

Data Mining

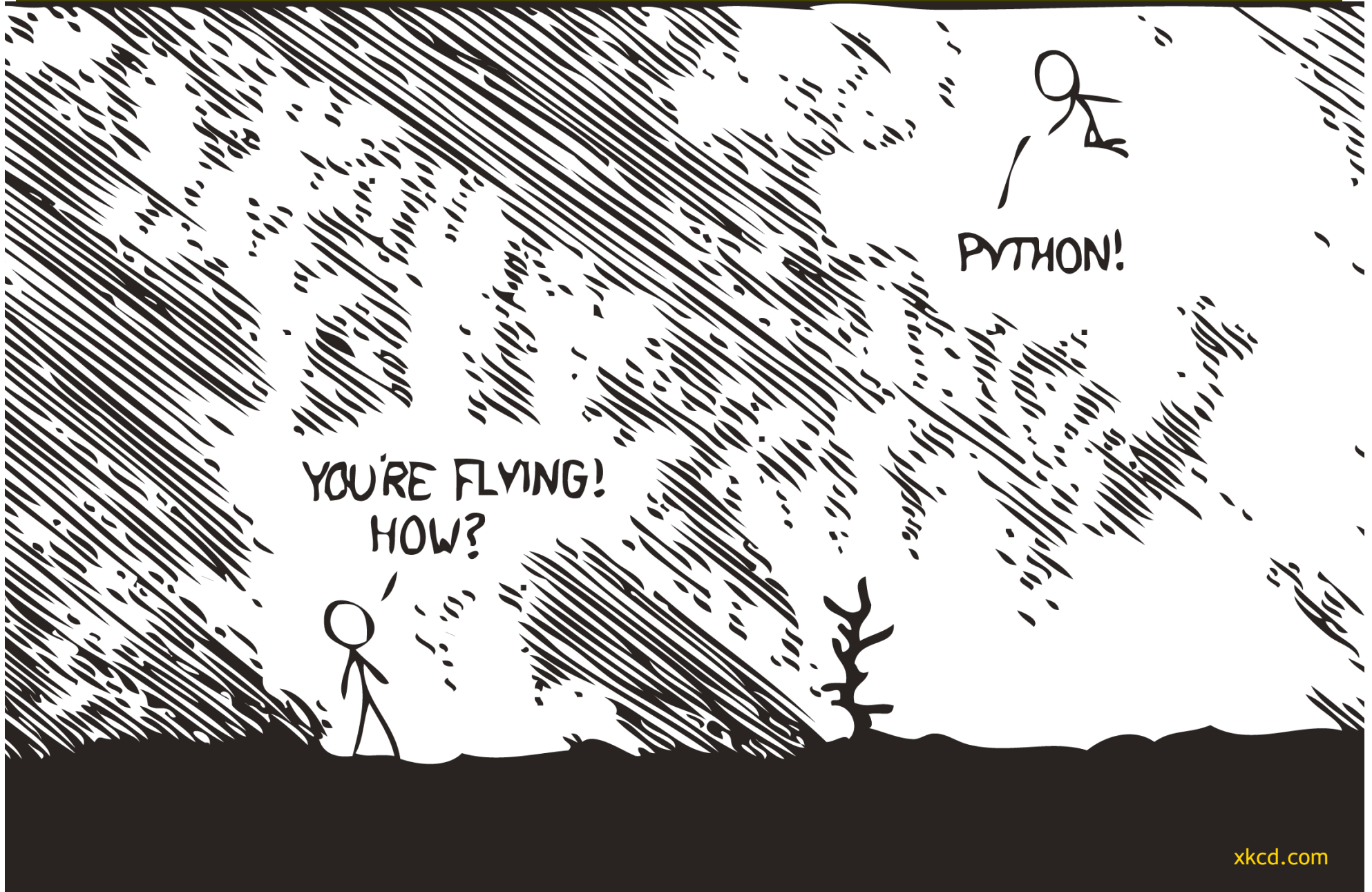
Samir Paul
spaul@fas

- > Internet == information
- > Let's extract it from the tubez and put it in a form we can use

- > Collecting stats, prices, email addresses, names, etc.
- > Might need data in a certain format
- > **Copy/Paste by hand is for suckers**

...but how will we do it?

How?



- > Python makes many things easier
- > Three steps:

1
In Python script, use [Beautiful Soup](#) to parse a website's HTML and store result. We'll call it "soup."

2
Use findAll function on the "soup" to get a list of all appearances of the tag you're looking for.

3
Use regular expressions to sort through each item in list (as a string). Do what you want.

- > Setting up Python
 - > Windows: python.org
 - > Mac/Linux: type “python” at command line
 - > Everyone: type “python” at NICE prompt
- > Interpreted language (can also be compiled)
- > No explicit types

- > Whitespace matters
- > No {}'s, just :'s
- > For loops are a bit different
- > No types!
~~int x = 5~~ instead... **x = 5**
- > ; at end of line is optional

```
def fact(n):  
    if(n == 0):  
        return 1  
    else:  
        return n * fact(n-1)
```

Then, we use the function:

```
> fact(5)  
120
```

1

```
a = [] #makes an array
for i in range(10):
    a.append(i*2) #fills the array with even numbers
```

```
> range(5)
[0,1,2,3,4]
> a
[0,2,4,6,8,10,12,14,16,18]
```

Can also do:

```
for ele in a:
    print ele
```

1



- > pattern-matching tool.
- > Very powerful.
- > We'll only explore a small subset.
- > Learn about regexps:

[on Wikipedia](#)
[in Python](#)

- > A language that allows us to see if a string matches a pattern we feed
- > Some examples:
 - > `[abc]` == match a,b, or c
 - > `[a-z]` == match any lowercase letters a through z
 - > `.` == matches any character other than new line (`'\n'`)
 - > `a*` == 'a' can be matched 0+ times
- > Make sure you import `re/util` in your python module

Let's write some code!