
Data Science'
cience
```

Save or Download File

Write Your First Python Code

Type in the first code block:

```
print “Hello Data Science”
```

Press Shift + Enter

Data Types

- **Numeric Types**
  - Integer (whole numbers)
  - Float (includes decimals)
  - Boolean (True/False)

- **Strings**
  - Text
  - Must be in single or double quotes

Python has a function to return data type:
type(<value>)

Try Data Types

```
type(1)
```

```
type(2.5)
```

```
type(True)
```

```
type(‘string’)
```

String Insertion Syntax

Text in quotes

Placeholder for insertion

Command for string insertion

```
“Bunch of text {}”.format(<value>)
```

Value in parentheses

Try Basic String Insertion

```
“My name is {}”.format(‘Craig’)
```

```
name = “Waldo”
“Where in the world is {}”.format(name)
```

Multiple Insertions

Multiple insertions: helpful to put values in the brackets

```
place = “SF”
“{0} is in {1}”.format(name, place)
```

What happens when you change the order of the variables?

Basic Math

Some operators are pretty obvious

```
5 + 5
```

```
3 * 7
```

Basic Math

Some are less intuitive

```
print “Hello ” + “World”
```

```
10 % 4  # modulo
```

```
10 ** 2  # exponent
```

```
1E3 + 1E-3  # exponent base 10
```

Variables

- Variables are objects that hold values
- Name variable using letters, numbers and underscore
- Special characters can't be used for naming variables (e.g., [,*,@)
- Python commands can't also be used as variable names
- Assign values to variables using single =
- You can re-assign values to variables

Assign Values to Variables

Create a few variables

```
x = 10
y = 5
z = 4
```

Try math with variables

```
x - y
```

```
x * y
```

Data Types in Math

Try dividing two integers

```
x / z
```

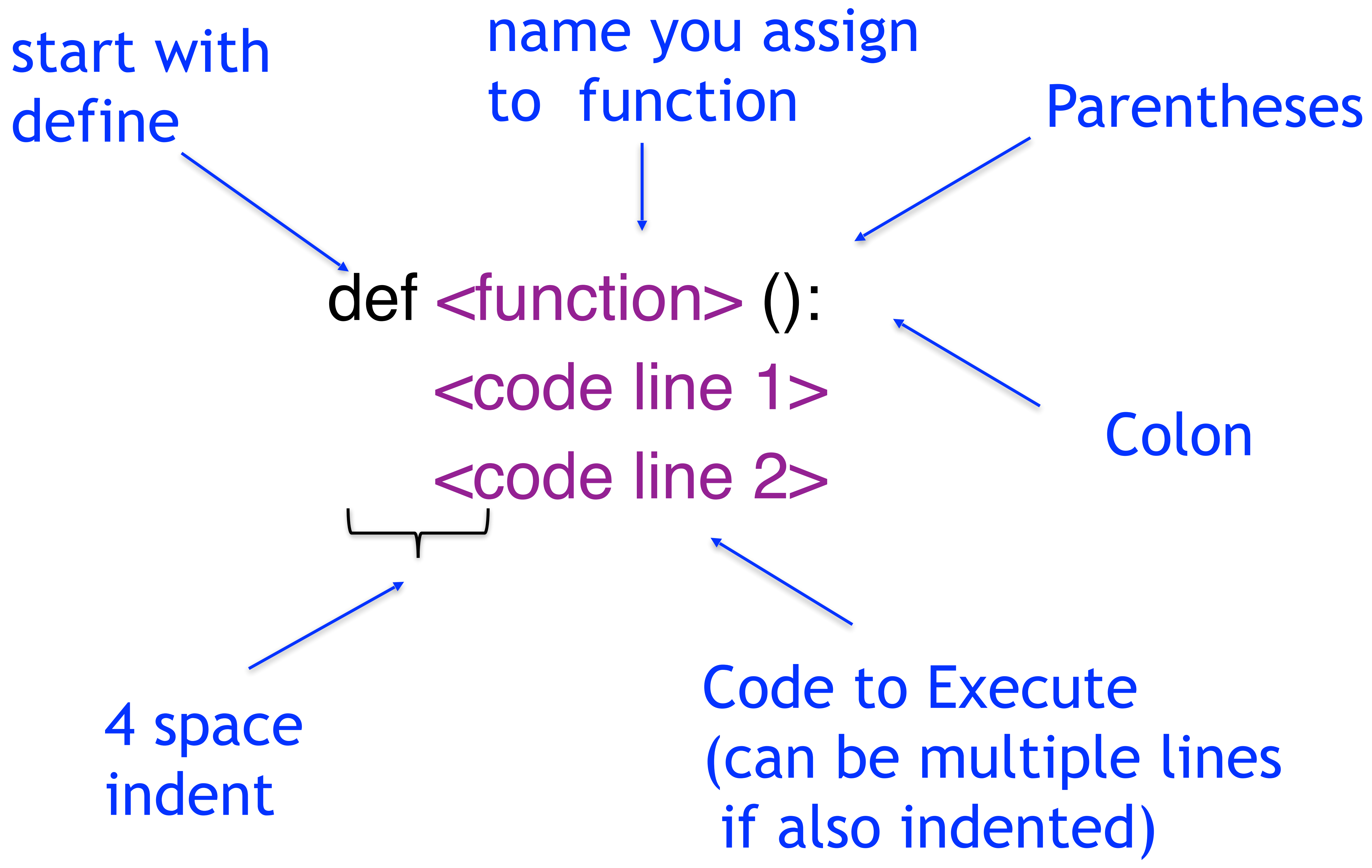
Now try using one float

```
x / 4.0
```

Functions

- Reusable snippets of code
- Define the function once
- Call the function to execute your code as many times as you like
- Can receive inputs and return results

Function Syntax



```
def <function> ():
    <code line 1>
    <code line 2>
```

start with define
name you assign to function
Parentheses
Colon
4 space indent
Code to Execute (can be multiple lines if also indented)

Create a simple function

Write a function

```
def simplestFunction():
    print “I made a function”
```

Call the function

```
simplestFunction()
```

Function with Input

Write a function that requires an input

```
def square(x):
    return x ** 2
```

Call the function

```
square(5)
```

Line Continuation

- Sometimes code gets too long to write on one line
- Python automatically recognizes line continuation in specific cases like commas
- Backslashes (\ ) can be used to continue line of code

Line Continuation

Line continuation with commas

```
numbers = [1, 2, 3,
           4, 5, 6,
           7, 8, 9]
print numbers
```

Backslash for line continuation

```
long_string = 'This is a really, really, really ' \
              + 'long sentence’
print long_string
```

Instructions for Exercises

- Pair programming
  - Using only one computer
  - Take turns typing
  - Collaborate on solutions

- Save Examples for Future Reference
  - Add notes using # Comments

- Error Tracking
  - Create a text file to keep notes on your errors

- Trouble-shooting References
  - Online documentation
  - Stackoverflow / Google

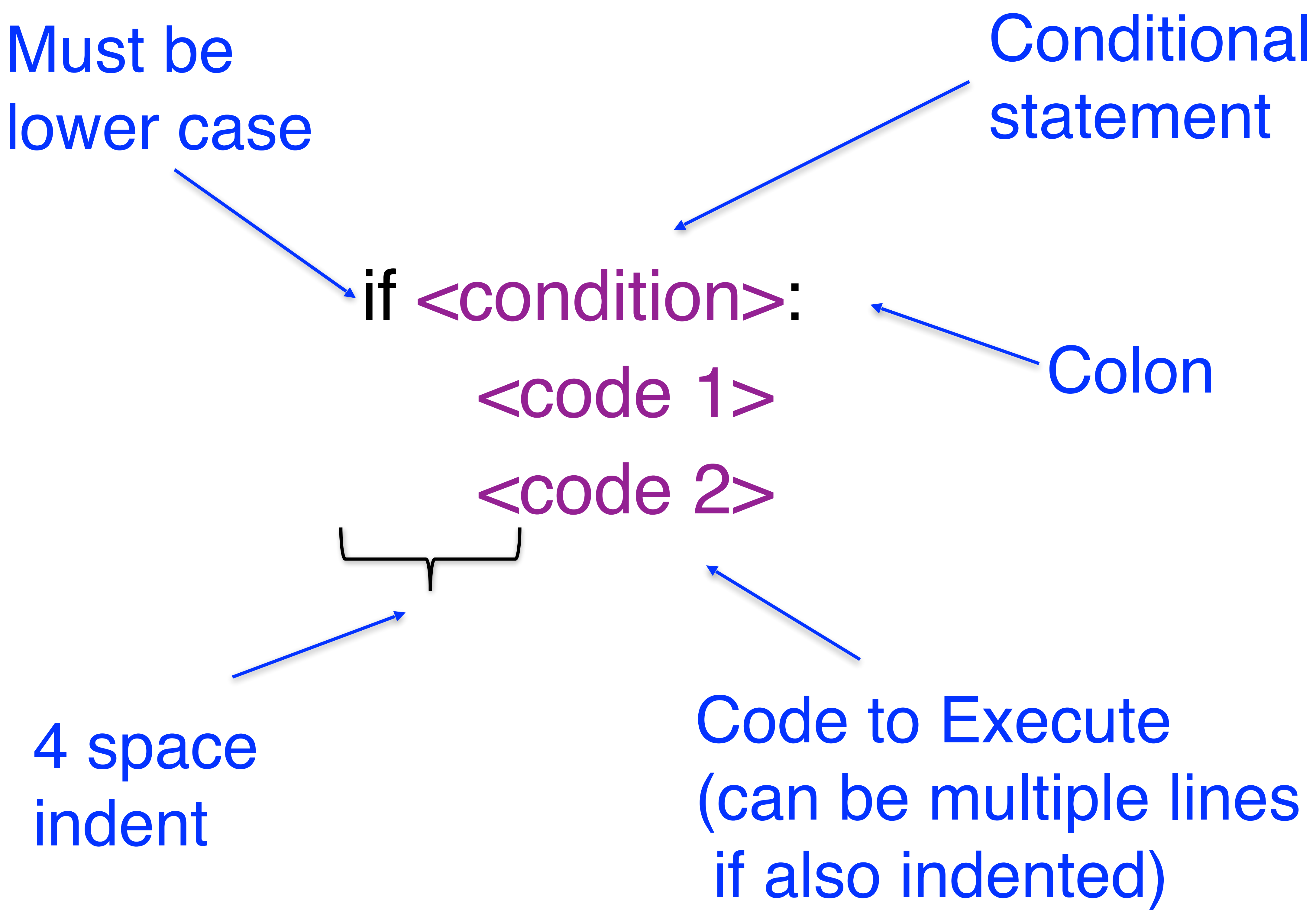
Exercise

1. Create a function that converts Celsius to Fahrenheit. Returns converted value. Results should be accurate to at least one decimal point.
2. Update your function to return a sentence (string type) with the Celsius and Fahrenheit values inserted into the string. ”You input 10 the converted value is …”

If Statements

- Used to execute commands when defined conditions are met
- Contains a conditional statement that has a True/False value
- If statement is True then a series of commands will be executed
- If the statement is False then commands are skipped

If Statements Syntax



```
if <condition>:
    <code 1>
    <code 2>
```

Conditional Statements

a == b	***Equal***
a != b	***Not Equal***
a > b	***Greater Than***
a >= b	***Greater Than or Equal***
a <= b	***Less Than or Equal***

Multiple Conditions

True and True	***= True***	and, & are interchangeable
True & False	***= False***	
True or False	***= True***	or, l are interchangeable
False l False	***= False***	

If Statement

Write a simple if statement

```
x = 3
if x > 0:
    print x
```

Else If Statement

If statement with Else If

```
x = 3
if x > 0:
    print “{} is a positive number”.format(x)
elif x < 0:
    print “{} is a negative number”.format(x)
else:
    print “x equals 0”
```

Exercise

1. Create a function that checks the type of an input and returns True if the input is numeric (float or integer) or a False if it is another data type.
2. Update your temperature function from the Python Fundamentals exercise to return an error message if a string is entered instead of a number

Lunch

Lists, Tuples and Dictionaries

- Python has built-in objects that can hold multiple values
- Can be assigned to variables
- Has built-in methods
- Methods are functions for object

object.method

car.drive

dog.bark

Lists

- Lists are ordered data containers
- Lists are defined with square brackets [ ]
- They can contain any type of objects
  - Mix of data types (e.g., integer, string, float)
  - Lists can even contain other lists
- List are mutable (you can edit them)
- Uses index to reference items in lists
- Lists can be empty
- Very important to understand as they feed directly into data analysis (image excel as a bunch of lists of data)

List Basics

Use brackets to define list

```
x = [1, ‘b’, True]
```

Use index position to reference items

```
print x[2]
```

Reassign values in a list

```
x[1] = ‘a’
print x
```

Indexing Lists

Create list of lists

```
a = [[1,2,3], 4, 5]
```

Use multiple indexes for lists within lists

```
print a[0][1]
```

Index from the end of the list

```
print a[-1]
```

Appending and Indexing

Append an item to a list

```
a.append(‘one more item’)
print a
```

Reference multiple items in a list

```
print a[2:4]
```

Open ended indexes go to the ends of lists

```
print a[:3]
```

Tuples

- Tuples are similar to lists
- Tuples are defined using parentheses ( )
- Only difference is that tuples are immutable (you can’t change them)
- Tuples with single value must have a comma (1,)

Tuple Basics

Use parentheses to define tuple

```
y = (1, ‘a’, 2.5)
print y
```

Use index position to reference items

```
print y[0]
```

Try reassigning values in a tuple

```
y[0] = 2
```

Dictionaries

- Dictionaries are collections of key-value pairs
- Dictionaries are indicated by curly braces { }
- Values are looked up by key
- Dictionaries are ***<u>unordered</u>***

Dictionary Basics

Create a dictionary

```
info = {'name': 'Bob', 'age': 54, 'kids': ['Henry', ‘Phil’]}
print info
```

Use key to reference a value

```
print info[‘name’]
```

Change the value for a key-pair

```
info[‘age’] = 55
print info
```

Dictionary Methods

View all keys

```
info.keys()
```

View all values

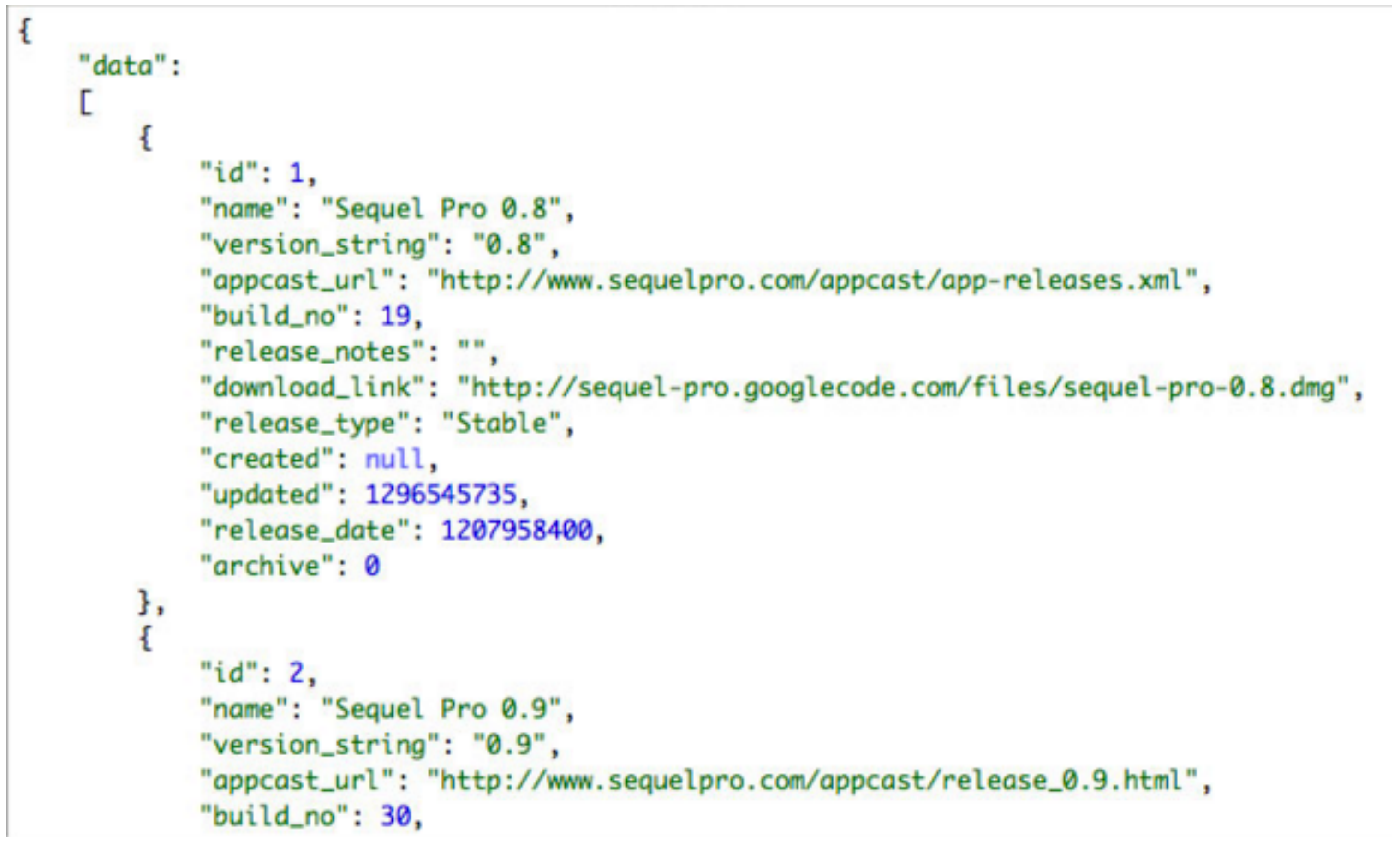
```
info.values()
```

Check if a key exists

```
info.has_key(‘age’)
```

JSON!

- Understanding dictionaries is important because most web data is transferred in JSON format which works like a dictionary in python



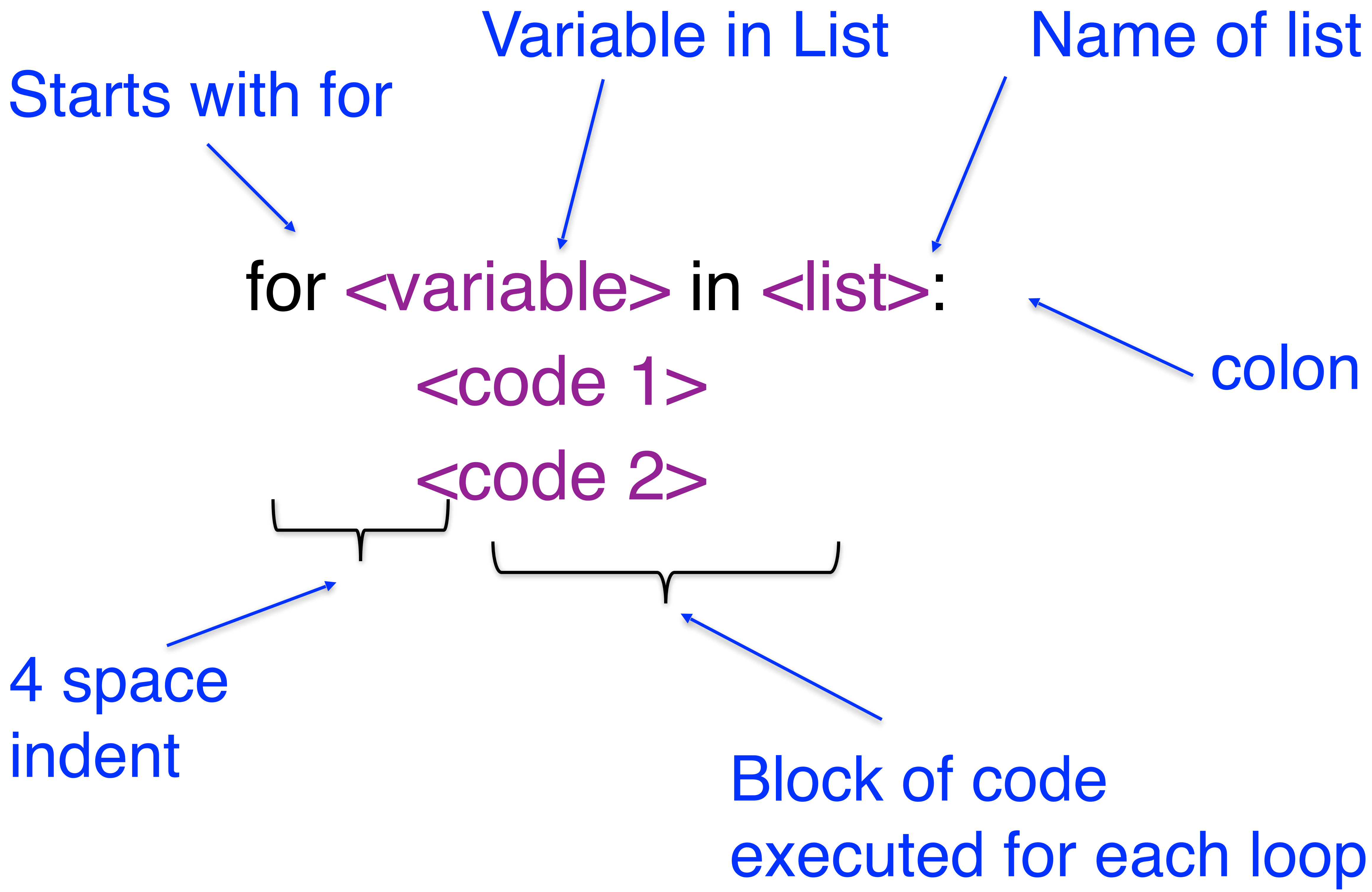
```
{
    "data":
    [
        {
            "id": 1,
            "name": "Sequel Pro 0.8",
            "version_string": "0.8",
            "appcast_url": "http://www.sequelpro.com/appcast/app-releases.xml",
            "build_no": 19,
            "release_notes": "",
            "download_link": "http://sequel-pro.googlecode.com/files/sequel-pro-0.8.dmg",
            "release_type": "Stable",
            "created": null,
            "updated": 1296545735,
            "release_date": 1207958400,
            "archive": 0
        },
        {
            "id": 2,
            "name": "Sequel Pro 0.9",
            "version_string": "0.9",
            "appcast_url": "http://www.sequelpro.com/appcast/release_0.9.html",
            "build_no": 30,
```

Break

For Loops

- Iterates through multiple values
- Commonly used to process values in a list
- Loop of code is executed for each item

For Loop Syntax



```
for <variable> in <list>:
    <code 1>
    <code 2>
```

Functions Used with For

range(<integer>)

- Creates list of integers
- Starts with zero and each subsequent value is incremented by 1
- Returns list with length = input integer
- Last item in list is input -1 since list starts with zero

Functions Used with For

len(<object>)

- Checks the length of your object
- Useful as an input to your range() function
- For example, with a list:

  ```
  range(len(mylist))
  ```
- Or with a dictionary:

  ```
  range(len(mydict.keys()))
  ```

Basic For Loops

Create basic for loop

```
for x in [1,2,3]:
    print x
```

Create a for loop with range

```
for x in range(10):
    print x
```

For Loops with Empty List

Capture the all the results of a for loop

```
results = []
for x in [1,2,3]:
    squared = x ** 2
    results.append(squared)

print results
```

You can break out of a loop by using the keyword “break”

Exercise

1. Create a function that finds the maximum integer in a list of integers.
2. Update your function to return a sentence (string type) that states “The maximum value of the list [1,4,2,3] is 4”

Bonus:

Add error handling to ensure all items in the list are indeed integers.

Python Packages

- Data analytics packages are what make python so powerful
- Packages are just files/folders of python code
- Importing packages allow you to use the functions from these files
- Most packages have online documentation and code examples

Common Packages for Data Science

Package	Usage
numpy	Scientific computing
pandas	Data slicing and manipulation
datetime	Manage date and time formats
matplotlib	Creating charts and graphs
scikit-learn	Machine learning
statsmodels	Statistics

Lets Go Through Pandas

Coding Best Practices

PEP-8 Style Guide

- https://www.python.org/dev/peps/pep-0008/
- maximum line length of 79 characters
- indentation
- line continuation
- commenting

Hard to Use at First But Review Once a Month and You'll Incrementally Improve

Debugging Tips

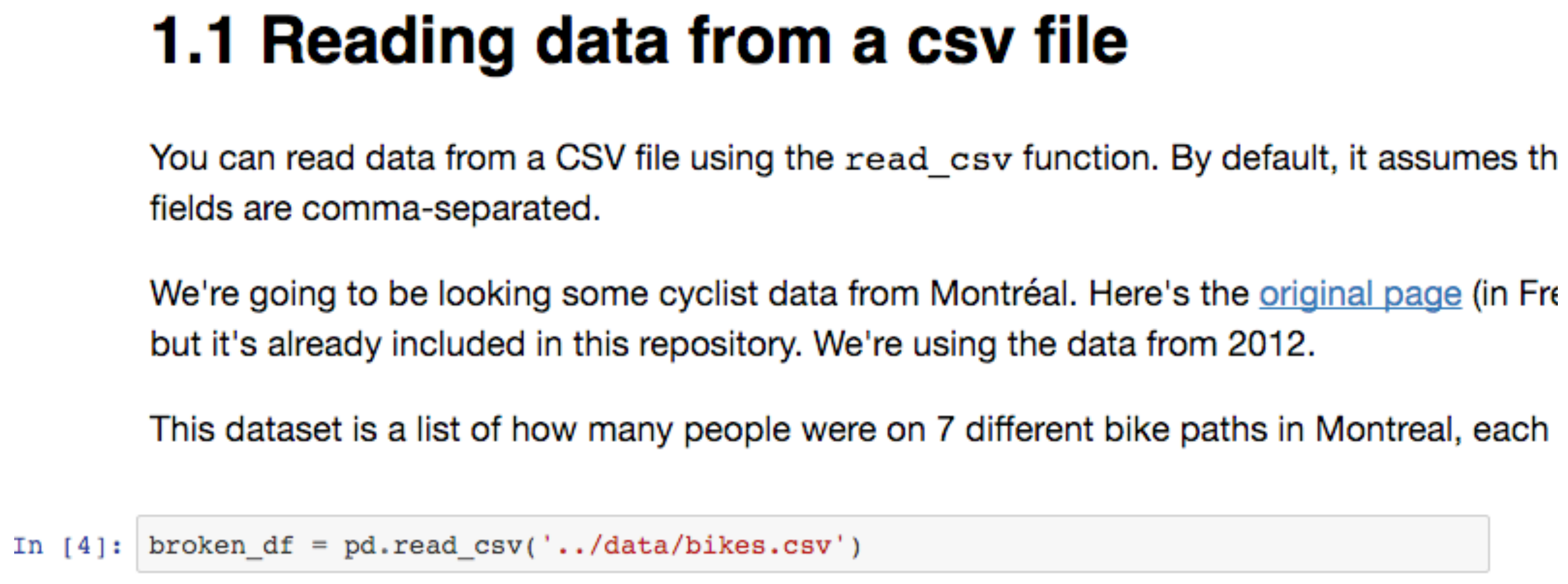
- Test early and often
- print command is your best friend
- Identify common mistakes and use as checklist for debugging
- Use google and stackoverflow
- Check documentation
- Ask for help

Getting Started with Pandas

Pandas Cookbook

https://github.com/jvns/pandas-cookbook

Copy of this zip file on dropbox



1.1 Reading data from a csv file

You can read data from a CSV file using the `read_csv` function. By default, it assumes th
fields are comma-separated.

We're going to be looking some cyclist data from Montréal. Here's the original page (in Fre
but it's already included in this repository. We're using the data from 2012.

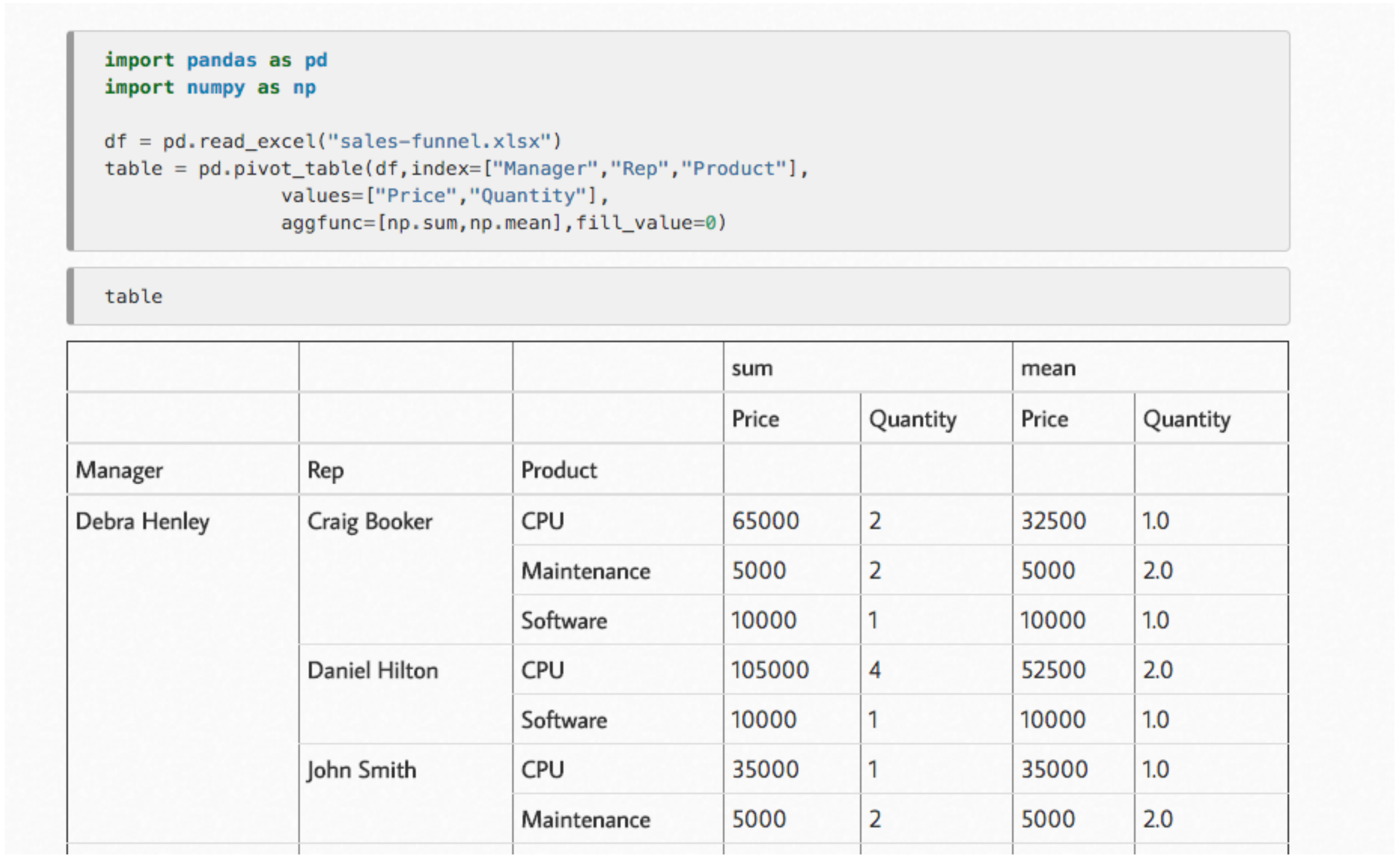
This dataset is a list of how many people were on 7 different bike paths in Montreal, each

```
In [4]: broken_df = pd.read_csv('../data/bikes.csv')
```

Getting Started with Pandas

"Generating Excel Reports from Pandas":

http://pbpython.com/pandas-pivot-report.html



```
import pandas as pd
import numpy as np

df = pd.read_excel("sales-funnel.xlsx")
table = pd.pivot_table(df,index=["Manager","Rep","Product"],
               values=["Price","Quantity"],
               aggfunc=[np.sum,np.mean],fill_value=0)
```

```
table
```

			sum		mean	
			Price	Quantity	Price	Quantity
Manager	Rep	Product				
Debra Henley	Craig Booker	CPU	65000	2	32500	1.0
		Maintenance	5000	2	5000	2.0
		Software	10000	1	10000	1.0
	Daniel Hilton	CPU	105000	4	52500	2.0
		Software	10000	1	10000	1.0
	John Smith	CPU	35000	1	35000	1.0
		Maintenance	5000	2	5000	2.0

Next Steps

- Practice, Practice, Practice
    - Find a fun project
    - Get a pair programming buddy
    - Don’t Be Afraid to Ask for Help
- Join a Python Meetup
- Keep Learning
    - Pycon videos on YouTube
- Keep in Touch

Thanks!

- Post course survey will be sent out soon. Please complete it! It helps me get better.