# Lecture 1:

Setup

# Lecture 2:

<https://www.tutorialspoint.com/python/python_basic_operators.htm>

# Programming Basics

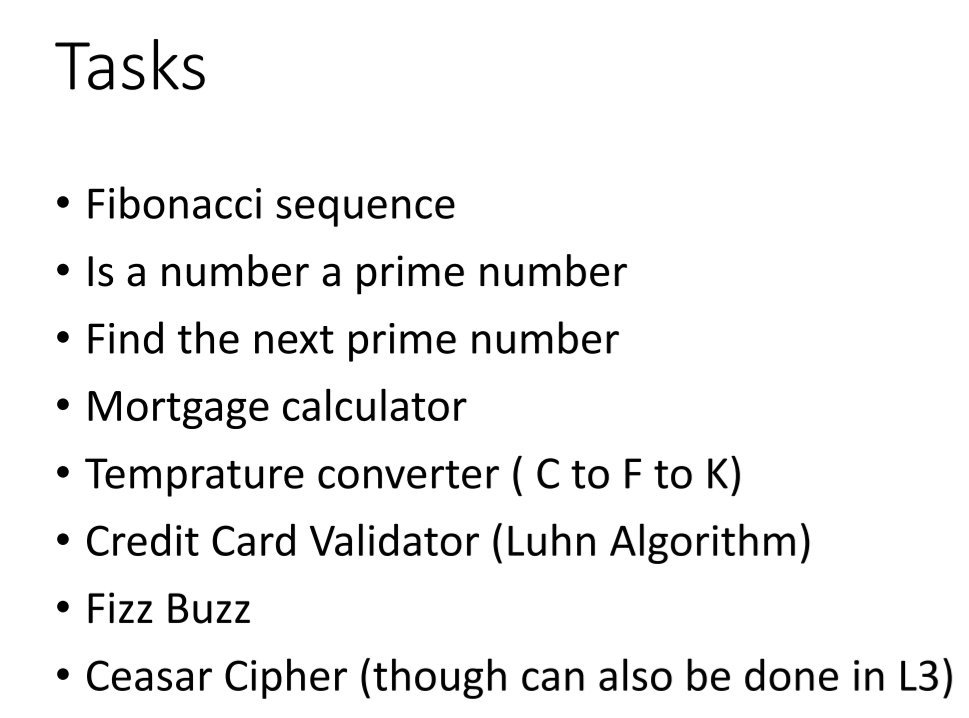
& Binary Operator

// result is the next smaller whole number that is divideable:

-11.0//3 = -4.0 # takes -12 as next smaller number



Tasks 2:

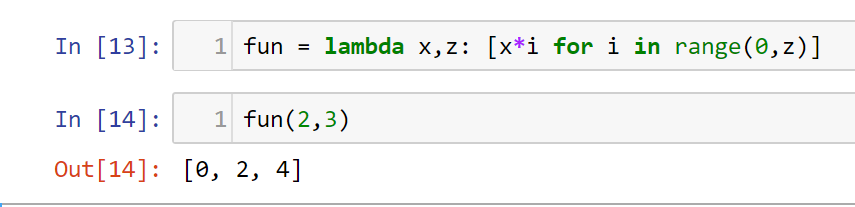


# Lecture 3:

Functions

Lambda Functions

Function = lambda input: operation



# Lecture 4:

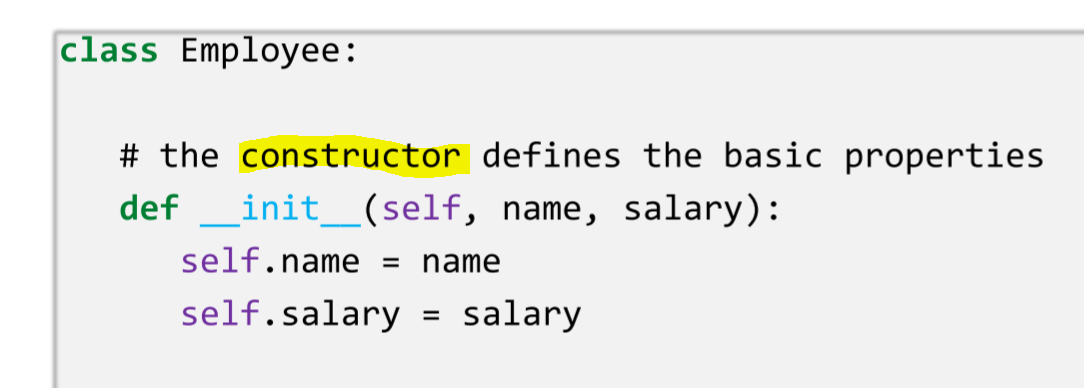
OOP

Class file names should be all lowercase.py while classes should be capital

# Lecture 5:

## OOP

## Constructor and Methods



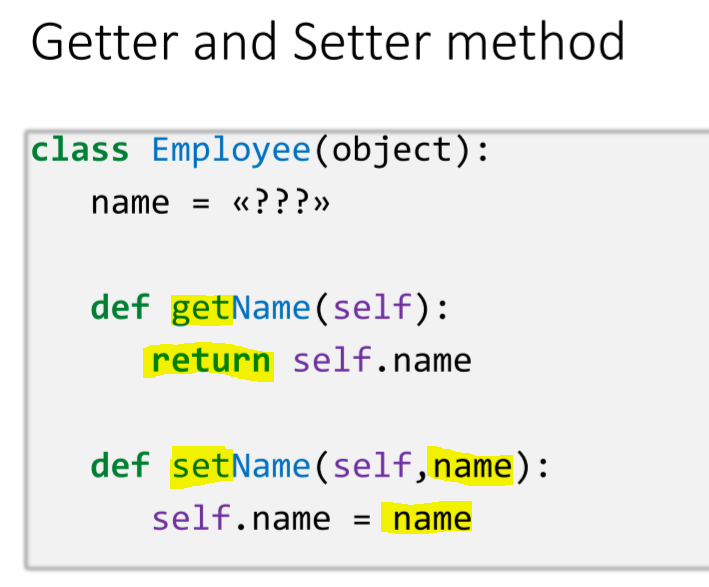
## Instantiate/Initialize and Object

Effects = Estim\_Effects(df, y, compare, x1, x2)

Call Method:

Effects.theta\_hat()

## Getter and Setter



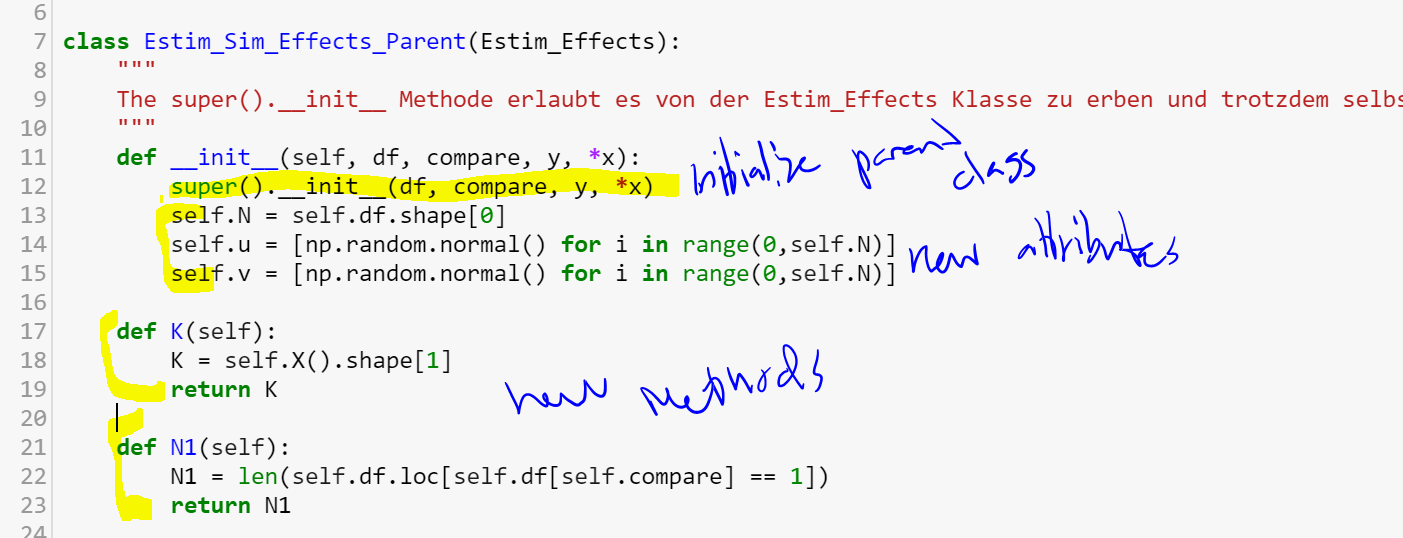
## Importing a class

from estimeffects import Estim\_Effects

## Inheritance

\_\_init\_\_(self, y, compare, x):

Super().\_\_init\_\_(y,compare,\*x)



# Exercises 5:

Library w/ code make it run

Text based Zork game (action adventure)

Airline challenge. I.e. build an airline operation in OOP

Try to model international trade.

# Lecture 6:

APIs

Def

Application programming interfaces are access points of for developers to tap into systems and make two software programs communicate with one another.

An API facilitates the connection process of two programs by returning a certain output when a request with certain parameters is made

Through an API you allow someone else to access your code by giving her certain methods and properties she can use.

The end user is not in contact with the API. The developer can connect two softwares, so that they communicate through an API.

An API can tap into a database and return outputs from this database.

Call:

Ask for information which is received through an http request.

Rest:

Representational State Transerfer

graph.facebook.com/youtube = [www.facebook.com/youtube](http://www.facebook.com/youtube)

# Exercises 6:

http connectionless protocol connection closes

# Lecture 7:

Def of a DB:

* Data structure which stores organized information
* Organized:
  + Ease of acess, management, updating
  + Computer can easily find the Information

Data: distinct pieces of information, organized in a specific way for use of analysis.

Overview:

Flat File Db

Relational:

* R-DB-MS:
  + Made up of tables that fit into a predefined cathegory
  + Entries (Fields) save das records/tuples
* 1 Primary Key per table
* Often linked to a foreign key. This key only exists once in the database

NoSQL:

* Key-Value
* Column Oriented
* Document Oriented
* Graph DB

Keys:

Foreign keys:

* key used to link two tables together.
* Foreign key of one table refers to the primary key of another table.

Primary Keys:

* Unique identifier of a record within this table
* Cannot contain Null values
* A table can have only one primary key. This one can have one or multiple fields.
  + E.g. iid, or, lastname AND Firstname

SQL:

Structures queary language

For relational db management.

Data Definition Language (DDL):

* Used to define the db structure or schema
  + CREATE, ALTER (changes struct of db), DROP, TRUNCATE (remove all records from a table)

Data Manipulation Language:

* Used for managing data within schemas:

# Lecture 8:

Normalization:

* Process of reorganizing data in a db so that it meets two basic requirements.
  + No redundancy (no records exists twice; every information is stored in only one place.
  + Data integrity: data dependencies are logical = related data is stored together.
  + == > We don’t store course information with each course of each student.
    - bc we would store information multiple times and relate data that does not need to be related.

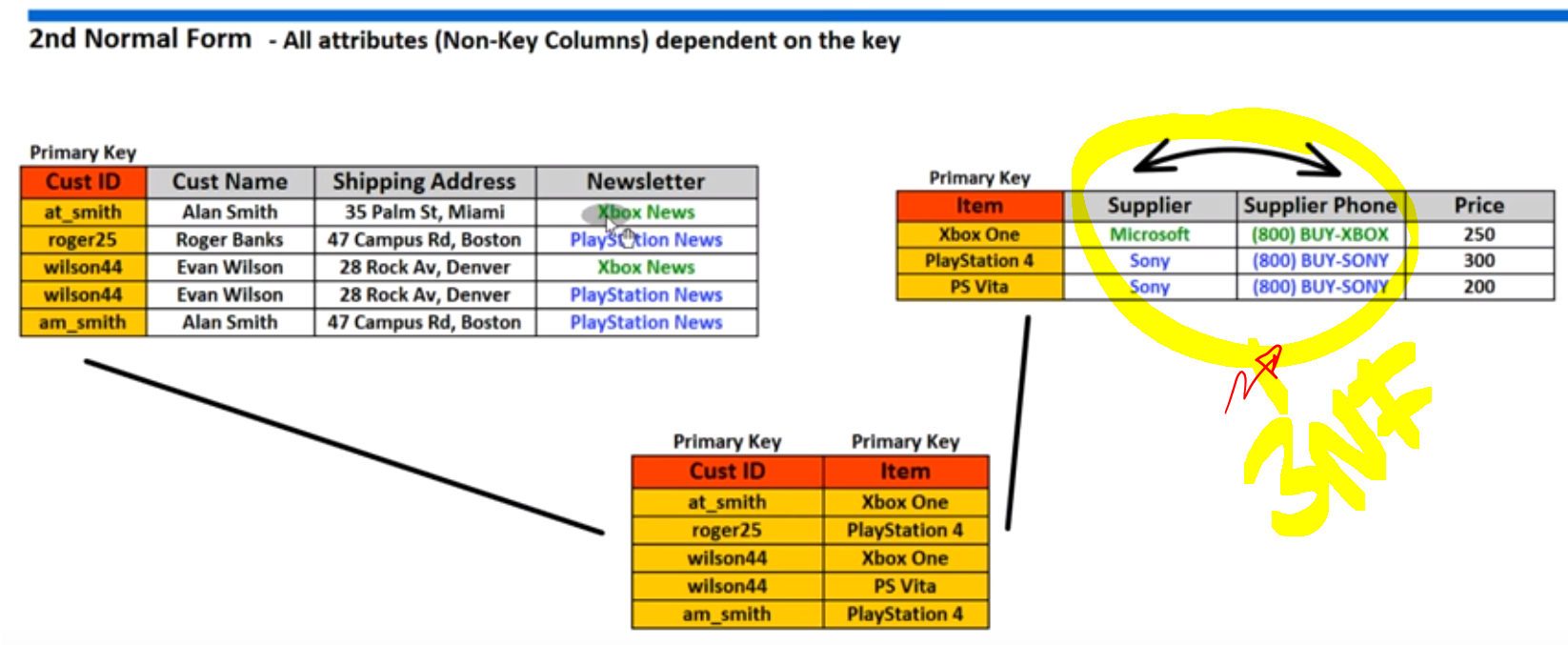
3 types of normalization:

1. First normal form (1NF)

* No redundancies in individual tables (no duplicates in individual tables)
* Single value in each cell
* Entries within one filed are the same type
* Unique Identifier:
  + Each row has a unique id
  + Columns are grouped together with a primary key e.g. [name, first name, birthday] !Not allowed for 3NF Third Normal Form!

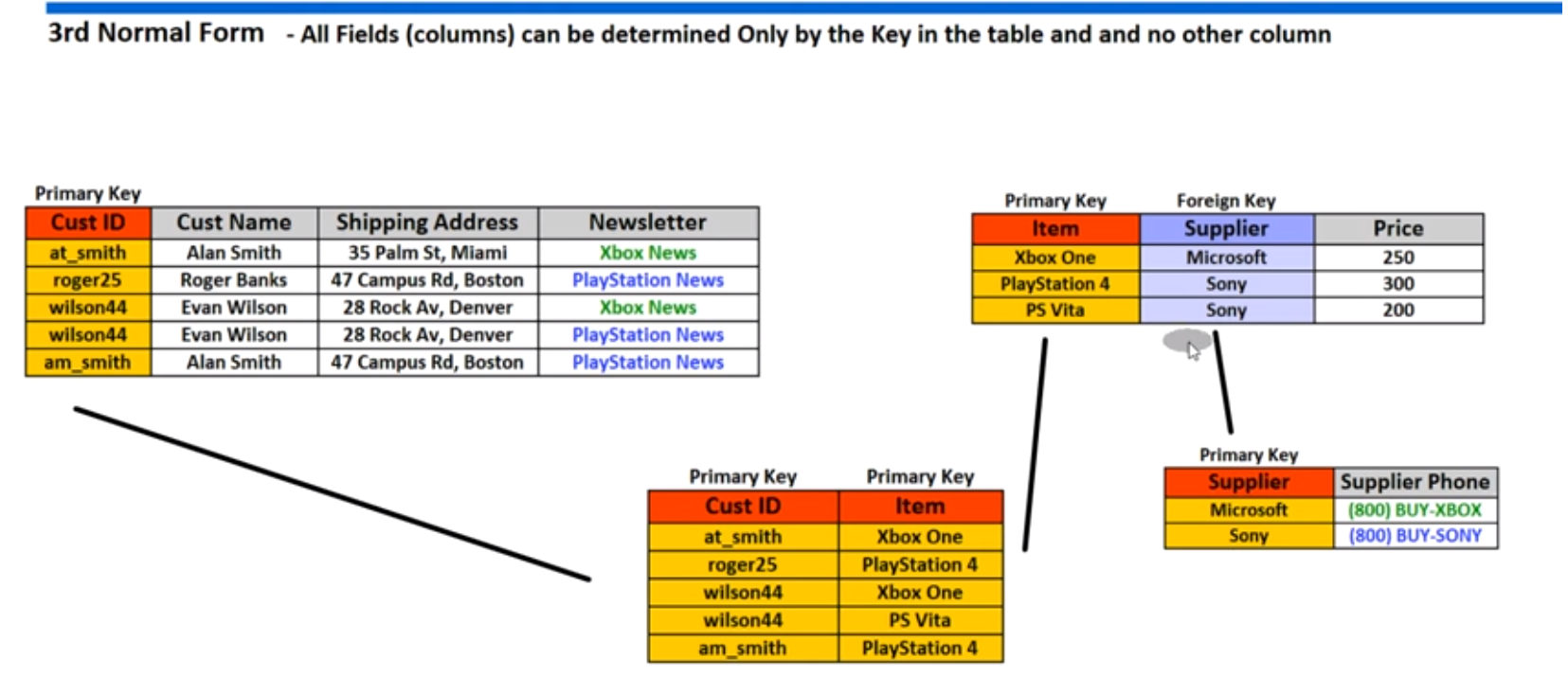
1. Second Normal From (2NF)

* 1NF
* All attributes dependent on 1 Key



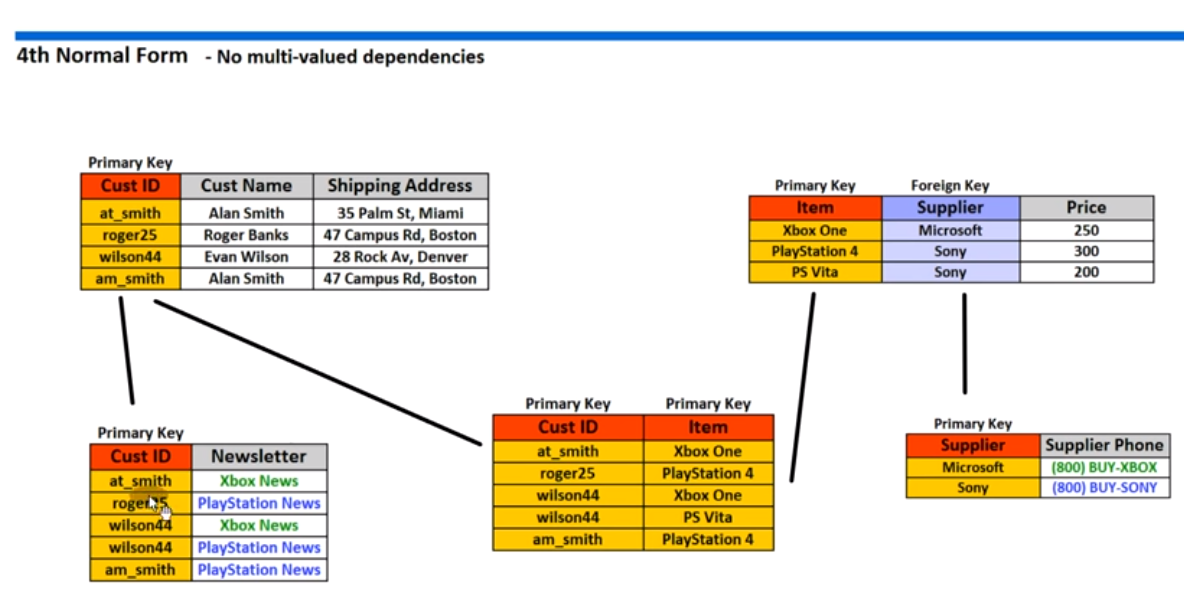
1. Third normal form (3NF)

* 2NF
* All fields can be determined only by the key in the table and no other columns.
* “No multivalued dependencies”



This allows me to change the phone number of the supplier in one place only.

**4NF according to video** (<https://www.youtube.com/watch?v=UrYLYV7WSHM> ):



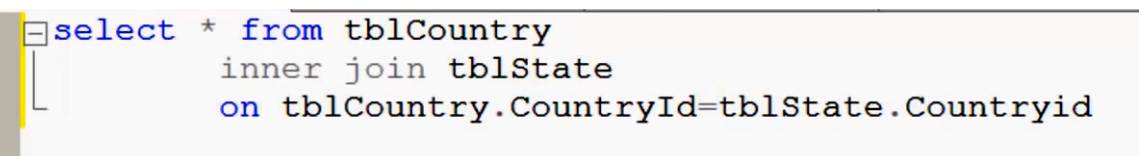
The same is true for the address of the customer. I can change it in one place only.

# DML (Data Manipulation Language)

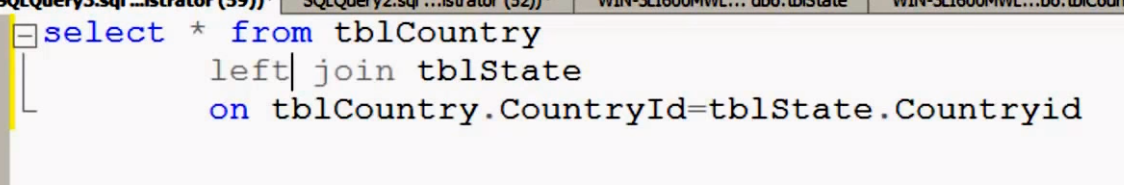
* + SELECT
    - SELECT DISTINCT last\_name, first\_name from actors; # delete duplicates
    - SELECT FROM actors occupation AS “Part-Time Actor” WHERE film\_count < 5;
    - SELECT FROM actors WHERE film\_count > 5 ORDER BY film\_count DESC
  + INSERT
    - INSERT INTO actors(id,last\_name,first\_name) VALUES (3, “Keira”, “Knightley”)
  + UPDATE
    - UPDATE actors set film\_count = film\_count \* 3 where id = 3;
  + DELETE
    - DELETE FROM actors WHERE last\_name = “Walker”;

JOINS

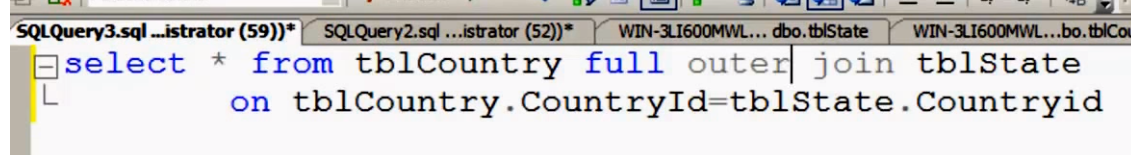
Inner:

* Selects records from the two tables which have matching values. These are joined together.
* Only the values which exist in both tables. E.g. only country ids which exist in both tables.
* 

Left/right outer:

* All data from left/right table but only the matching data from the other table
* 

Full outer:



* All values from both tables

Cross join:

* Every row of table 1 is joined with every row of table two.
* Customer 1 & 2 joined with Product 1-3

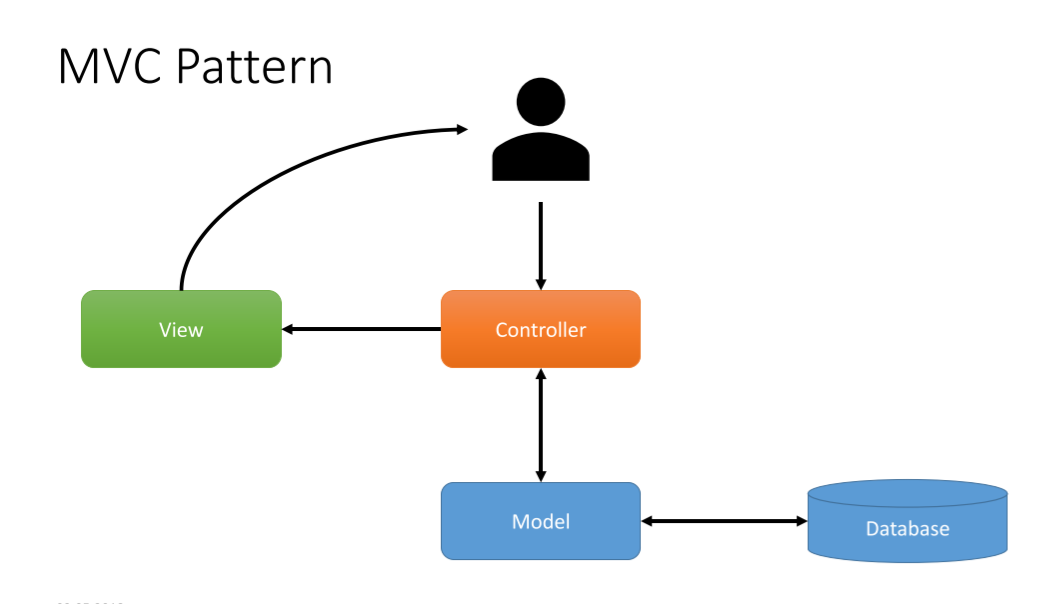
|  |  |
| --- | --- |
| * Customer 1 | * Product 1 |
| * Customer 1 | * Product 2 |
| * Customer 1 | * Product 3 |
| * Customer 2 | * Product 1 |
| * Customer 2 | * Product 2 |
| * Customer 2 | * Product 3 |

* Select \* From table1 cross join table 2;

# Lecture 9:

MVC – Model View Controller:

* + Software Architectural Design Pattern
  + One of the most frequently used patterns
  + Seperates appliaction functionality
  + Promotes organized and structured programming



Interactions with the database (Insert, Select, Update, Delete)

Processing data from or to the database

Communicates (only) with the Controller

Middle-Man (brain)

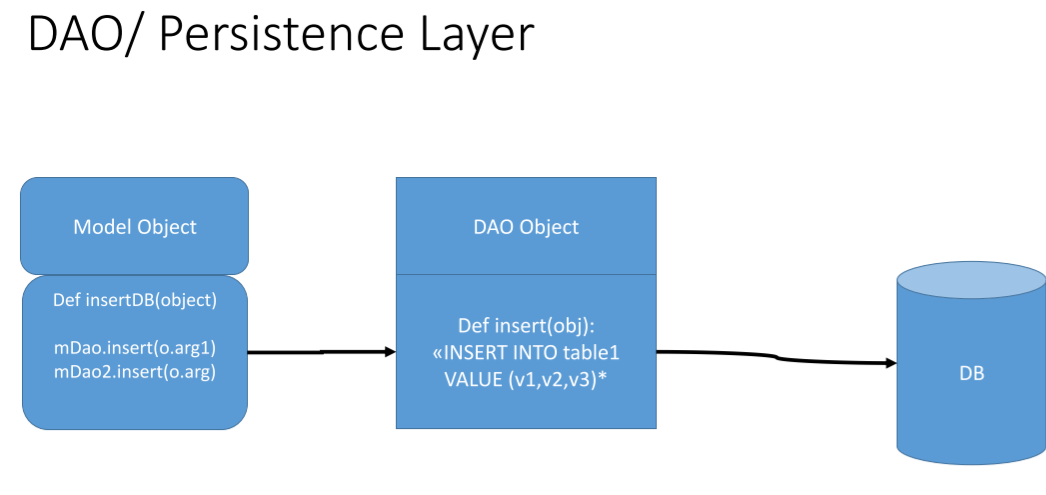
Receives input from the user

UI

Communicates only with controller

Between Model and Database:

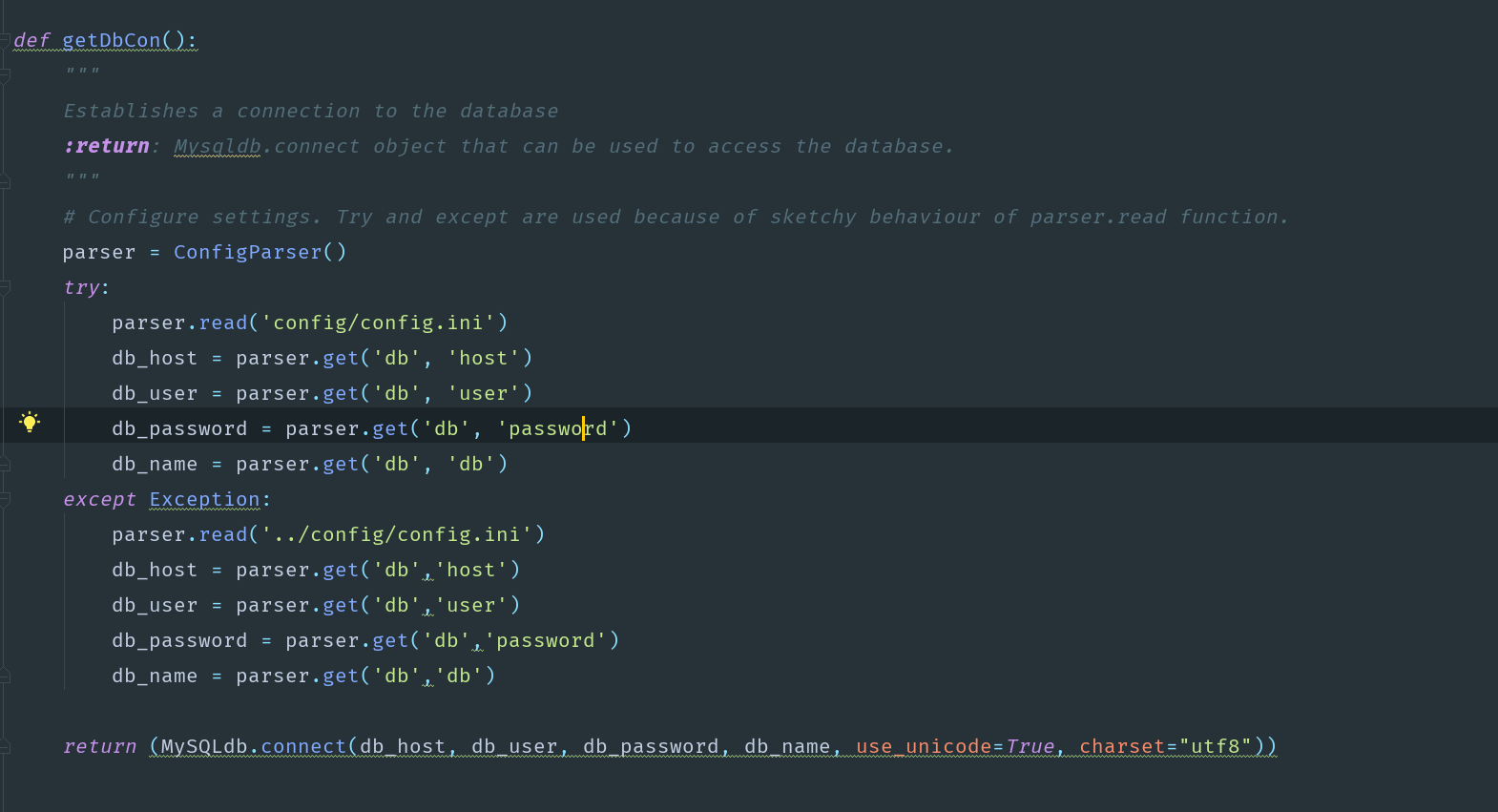
DAO – Data Access Object



Implements commands into the database.

If you change the database, you won’t have to rewrite the whole model but only the DAO.

Connection to the database



Could add:

charset = ‘utf8mb4’

cursorclass = pymsql.cursors.Dictcursor (for pymysql)

Commit:

Confirms the executed statement to the db.

No *best* time for committing. Best, when all data was executed and is now available.

# Databases nice to know:

## Forward Engineering:

Taking model and letting it live on the server.

## Reverse Engineer:

Taking db from server and allowing editing or saving it locally.

Create a model of a n existing database by reverse engineering it.

## Relationships:

<https://code.tutsplus.com/articles/sql-for-beginners-part-3-database-relationships--net-8561>

Customer ------ Address

**One to One | 1:1**

Each Customer only has one address. Each customer and each address only exist once. This mean different family members have to be saved with different addresses.

**One to Many / One to many | 1:n**

One customer can have many addresses

One customer can place many orders

**Many to many | n:m**

One item can be in many orders and one order may contain many items.

**Rest API – Restful API (i.e. conforming with the rest constraints)**

We are talking about web APIs

* Client (give me this data)
* Waiter (API, programmed to return a specific data)
* Kitchen (Server)

Representational State Transfer.

Treat objects on server side as resources e.g. blogpost in database.

Create with post request

Delete with delete request

Rest operates just using HTTP and json (can be used by any language bc they can all work with RESTful interfaces)

**Requests:**

GET: retrieve data from a specific resource

POST: submit data to be processed to a specific resource

PUT: update a specified resource

DELETE:

HEAD: same as get but does not return a body

OPTIONS: see supported methods of a server

PATCH: Update partial resources

## Authentication:

## To Do:

Study API calling

3 DB Tasks from Lecture 7

Lecture 8:

Tasks • Database Exercises 1 and 2 on Dropbox (1.16 and from ~2.10 the exercises get more difficult => you may need to google the GROUP BY command and also what queries within queris are)

• For those who want to learn more i can recommend the following: • https://www.w3resource.com/sql-exercises/sql-retrieve-from-table.php • https://www.w3resource.com/sql-exercises/sql-aggregate-functions.php • https://www.w3resource.com/sql-exercises/sql-exercises-quering-onmultiple-table.php

• <https://www.w3resource.com/sql-exercises/sql-joins-exercises.php>

## Topics:

Circular dependence

Injection testing

Questions:

What if I want to call a class in a method of itself

Estim\_Effects.std\_theta\_hat\_boot()

Naming of class files. Why lowercase? Cannot differentiate form functions if I do this.