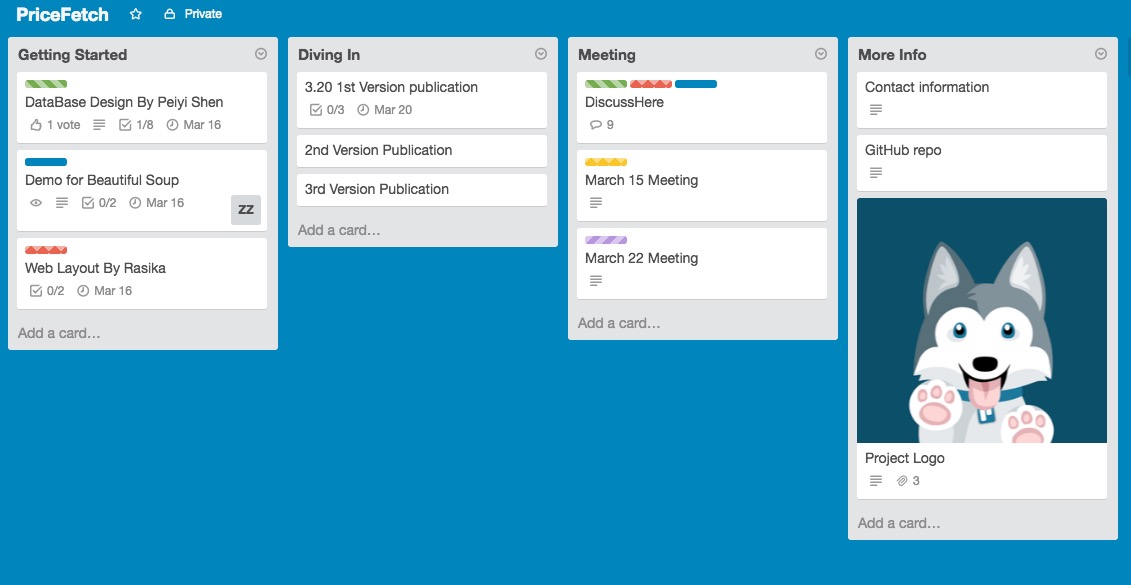
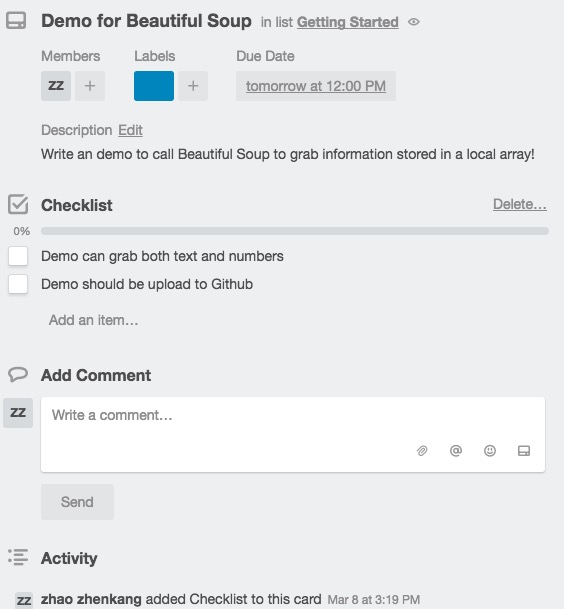


- GROUP 6

**TASK PLAN AND TRACKING:**

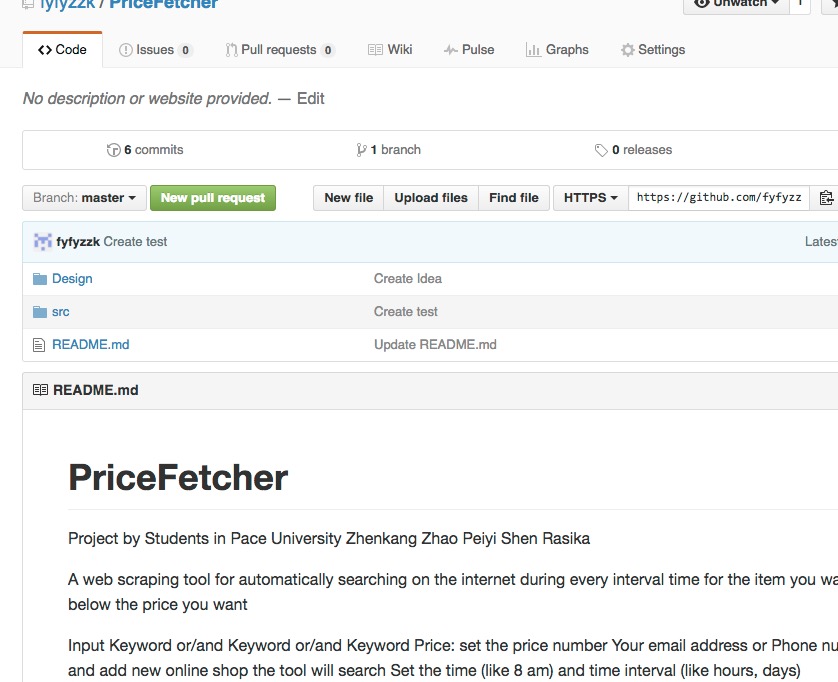
We choose trello.com





Source Control

<https://github.com/fyfyzzk/PriceFetcher>



Source Code

BeautifulSoup Demo for geting images

# -\*- coding: utf-8 -\*-  
  
#---------------------------------------   
#---------------------------------------  
  
import urllib.request  
import bs4,os  
  
page\_sum = 1 #设置下载页数  
  
path = os.getcwd()  
path = os.path.join(path,'暴走GIF')  
if not os.path.exists(path):  
 os.mkdir(path) #创建文件夹  
  
url = "http://baozoumanhua.com/gif/month/page/" #url地址  
headers = { #伪装浏览器  
 'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)'  
 ' Chrome/32.0.1700.76 Safari/537.36'  
}  
  
for count in range(page\_sum):  
 req = urllib.request.Request(  
 url = url+str(count+1),  
 headers = headers  
 )  
 print(req.full\_url)  
 content = urllib.request.urlopen(req).read()  
  
 soup = bs4.BeautifulSoup(content) # BeautifulSoup  
 img\_content = soup.findAll('img',attrs={'style':'width:460px'})  
  
 url\_list = [img['src'] for img in img\_content] #列表推导 url  
 title\_list = [img['alt'] for img in img\_content] #图片名称  
  
 for i in range(url\_list.\_\_len\_\_()) :  
 imgurl = url\_list[i]  
 filename = path + os.sep +title\_list[i] + ".gif"  
 print(filename+":"+imgurl) #打印下载信息  
 urllib.request.urlretrieve(imgurl,filename) #下载图片

**DESIGN THE DATABASE:-**

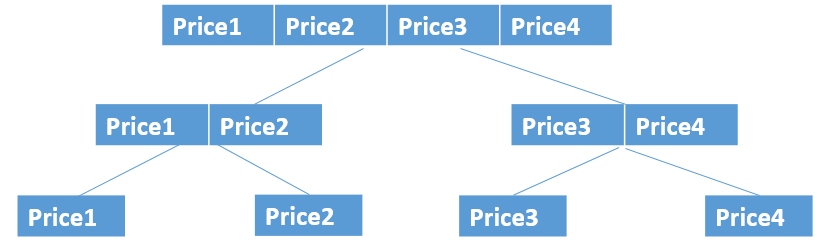
Analyze information

* We will use quick sort to deal with the information we’ve grabbed.
* Then we will generate multiple choices for users.
* For example:

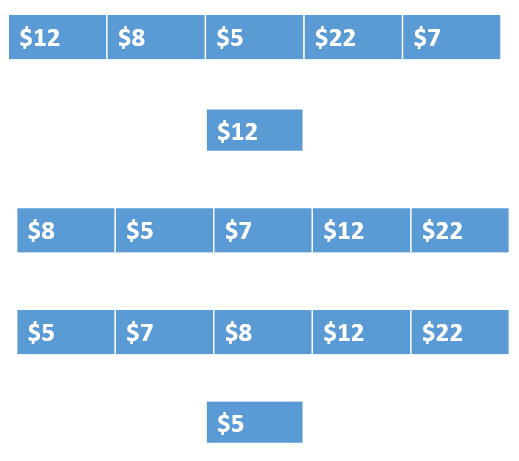


Algorithm

* A user submit a shopping sheet, then we start sorting them using quick sort algorithm.



* Quicksort is a divide and conquer algorithm.
* Quicksort first divides a large array into two smaller sub-arrays: the low elements and the high elements. Quicksort can then recursively sort the sub-arrays.
* First we generate an array according to user’s shopping sheet.
* Then we pick one price from the array as a pivot.
* Then we reorder the array, less than the pivot come before the pivot, and vice versa.
* Then we do this job recursively until the array is sorted.
* Finally we recommend our user to choose the first one of the array as the best choice.



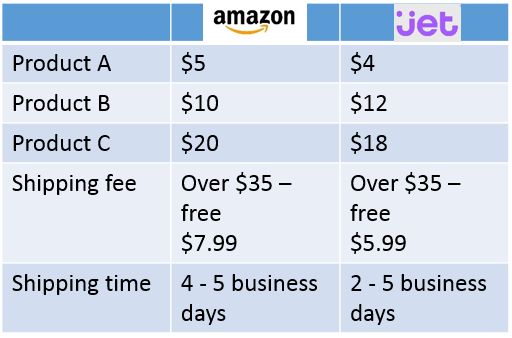
Analyze information

* If a user want to buy 2\*Product B and 1\*Product, then:

Best price: buy all stuff on Amazon, $32.99 in total.

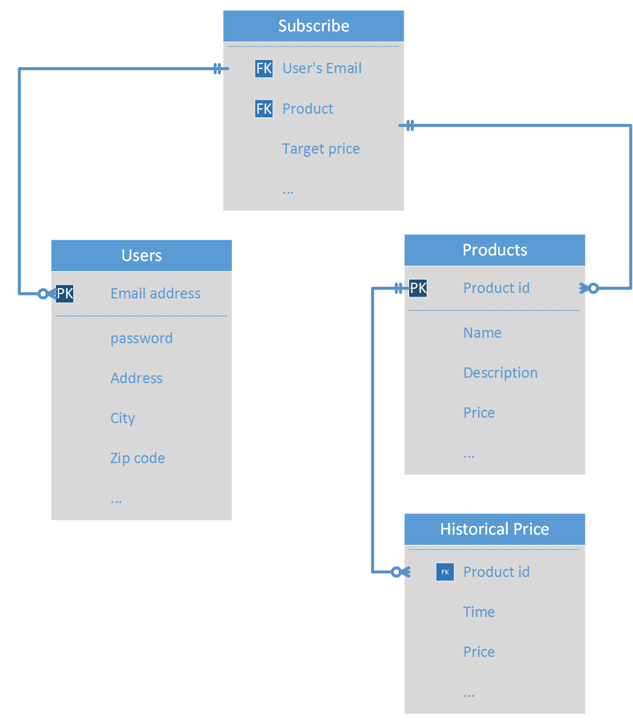
Least waiting time: buy all stuff on Jet, $33.99 in total, and wait for only 2-5 business days.

* This is just a simple example. In reality, we will consider more aspects, such as Amazon’s prime membership, location of users and other special conditions.



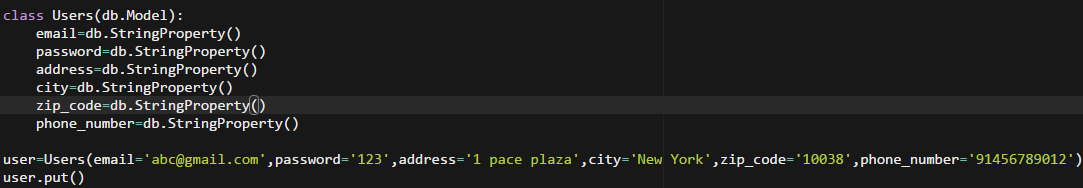
Database

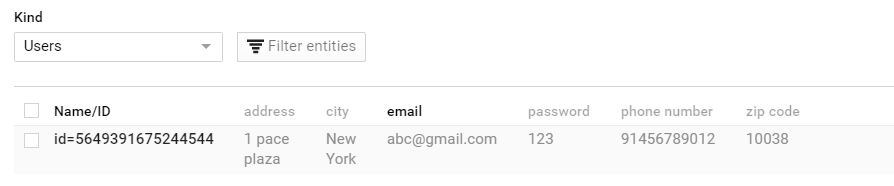
* We will create several tables, like Users, Products, Subscribe, Historical Price, and more in future.



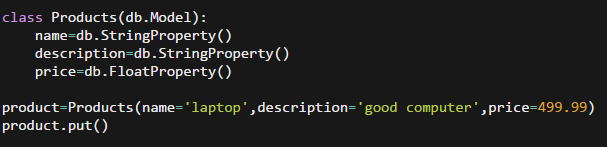
Creating database:

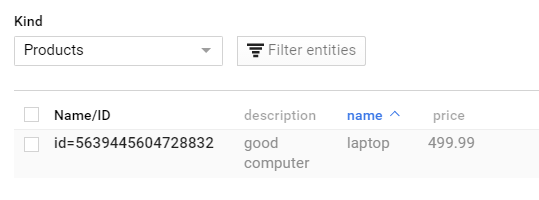
Users:



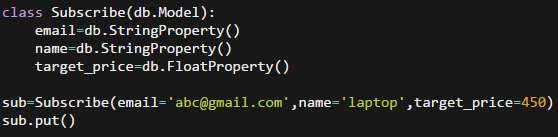


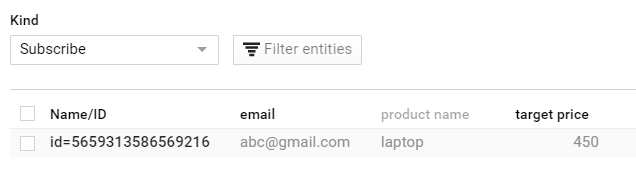
Products





Subscribe:





Historical Price:

