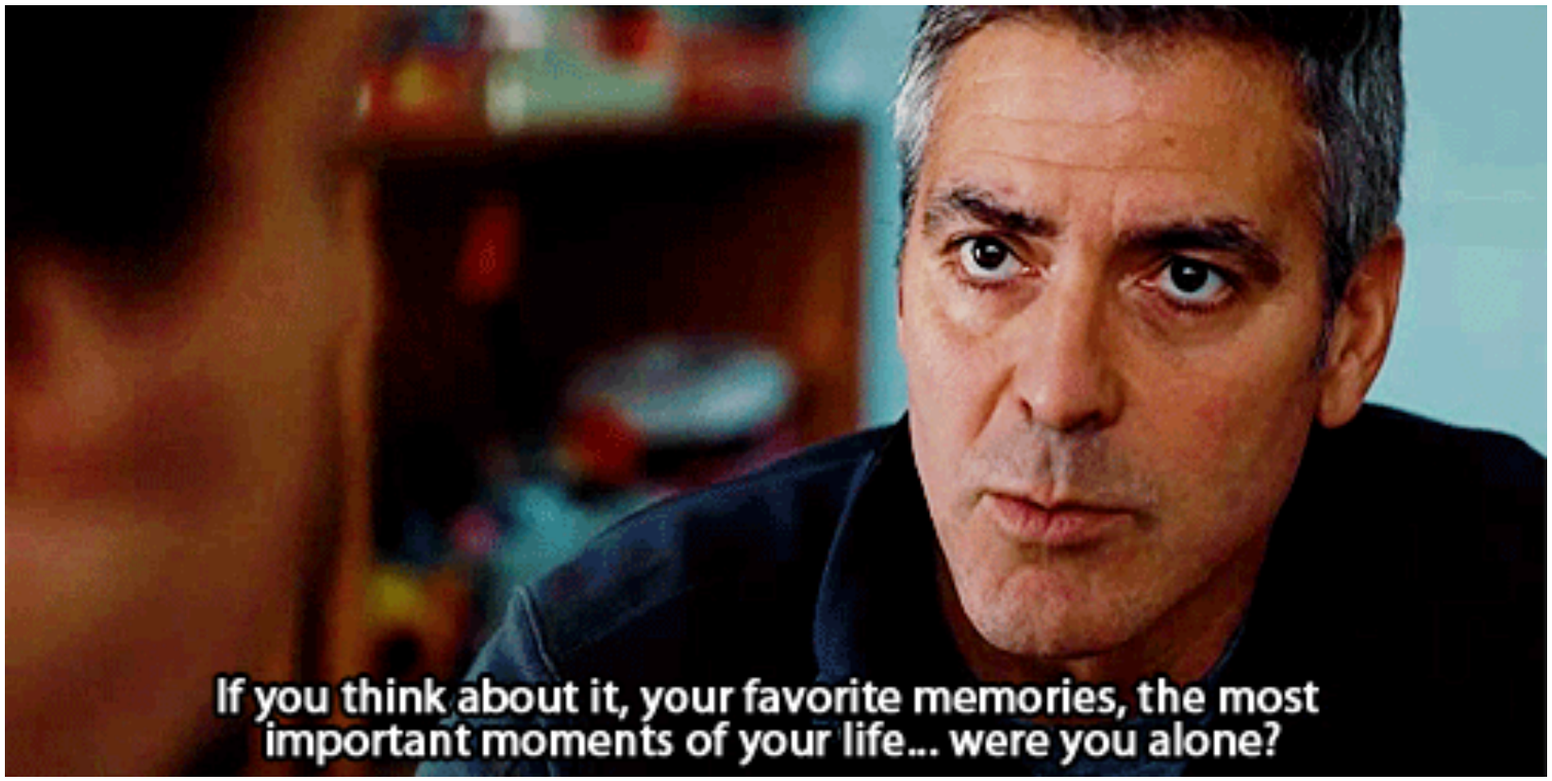
GENERAL ASSEMBLY

INTRO TO DATA SCIENCE
LECTURE 1: Python 101

Neo Ellison
Data Science Partner, Fix It With Code
Lead Data Scientist, Enertiv

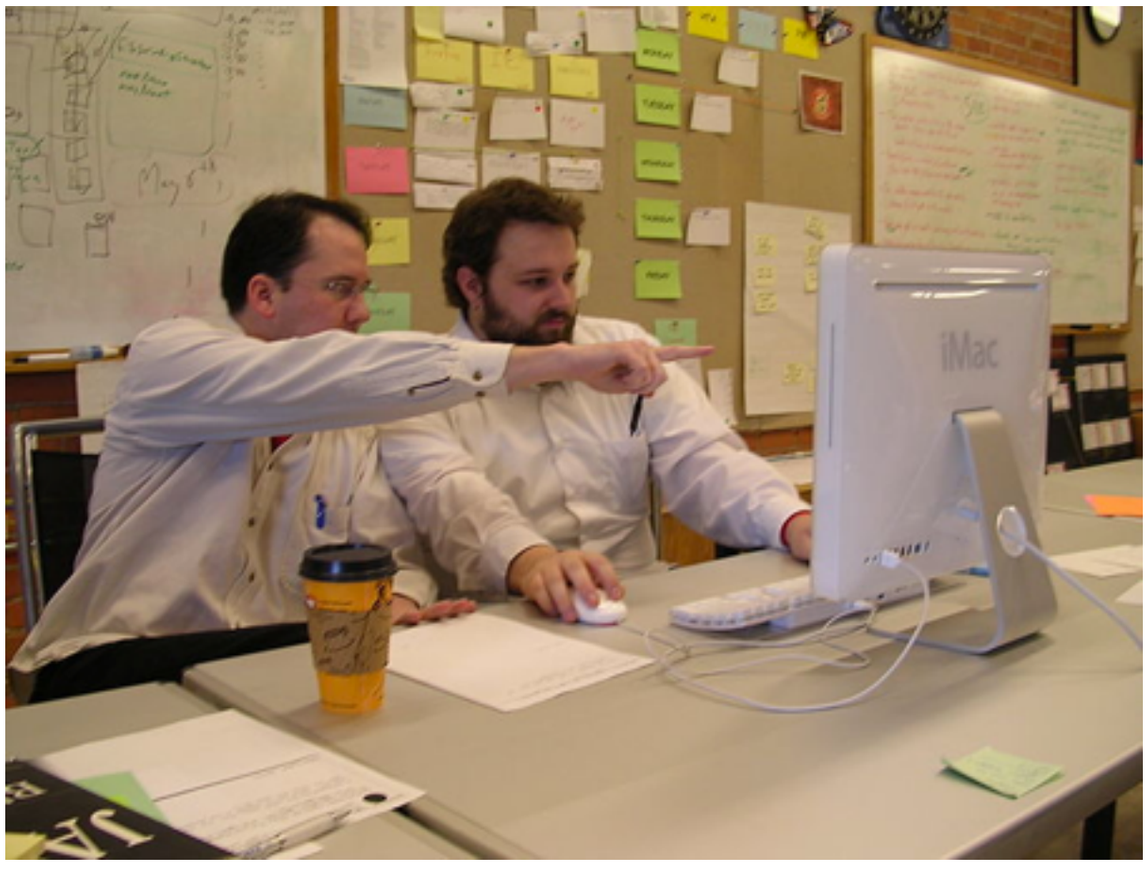
Think of the Best Times of your Life…

Stolen From



- Ryan Bingham (George Clooney)
Up in the Air (2009)

Pair Programming



Pair programming (aka peer programming)

An agile software development technique in which two programmers work as a pair together one workstation. One, the driver, writes code while the other, the observer, pointer or navigator,[1] reviews each line of code as it is typed in. The two programmers switch roles frequently.

AGENDA

I. Command Line Basics
II. Python Scripting
III. iPython Intuition
IV. Python 101
    - Types & Assignment
    - Datatypes
    - Operators
    - Control Flow
    - Functions
    - Classes
    - Libraries

I. COMMAND LINE BASICS

COMMAND LINE BASICS

Lets get comfortable with the command line

COMMAND LINE BASICS

Need to Know Commands

- **cd**
- **ls/dir**
- **pwd**
- **mkdir**
- **rm**
- **mv**
- **cat**
- **cp**

COMMAND LINE BASICS

Need to Know Commands

- **cd** Moving around $ cd folder_name, cd ../
- **ls/dir** See what is here $ ls, > dir
- **pwd** See where you are $ pwd
- **mkdir** Creates a directory $ mkdir new_dir_name
- **rm** Deletes a file (does not go to trash folder) $ rm file.dead
- **mv** Moves and renames a file $ mv old_folder/old.txt new_folder/new.txt
- **cp** Copies a file to a new location $ cp file.txt file_copy.txt

3 minutes:
Using these commands create a directory where your Data Science work will live

II. PYTHON SCRIPTING

Python Scripting

Terminology:

- Variable
- Function
- Program
- Library
- Script

PYTHON SCRIPTING

Terminology:

- Variable - Anything, a place holder for some data which can be defined and redefined
- Function - instructions for the computer to apply to an input to get an output
- Program - a collection of functions and variables, think of it like a .py file
- Library - is a collection of programs generally solving a common set of problems
- Script - a chain of programs and libraries that perform an action on the computer when called.

My First Script

3 minutes:
With your partner create a new file in your data science directory using sublime text:

helloworld.py

Enter the following text:

```
print “Hello World”
```

In the terminal/command line run, what happens?:

```
$ python helloworld.py
```

II. IPYTHON INTUITION

IPYTHON INTUITION

What is iPython Notebook?

IPYTHON INTUITION

What is iPython Notebook?

How do iPython Notebook files work?

JSON

Json (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write as well as is easy for machines to parse and generate.

```json
{
  "orderID": 12345,
  "shopperName": "John Smith",
  "shopperEmail": "johnsmith@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "SuperWidget",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "WonderWidget",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

CONTROLLING THE NOTEBOOK

The key actions:

- Creating a new cell
- Moving cells
- Running a cell
- Deleting cells
- Cell types

BUILDING AN IPYTHON NOTEBOOK

10 minutes:

With your partner create a new file called python_101.ipynb in the DS folders.

Follow the link in Slack from the course git to get the raw text for today's lab.

Copy that json into python_101.ipynb and save it, and open it up. Repeat until everything has this ready to go.

If you have extra time be sure to practice the key iPython Notebook actions:

- Creating a new cell
- Moving cells
- Running a cell
- Deleting cells
- Cell types

II. PYTHON 101 - ASSIGNMENT

Types & Assignment

Data types

- none
- strings
- numeric
- tuples
- lists
- sets
- dictionaries

Variable Assignment

```
x = "hello"
x = x + " world"
y = [1, 2, 3, 4]
y[0] = 7
z = {"a": 1, "b": 2}
z["a"] = 3.14159
```

II. PYTHON 101 - OPERATORS

OPERATORS

Boolean

Operation	Result
`x or y`	if *x* is false, then *y*, else *x*
`x and y`	if *x* is false, then *x*, else *y*
`not x`	if *x* is false, then `True`, else `False`

OPERATORS

Comparison

Operation	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal
is	object identity
is not	negated object identity

OPERATORS

Operation	Result
`x + y`	sum of *x* and *y*
`x - y`	difference of *x* and *y*
`x * y`	product of *x* and *y*
`x / y`	quotient of *x* and *y*
`x // y`	(floored) quotient of *x* and *y*
`x % y`	remainder of `x / y`
`-x`	*x* negated
`+x`	*x* unchanged
`abs(x)`	absolute value or magnitude of *x*
`int(x)`	*x* converted to integer
`long(x)`	*x* converted to long integer
`float(x)`	*x* converted to floating point
`complex(re,im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.
`c.conjugate()`	conjugate of the complex number *c*. (Identity on real numbers)
`divmod(x, y)`	the pair `(x // y, x % y)`
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*
`math.trunc(x)`	*x* truncated to Integral
`round(x[, n])`	*x* rounded to n digits, rounding ties away from zero. If n is omitted, it defaults to 0.
`math.floor(x)`	the greatest integral float <= *x*
`math.ceil(x)`	the least integral float >= *x*

Numeric

OPERATORS

Sequence

Operation	Result
`x in s`	`True` if an item of *s* is equal to *x*, else `False`
`x not in s`	`False` if an item of *s* is equal to *x*, else `True`
`s + t`	the concatenation of *s* and *t*
`s * n, n * s`	*n* shallow copies of *s* concatenated
`s[i]`	*i*th item of *s*, origin 0
`s[i:j]`	slice of *s* from *i* to *j*
`s[i:j:k]`	slice of *s* from *i* to *j* with step *k*
`len(s)`	length of *s*
`min(s)`	smallest item of *s*
`max(s)`	largest item of *s*
`s.index(x)`	index of the first occurrence of *x* in *s*
`s.count(x)`	total number of occurrences of *x* in *s*

II. CONTROL FLOW

CONTROL FLOW

If Statements

```
if condition:
    then
elif condition:
    then
else:
    then
```

CONTROL FLOW

For Loops

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

CONTROL FLOW

For Loops

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

```
for_list = ["a", "b", "c", "d"]
for x in for_list:
    print x
```

CONTROL FLOW

For Loops

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

```
for_list = ["a", "b", "c", "d"]
for x in for_list:
    print x
```

```
for_list = ["a", "b", "c", "d"]
for i, element in enumerate(for_list):
    print i, element
```

II. FUNCTIONS

FUNCTIONS

```
def hi_mom(message="Hi Mom"):
    """A good function always has a doc string"""
    return message

hi_mom()
```

Terminology:

- Function Name
- Arguments
- Default Value
- Doc String
- Return

II. CLASSES & METHODS

CLASSES

```
class Car():
    have_wheels = True
    def __init__(self, model='Ford'):
        self.model = model
        self.running = False
    def start(self):
        if self.running != True:
            print 'The car started!'
            self.running = True
        else:
            print 'The car is already running!'
    def stop(self):
        if self.running == True:
            print 'The car stopped!'
            self.running = False
        else:
            print 'The car was not running!'

Car.have_wheels
ford = Car()
nissan = Car(model = 'Nissan')
ford.running
ford.start()
ford.running
nissan.running
nissan.stop()
```

Terminology:

- Instance Method
- Instance Variable
- Instantiation
- Inheritance
- Class Variable
- State