

```
>>> print '''  
INSEAD Python  
Bootcamp –  
Web Scraping  
'''
```

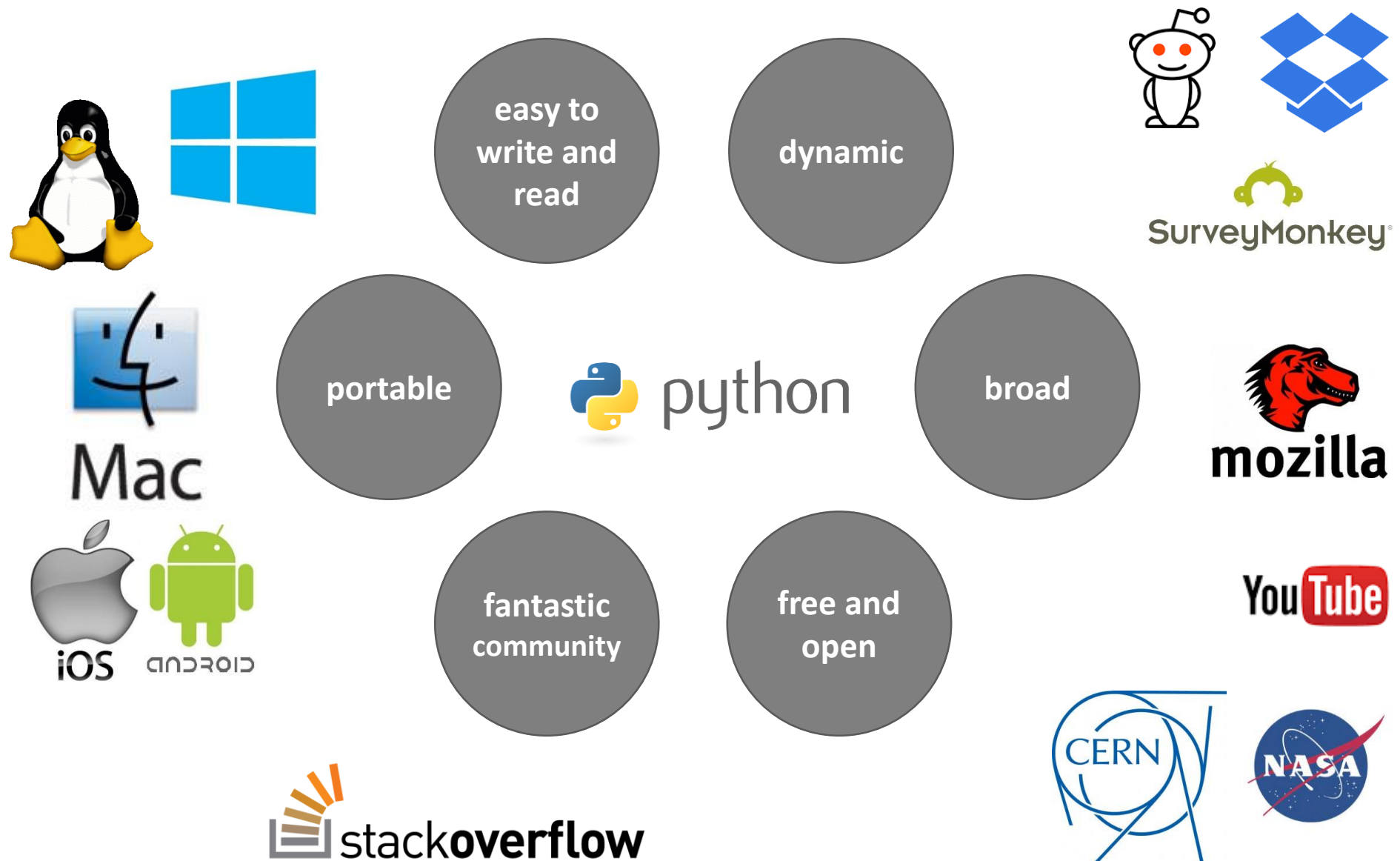


What is Python (hint: it is not about the snake)

- Created in the late 1980's by Guido van Rossum
- It is a general-purpose, high level programming language
- The name comes not from a snake but the British comedy group “Monty Python’s Flying Circus”

```
brian = “Hello life” # corresponds to “The Life of Brian” movie
```

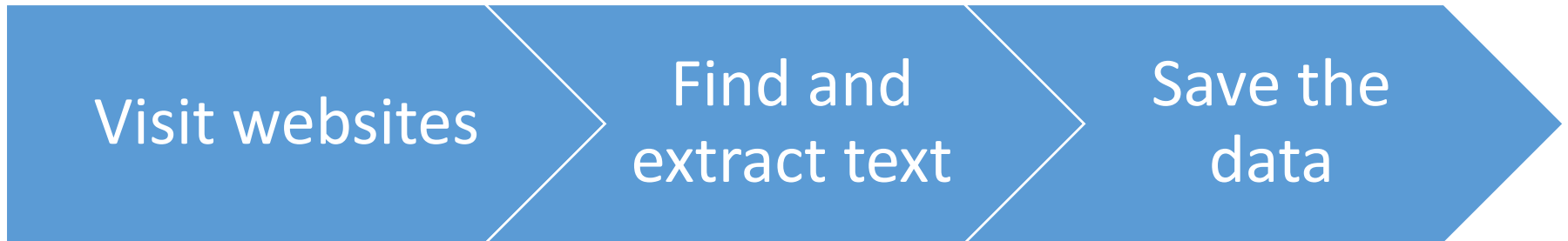
Why Python?



Overview of the workshop

Objective: Learn how to do a simple web crawler and collect data

We will build a database with network links between scholars based on data from the AOM 2014 Annual Meeting



Using Anaconda Python environment - Spyder

Start a new project

Save current project

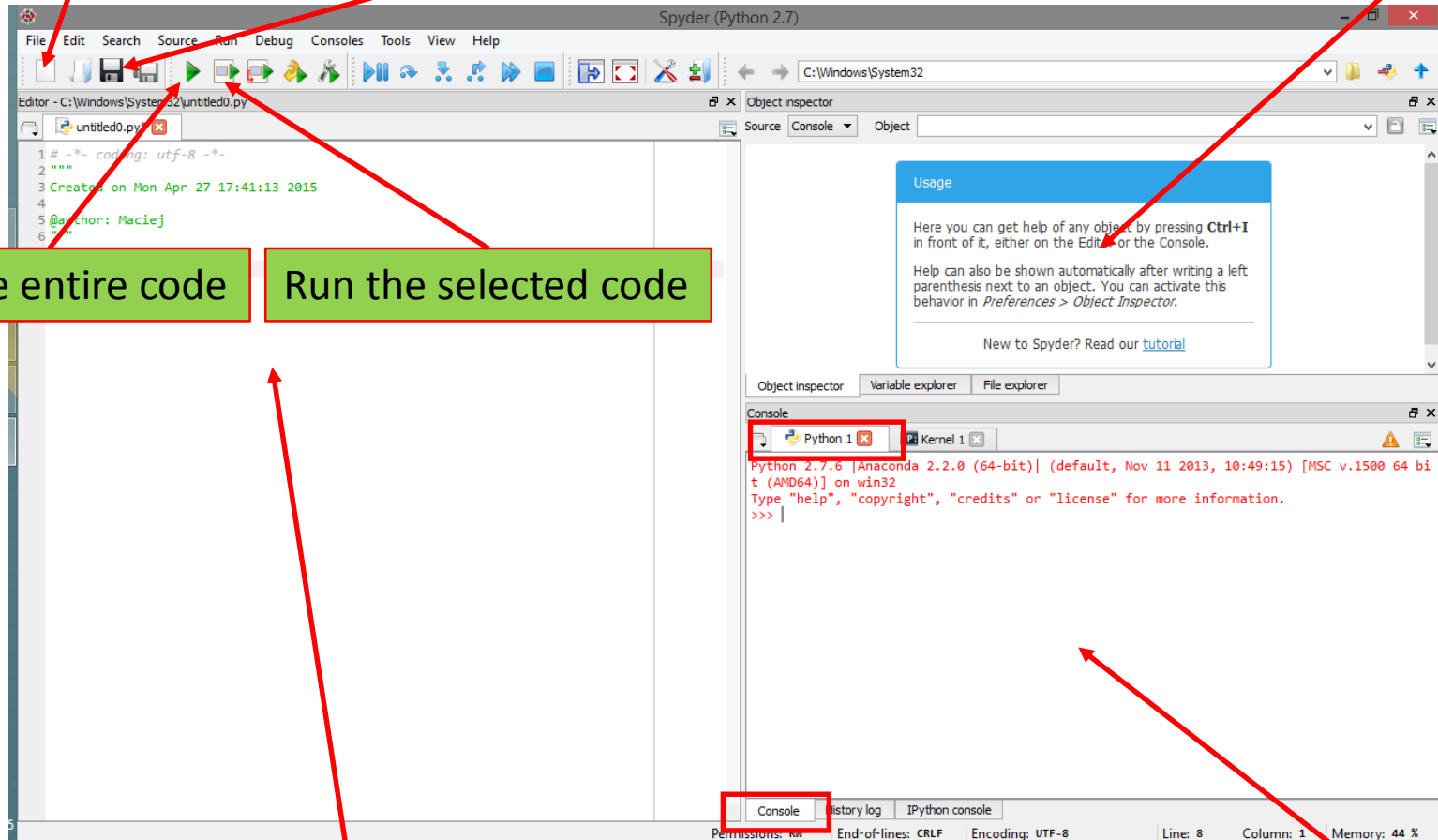
information on objects
press CNTR + I

Run the entire code

Run the selected code

This is where you write your code

Here you see your
code executed



Let's try some code

Have computer display “Hello Brian”

```
print “Hello Brian”
```

Add two numbers and divide them by another number

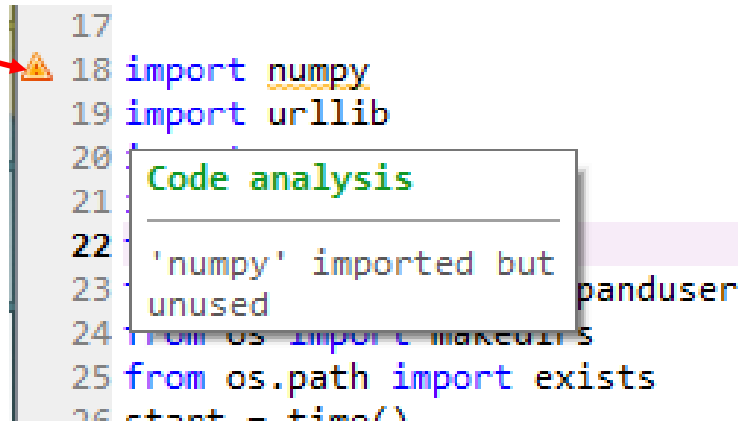
```
(34 + 45)/4.0
```

Print integers from 0 to 9 in the same line

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

Spyder and programming

linting (error analysis)



```
17
18 import numpy
19 import urllib
20
21 Code analysis
22 'numpy' imported but
23 unused
24 from os import mkdir
25 from os.path import exists
26 start = time()
```

A few coding tips

1. Comment your code as if you were sharing it with someone else (you will forget very quickly what you have done and why)
2. Keep a journal of changes you make to your code (either within the file itself or in a separate file). You can also use git, but that is for more advanced projects.
3. Try not to write lines longer than 80 characters (74 is the most common). If you need to you can use brackets to wrap a long line (you will see some examples shortly)

But before we begin – a little refresher

Warm-up exercise

Using what you've learned during the Codecademy course, create **a script which prints the first 100 prime numbers**.

As a reminder, a prime number is a natural number higher than 1 that is only divisible by 1 and itself – for example 2, 3, 5 are the three first prime numbers.

Hint 1.

Think about the particular steps that your computer needs to perform. In short, you want to iterate over integers starting with 2, check whether a given integer is a prime number and record it if it is. Then keep doing this until you have 100 of them.

Hint 2.

You may consider using `while` loop to have the code run until some condition is met and `for` loop to

But before we begin – a little refresher

An example of the code may look like this

```
11 primes = [] # we set up an empty list to collect results
12 x = 2 # we start with the first integer larger than 1
13 while len(primes) < 100: # the loop will run until we have 100 prime numbers
14     prime = True # we assume the number is prime
15     for y in range(2, int(x**0.5) + 1): # checking if we can find a divisor y
16         if x % y == 0: # if there is a divisor, i.e. the number is not prime
17             prime = False # since we found a number that is a divisor of x
18     if prime is True: # we didn't find a divisor of x, meaning it is a prime
19         primes.append(x) # we write down the prime
20     x = x + 1 # we move one integer up
21 print primes # finally we can print the list of 100 primes
```

Output

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109,
113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179,
181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241,
251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313,
317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461,
463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541]
```

Plan for today

1

Have Python visit AOM 2014 session website

2

Extract text from the website

3

Find names of people participating in the session

4

Record the names in a list

5

Iterate over all sessions and add the names to the list of lists

6

Save the list of lists as a CSV file

1 Open an url to a given AOM 2014 session

First we need to find an url of some AOM session website

```
http://program.aom.org/2014
```

Let's search for the keyword "INSEAD"

```
http://program.aom.org/2014/submission.asp?mode=showsession&
SessionID=130&print=true
```

Time to send Python to this location.

```
import urllib # a standard Python module to handle urls
myurl = ('website_address')
urlopen = urllib.urlopen(myurl)
urlopen
```

This doesn't look too impressive

Hint – wrapping lines that are longer than 80 characters

```
Longstring = ('My name is too long '
              'to fit one line')
```

Extract text from the website

Time to have Python read the text. Type the following command and inspect the outcome

```
_page = urlopen.read()
```

_page

**Ouch. That is hard to read.
Can we do better?**

[illegible]

With Python there are always several ways to accomplish the same task. Let's try this:

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(_page)
```

soup

Chair: Xiaowei Luo; INSEAD;

Chair: Hart E. Posen; U. of Wisconsin, Madison;

Participant: John Joseph; Duke U.;

Participant: Rahul Kapoor; U. of Pennsylvania;

Participant: Riitta Katila; Stanford U.;

Participant: Mitrabaran Sarkar; Temple U.; <p>The BPS D

3

Find names of people participating in the session

```
import re  
m = re.findall('Word1.*$', _page, re.MULTILINE)
```

any character after

until the end of line

from

search all lines

See if you can find the names of participants. Experiment with different words.

How do we instruct Python to search for multiple words? Using google find a way to search for multiple words within a line of text.

```
m = re.findall('(Word1.*$|Word2.*$)', _page, re.MULTILINE)
```

Let's check what we have?

```
type(m)  
len(m)  
m[0]  
m[1]
```

That's why **'dynamic'** is a good thing.

3

Find names of people participating in the session

Here is how we search for some pattern in a string.

```
mytext = re.search('pattern', sometext).group(1)
```

search phrase

where

gives the first instance of a match

We can use this to extract only the names. Try to do it by yourself.

Answer: we will look for text between two strings. It should look something like that:

expression between the two strings

```
mytext = re.search('%s(.*)%s' % ('string1', 'string2'),  
mytext).group(1)
```

Let's now loop over all elements of the list `m` to create a new list `p` containing only names

```
p = [] # we create an empty list first  
for z in range(len(m)):  
    mytext = re.search('%s(.*)%s' % ('<strong>', '</strong>'),  
                        m[z]).group(1)  
    p.append(mytext)
```

4

Record the names in a list

Let's see what we remember from the Codecademy

Let's create an empty list and add our new list to it

```
LoL = [] # empty list of lists  
LoL.append(my_list)
```

We are building a list of lists that looks something like that:

A, B, C, D

E, F, G

H

I, B, J, K, L, M, N

O, P, J

Plan for today – what have we done so far

1

Have Python visit AOM 2014 session website

2

Extract text from the website

3

Find names of people participating in the session

4

Record the names in a list

5

Iterate over all sessions and add the names to the list of lists

6

Save the list of lists as a CSV file

First we will try to record 10 sessions (from 100 to 110). Remember to indent your code
Check what other roles are present in the AOM program and update your code.

Add Chair, Organizer, Discussant, Participant, Facilitator, Presenter, and Speaker to your list of keywords

```
for x in range(100, 109):  
    # do something with x  
    # print the names
```

Now let's iterate over more sessions. Let's try sessions 100 to 150.
Something went wrong with our code

Use the console to investigate the error and try again.

```
if len(m) > 0:
```

usually good, but in this case session 122 causes error

```
if "<strong>" in m[y]:
```

much better in our case

First we need to pick a location to store our output file

Let's be ambitious and create a code that will do that on any machine

```
from os.path import expanduser
from os import makedirs
from os.path import exists

_path = expanduser('~/' + Folder_name + '/')
if not exists(_path): # only if the folder doesn't exist yet
    makedirs(_path)
```

Time to record our file

```
import csv
filename = _path + 'output_day1.csv'
with open(filename, "wb") as csvfile:
    file_out = csv.writer(csvfile)
    file_out.writerow(your_list)
```

Let's check if the folder is indeed there.

Hurray – it is there. Well done!

1. You may want to try to also record the affiliation of each person in the format: FirstName LastName, SchoolName. The code would be very similar. You only need to find a way to extract the affiliation. Hint – look for the characters surrounding the name of the school, just like you did with to find the name.
2. Run the code for sessions 100 – 2,600. It will take around one to two hours, depending on the speed of Internet connection and AOM server load.
3. You may also inspect the code for any remaining problems and try to find a way to change the code to eliminate any errors.

Thank you

Maciej WORKIEWICZ

m.workiewicz@gmail.com | www.maciejworkiewicz.com

<https://github.com/Mac13kW>