

INTRO to DATA SCIENCE

LECTURE 2: ETL AND DATA STORAGE

I. INTRO TO DATABASES

II. RELATIONAL DATABASES

III. NOSQL DATABASES

II. INTRO TO DATABASES

What is ETL?

- **E**xtract data
- **T**ransform data
- **L**oad data

Databases are a **structured** data source optimized for efficient **retrieval and storage**

Databases are a **structured** data source optimized for efficient **retrieval and storage**

structured : we will have to define some pre-defined organization strategy

retrieval : the ability to read data out

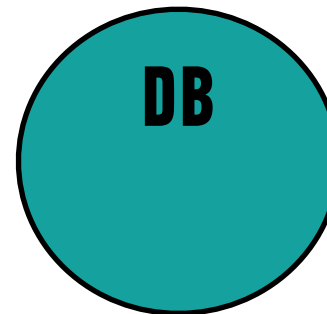
storage: the ability to write data and save it

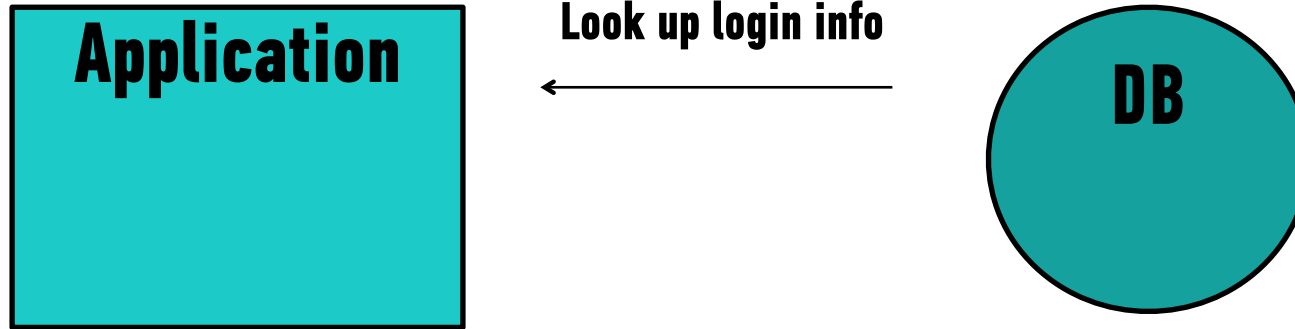
Databases are a **structured** data source optimized for efficient **retrieval and persistent storage**

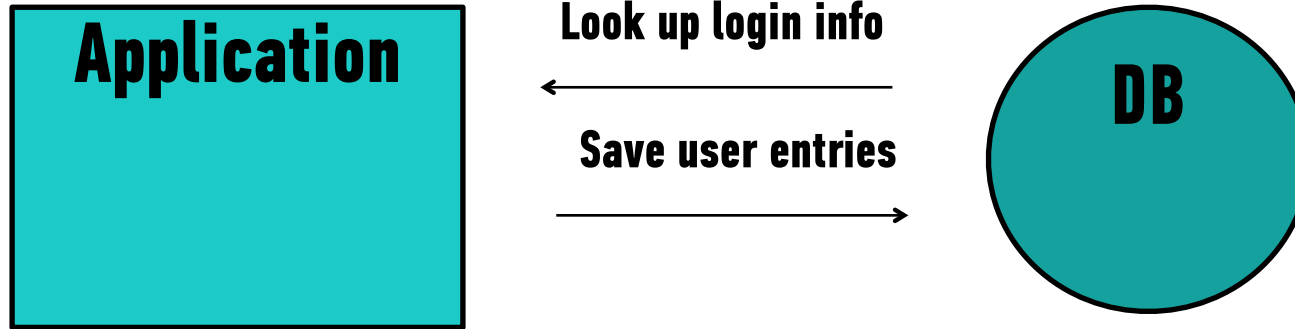
structured : we will have to define some pre-defined organization strategy

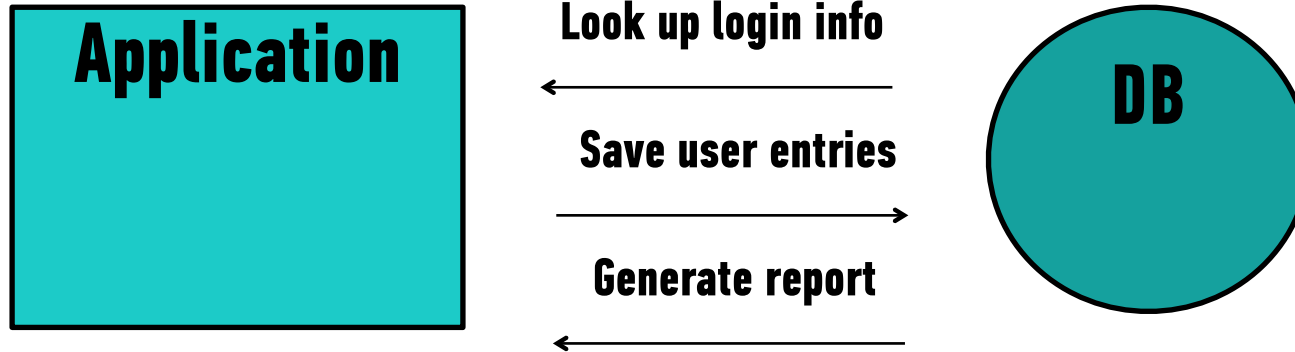
retrieval : the ability to read data our

storage: the ability to write data and save it







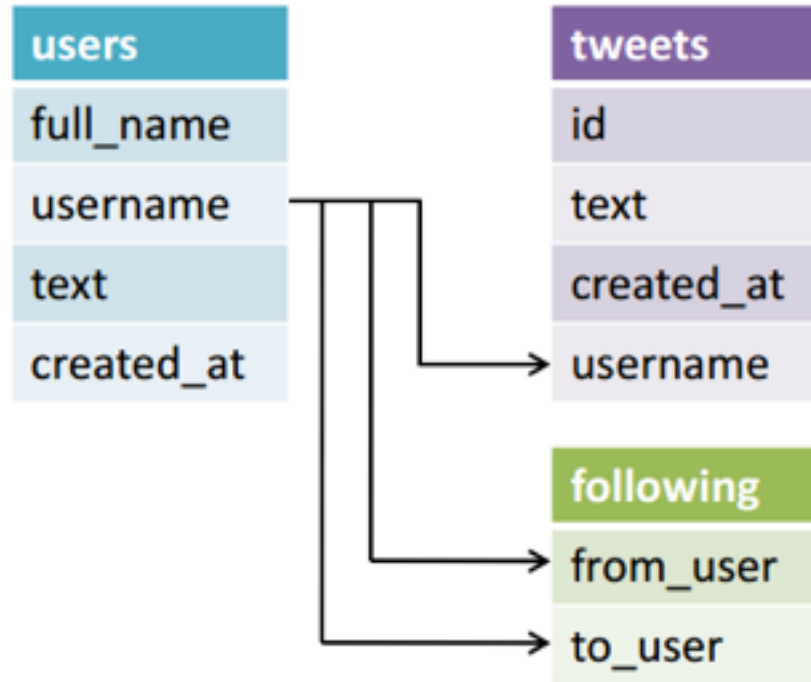


II. RELATIONAL DATABASES

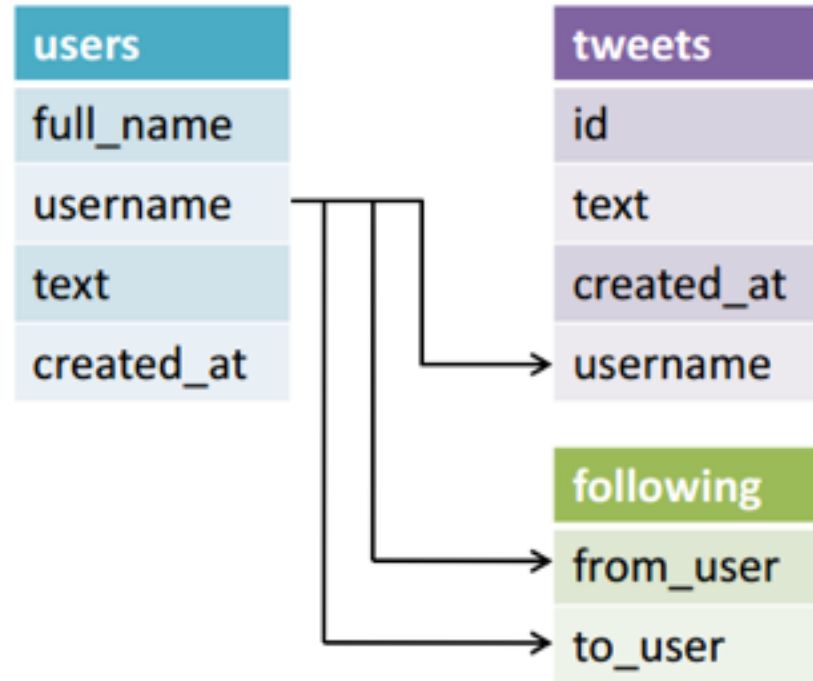
Relational database are traditionally organized in the following manner:

*A database has **tables** which represent individual entities or objects*

*Tables have a predefined **schema** – rules that tell it what columns exist and what they look like*

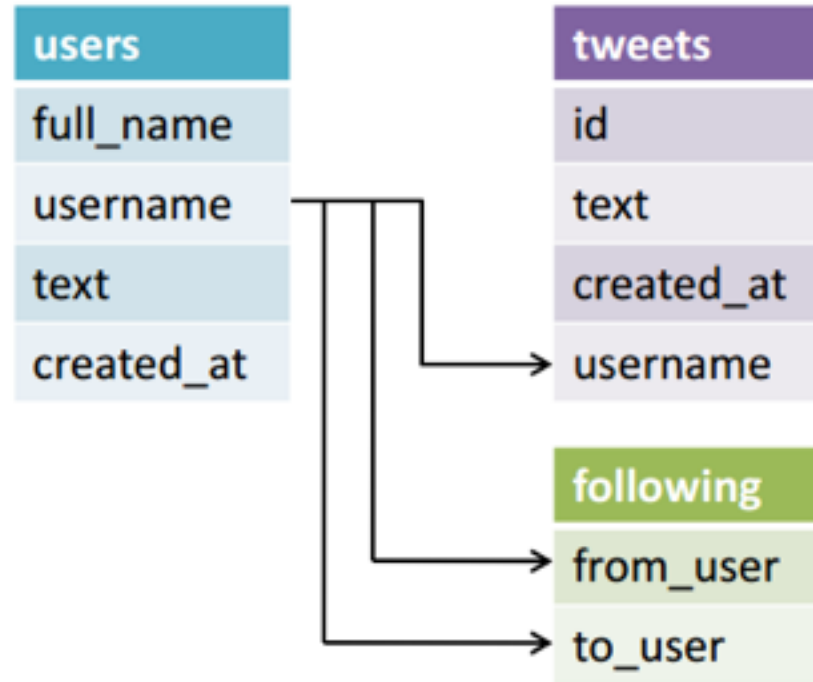


Each table should have a **primary key** column– a unique identifier for that row



*Each table should have a **primary key** column- a unique identifier for that row*

*Additionally each table can have a **foreign key** column- an id that links this to table to another*



We could have had a table structure as follow:
Why is this different?

tweets
id
text
created_at
username
full_name
username
text
created_at

We could have had a table structure as follow:

Why is this different?

We would repeat the user information on each row.

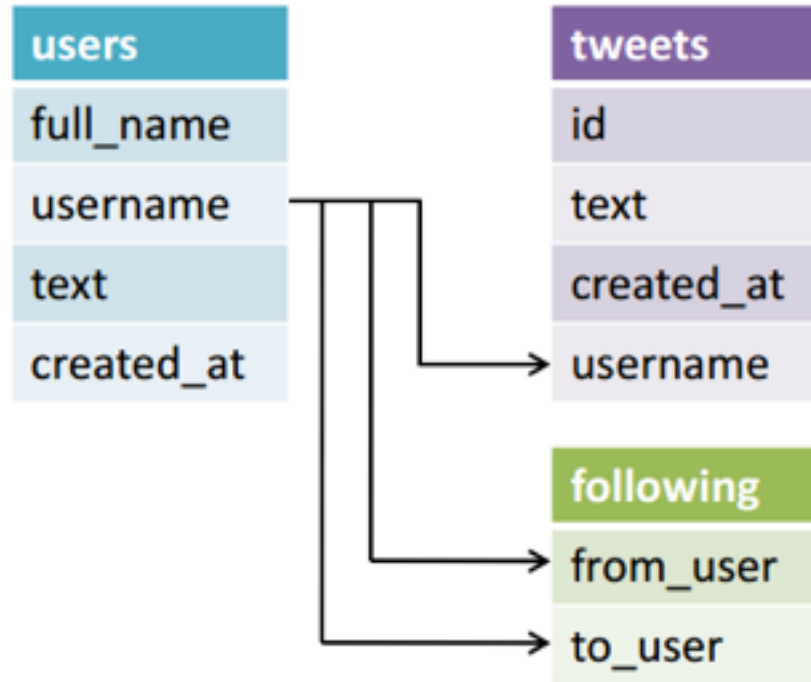
This is called
denormalization

tweets
id
text
created_at
username
full_name
username
text
created_at

Normalized Data: Many tables to reduce redundant or repeated data in a table

Denormalized Data:

Wide data, fields are often repeated but removes the need to join together multiple tables



tweets
id
text
created_at
username
full_name
username
text
created_at

Normalized Data: Many tables to reduce redundant or repeated data in a table

Denormalized Data:

Wide data, fields are often repeated but removes the need to join together multiple tables

Trade off of speed vs. storage

Q: How do we commonly evaluate databases?

read-speed vs. write speed

Q: How do we commonly evaluate databases?

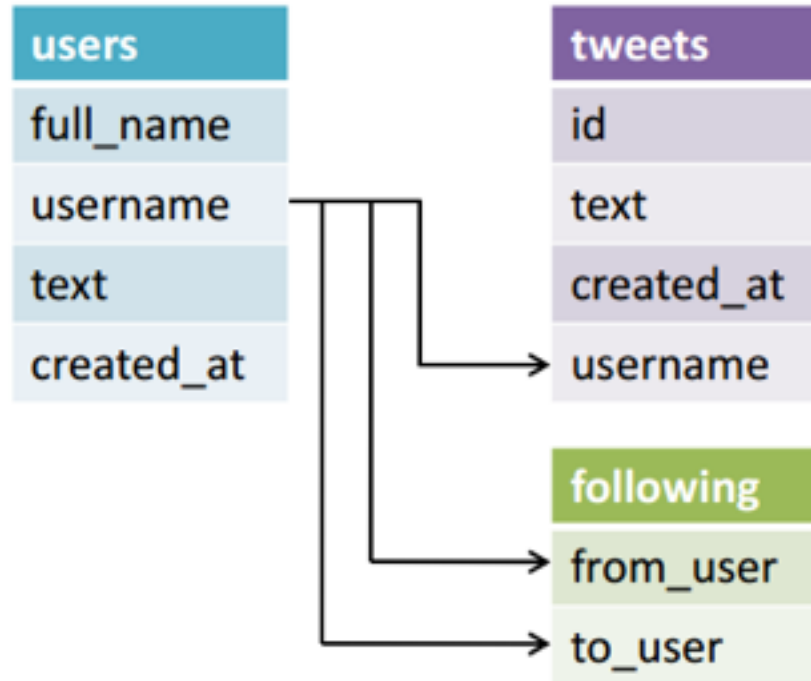
read-speed vs. write speed

space considerations

(...and many other criteria)

Q: Why are normalized tables (possibly) slower to read?

Q: Why are normalized tables (possibly) slower to read?



Q: Why are normalized tables (possibly) slower to read?

A: We'll have to get data from multiple tables to answer some questions.

Q: Why are denormalized tables (possibly) slower to write?

Q: Why are denormalized tables (possibly) slower to write?

tweets
id
text
created_at
username
full_name
username
text
created_at

Q: Why are denormalized tables (possibly) slower to write?

A: We'll have to write more information on each write

SQL is a query language to load, retrieve and update data in relational databases

SELECT: Allows you to **retrieve** information from a table

Syntax:

SELECT col 1, col2 FROM table WHERE <some condition>

Example:

***SELECT poll_title, poll_date FROM polls WHERE romney_pct >
obama_pct***

GROUP BY: Allows you to ***aggregate*** information from a table

Syntax:

SELECT col1, AVG(col2) FROM table GROUP BY col1

Example:

***SELECT poll_date, AVG(obama_pct) FROM polls GROUP BY
poll_date***

GROUP BY: Allows you to ***aggregate*** information from a table

Syntax:

SELECT col1, AVG(col2) FROM table GROUP BY col1

Example:

***SELECT poll_date, AVG(obama_pct) FROM polls GROUP BY
poll_date***

GROUP BY: Allows you to ***aggregate*** information from a table

Syntax:

SELECT col1, AVG(col2) FROM table GROUP BY col1

There are usually a few common built-in operations:

SUM, AVG, MIN, MAX, COUNT

JOIN: Allows you to **combine** multiple tables

Syntax:

SELECT table 1.col 1, table 1.col 2, table 2.col 2

FROM table 1 JOIN table 2 ON table 1.col 1 = table 2.col 2

JOIN: Allows you to **combine** multiple tables

Syntax:

SELECT table 1.col 1, table 1.col 2, table 2.col 2
FROM (JOIN table 1, table 2 ON table 1.col 1 = table 2.col 2)

INSERT: Allows you to ***add*** data to tables

Syntax and Example:

***INSERT INTO <table> (col1, col2)
VALUES(...)***

***INSERT INTO classroom (first_name, last_name)
VALUES('John', 'Doe');***

INTRO TO DATA SCIENCE

I. INTRO TO PYTHON

Q: What is Python?

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*

high-level: *interpreted (not compiled)*

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

open source: *free! (both binaries and source files)*

high-level: *interpreted (not compiled)*

dynamic: *things that would typically happen at compile time happen at runtime instead (eg, dynamic typing)*

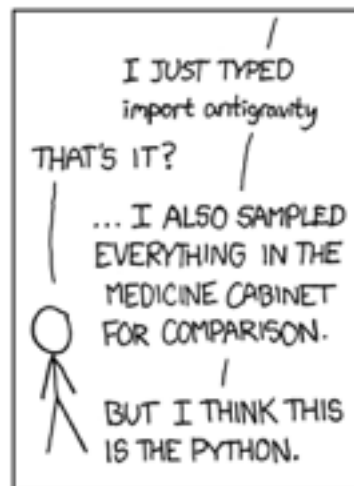
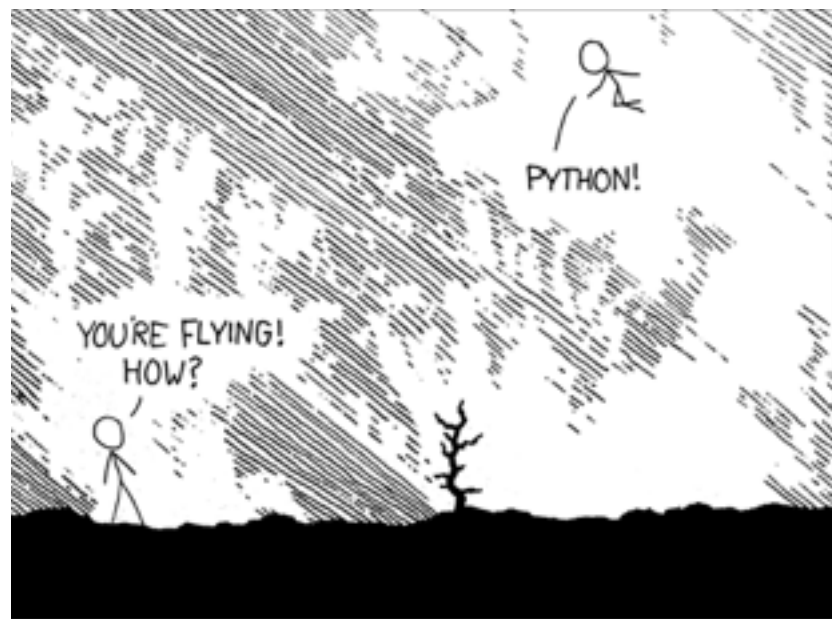
- Created by Guido van Rossum in 1991
- Benevolent Dictator for Life



- Created by Guido van Rossum in 1991
- Benevolent Dictator for Life
- Currently on version 3 ...
 - but most still use 2.7+



WHY PYTHON?



WHAT ARE THE ADVANTAGES TO PYTHON?

- Batteries Included: Large collection of built in libraries
- Multi-paradigm: many different programming methodologies apply
- Simple and clean syntax

WHAT ARE THE ADVANTAGES TO PYTHON?

- Batteries Included: Large collection of built in libraries
- Multi-paradigm: many different programming methodologies apply
- Simple and clean syntax - but we have to pay attention to whitespace

WHAT ARE THE ADVANTAGES TO PYTHON?

- Lots of tools built-in to the standard library
- Easy to install new package: pip, easy_install
 - Try
 - > pip install oauth
 - > pip install django

WHAT ARE THE ADVANTAGES TO PYTHON?

- Scripting language
- Functional programming
- Object oriented programming

WHAT ARE THE ADVANTAGES TO PYTHON?

- Java
 - `public static void main(String [] args)`
 - `{`
 - `System.out.println("Hello world");`
 - `}`

WHAT ARE THE ADVANTAGES TO PYTHON?

- Python:
 - print “Hello World”

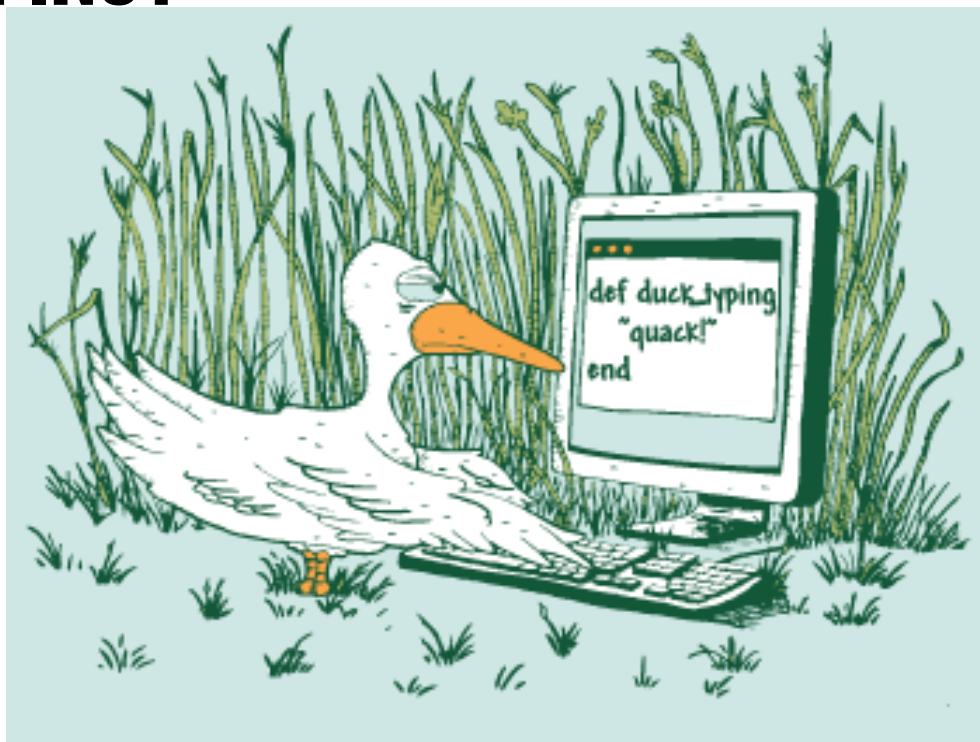
WHAT SETS PYTHON APART?

- Type system:
 - Dynamic typing!

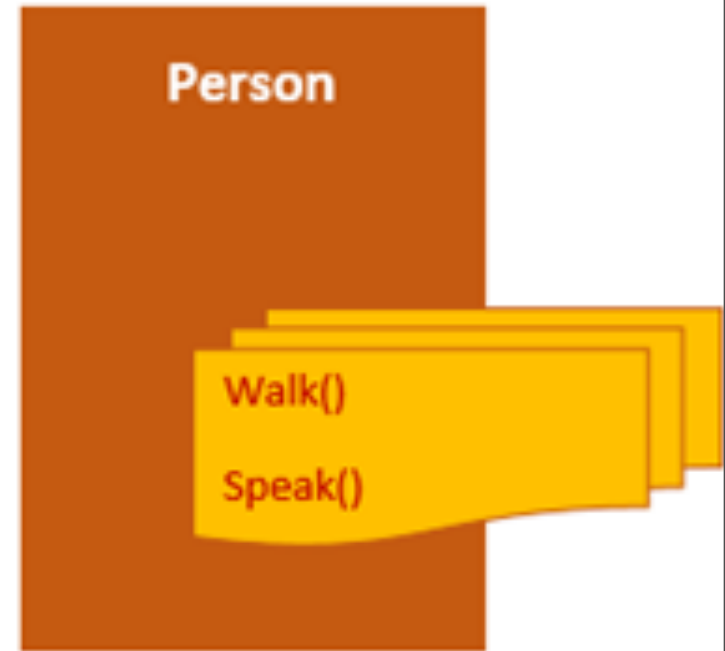
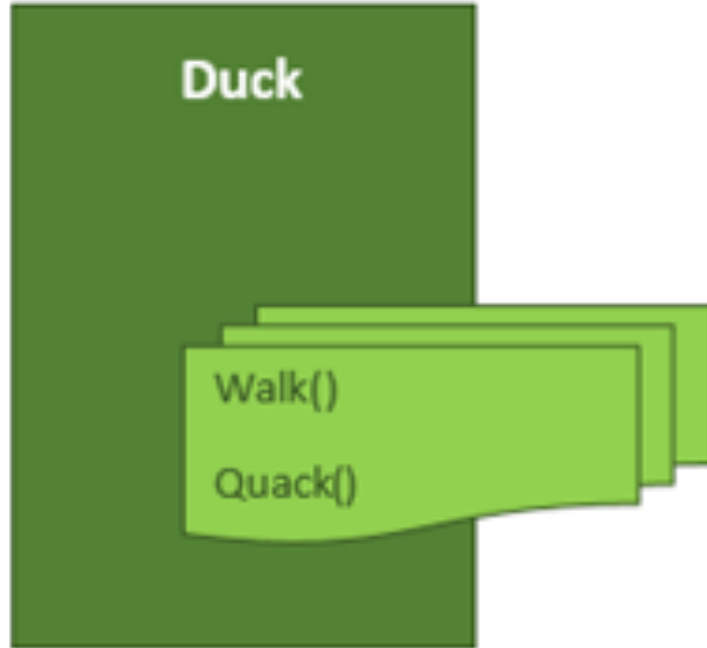
WHAT IS TYPING?

- Need to tell the program WHAT something is:
 - C, Java: `double pi = 3.14...`
- Can lead to hard to read to code
- But also means safer code

WHAT IS TYPING?



WHAT IS TYPING?



WHAT SETS PYTHON APART?

- Type system:
 - Dynamic typing!

```
>>> x = 1
>>> x
1
>>> x = 'horseshoe'
>>> x
'horseshoe'
>>> _
```

WHAT SETS PYTHON APART?

- Type system:
 - Dynamic typing!
- Interpreted language
 - No compilation

PYTHON SYNTAX

SETTING UP VARIABLES

- Python shell is just a complex calculator:
 - `10 * 15`
 - `x = 5`
 - `x #prints 5`
 - `x ^ 2 #prints 25`

*The most basic data structure is the **None** type. This is the equivalent of NULL in other languages.*

*There are four basic numeric types: **int**, **float**, **bool**, **complex**, **string***

```
>>> type(1)
<type 'int'>
>>> type(2.5)
<type 'float'>
>>> type(True)
<type 'bool'>
>>> type(2+3j)
<type 'complex'>
```

DATA TYPES

- Lists:

- `l = [1, 2, 3]`

- `l = ['happy', 'sad', 'indifferent']`

- Dictionaries (Maps):

- Key-Value datastructure

- `d = { 'first_name' : 'Arun', 'last_name': 'Ahuja' }`

-

IF/ELSE STATEMENTS

- If/Else statements allow us to take different paths through depending on some condition:
- `x = 5`
- `if x > 4:`
 - `print "This number was less than 4"`

LOOPING

- Looping allows us to pass through some set of values and perform an operation on each
- `l = ["happy", "sad", "don't care"]`
- `for x in l:`
 - `print x`
 - `if x == 'happy':`
 - `print x*3`

FUNCTIONS

- Functions allow us to save some piece of functionality to reuse later
- `def func(x):`
 - `if x > 4:`
 - `print "This number is less than 4"`
 - `>> func(6)`
 -

*Our final example of a data type is the Python **file object**. This represents an open connection to a file (eg) on your laptop.*

```
>>> with open('output_file.txt', 'w') as f:  
...     f.write(my_output)
```

*These are particularly easy to use in Python, especially using the **with statement context manager**, which automatically closes the file handle when it goes out of scope.*

*Python allows you to define custom **functions** as you would expect:*

```
>>> def x_minus_3(x):  
...     return x - 3  
...  
>>> x_minus_3(12)  
9
```

*Functions can optionally return a value with a **return statement** (as this example does).*

*Functions can take a number of **arguments** as inputs, and these arguments can be specified in two ways:*

As positional arguments:

```
>>> def f(x, y):  
...     return x - y  
...  
>>> f(4,2)  
2  
>>> f(2,4)  
-2
```

*Functions can take a number of **arguments** as inputs, and these arguments can be specified in two ways:*

*Or as **keyword arguments**:*

```
>>> def g(arg1=x, arg2=y):  
...     return arg1 / float(arg2)  
...  
>>> g(arg1=10, arg2=5)  
2.0  
>>> g(arg2=100, arg1=10)  
0.1
```

*Python supports **classes** with member attributes and functions:*

```
>>> class Circle():
...     def __init__(self, r=1):
...         self.radius = r
...     def area(self):
...         return 3.14 * self.radius * self.radius
...
>>> c = Circle(4)
>>> c.radius
4
>>> c.area
<bound method Circle.area of <__main__.Circle instance at 0x1060778c0>>
>>> c.area()
50.24
>>> 3.14 * 4 * 4
50.24
```