Development Testing

Obtaining all of the customer's order items/ Outputting data from the database

I had a problem doing this as it was not just a simple case of orderitems = OrderItem.objects.filter() as there is no direct relationship between the customer and their order items, it exists through the Order table. So I first obtained the Orders and then looped through the orders, retrieving the order items for that order and created a python list of the order items.

```
                    'orders': orders,})
#
#
#
#renders the customer detail page
def viewcustomer(request, id, slug):
        customer = get_object_or_404(MyUser,
                                     id = id,
                                     customerslug = slug)

        orders = Order.objects.filter(customer_id = customer.id)
        allOrderItems = []
        for order in orders:
                orderitems = OrderItem.objects.filter(order_id = order.id)
                allOrderItems.append(orderitems)

        return render(request,
```

Here is what my code then looked like.

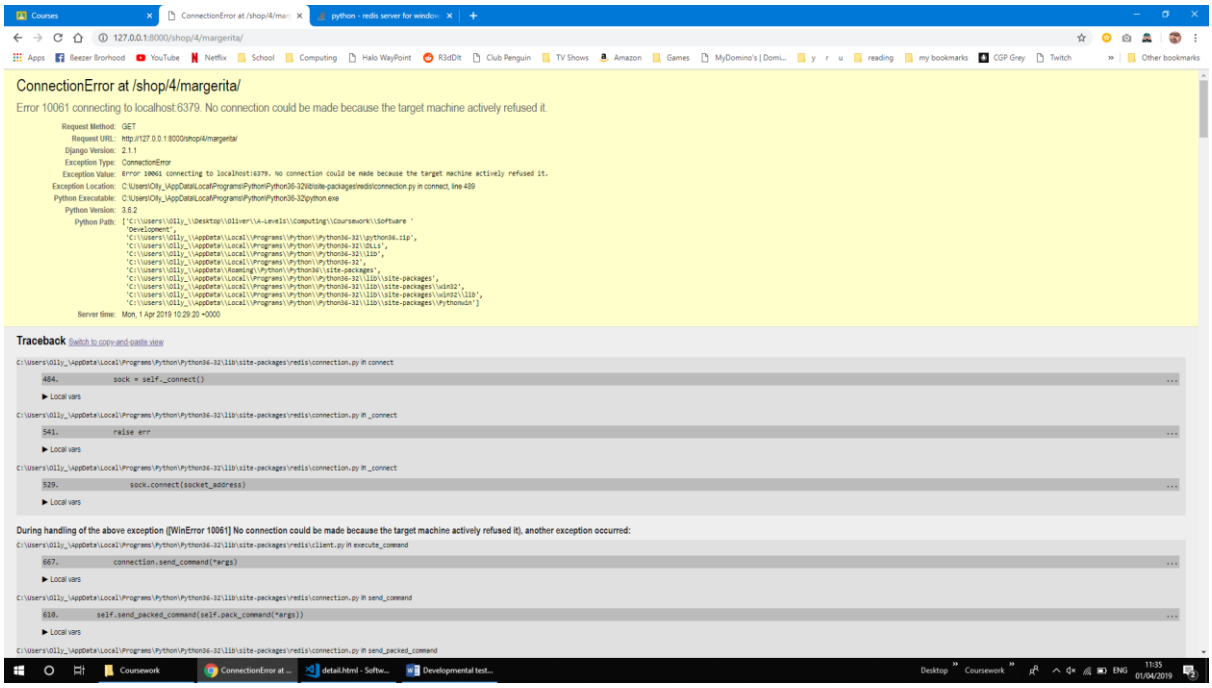
How ever this gave me an error when rendering the html template as it would only accept a tuple to be passed through. I got around this by switching to a tuple but then I could not add between the new order items for each order as tuples are immutable.

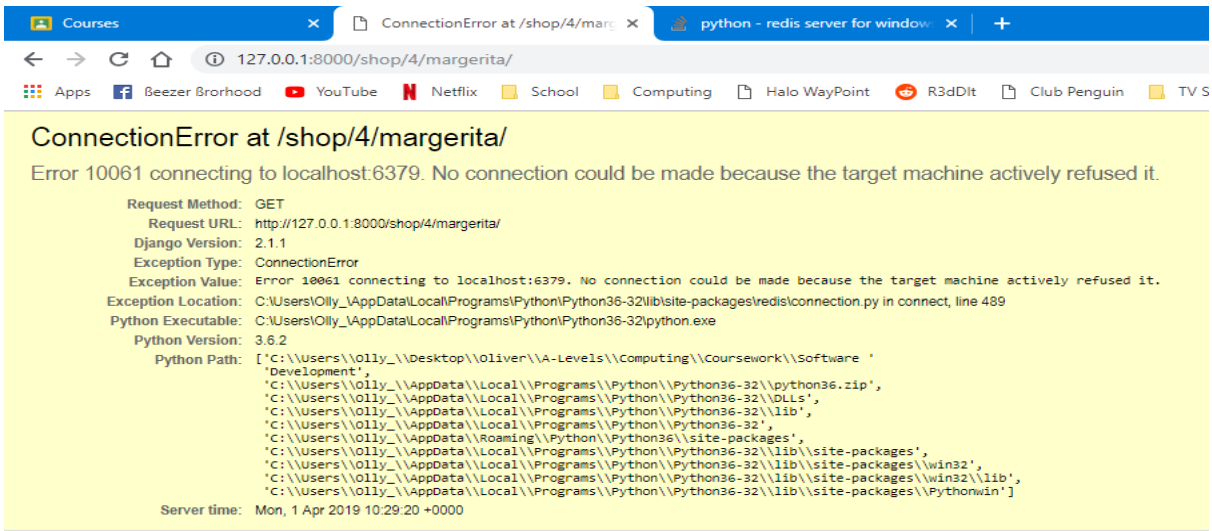
I got around this by using an array during the for loop, as they are mutable and then changing it to a tuple data type before passing it through. This allowed me to use the functionality of both that I needed.

```
inal  Help                    • views.py - Software Development - Visual Studio Code
views.py •
        return(viewcustomers(request))

#renders the customer detail page
def viewcustomer(request, id, slug):
        customer = get_object_or_404(MyUser,
                                     id = id,
                                     customerslug = slug)

        orders = Order.objects.filter(customer_id = customer.id)
        allOrderItems = []#originally list but could not pass through properly,
        #changing it to tuple caused another proplem so i changed it back and converted to tuple just before i passed it-+
        for order in orders:
                orderitems = OrderItem.objects.filter(order_id = order.id)
                allOrderItems.append(orderitems)

        allOrderItemsTuple = tuple(allOrderItems)#the art of the bodge
        return render(request,
                      'staff/viewcustomer.html',
                      {'customer': customer,
                      'allOrderItemsTuple': allOrderItemsTuple,
                      'orders': orders,})

#this function orders the customers based off the staff member's django form input
```

Creating the recommendation engine

In order to create the recommendation engine I used a library called redis. This library helps to find which products are bought the most with the current product that the customer is looking at of the items in the cart. However when I first ran it to see if it worked, I got an error message when I viewed any product page or the cart detail page. This was because I had a problem with redis.

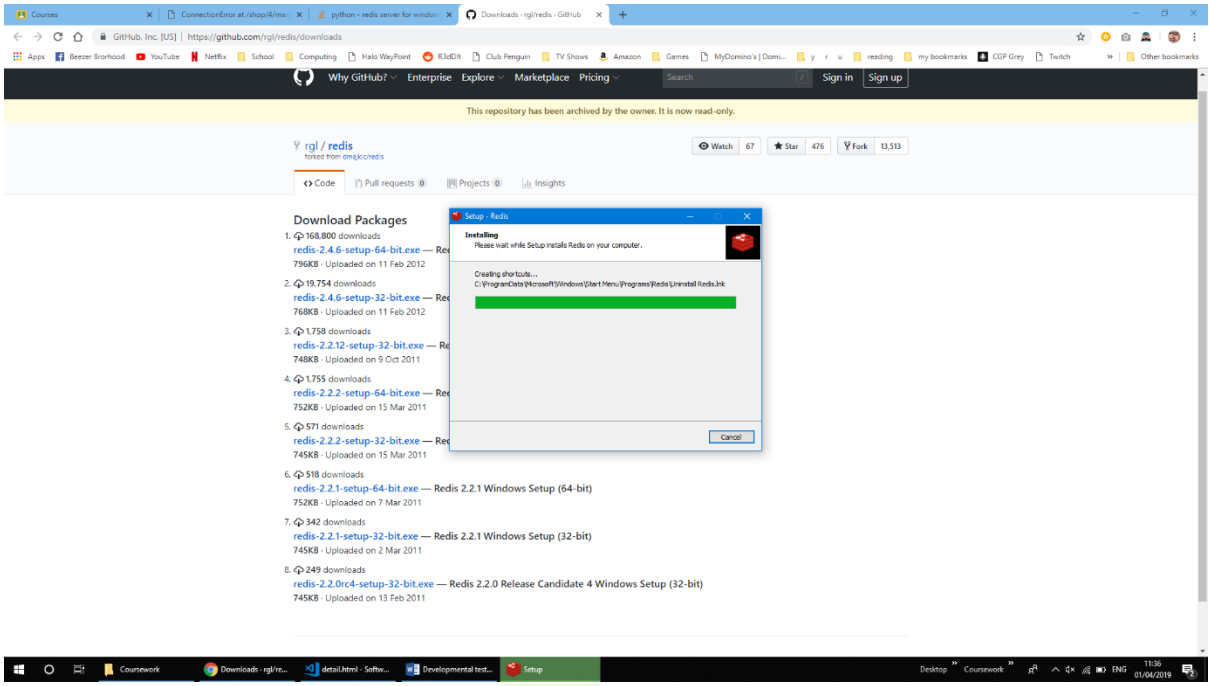




I received this error because the redis library needed to run a redis server to function properly. I had no idea about this however at the time, I thought there could have been a problem with my machine or my install of the redis library itself.

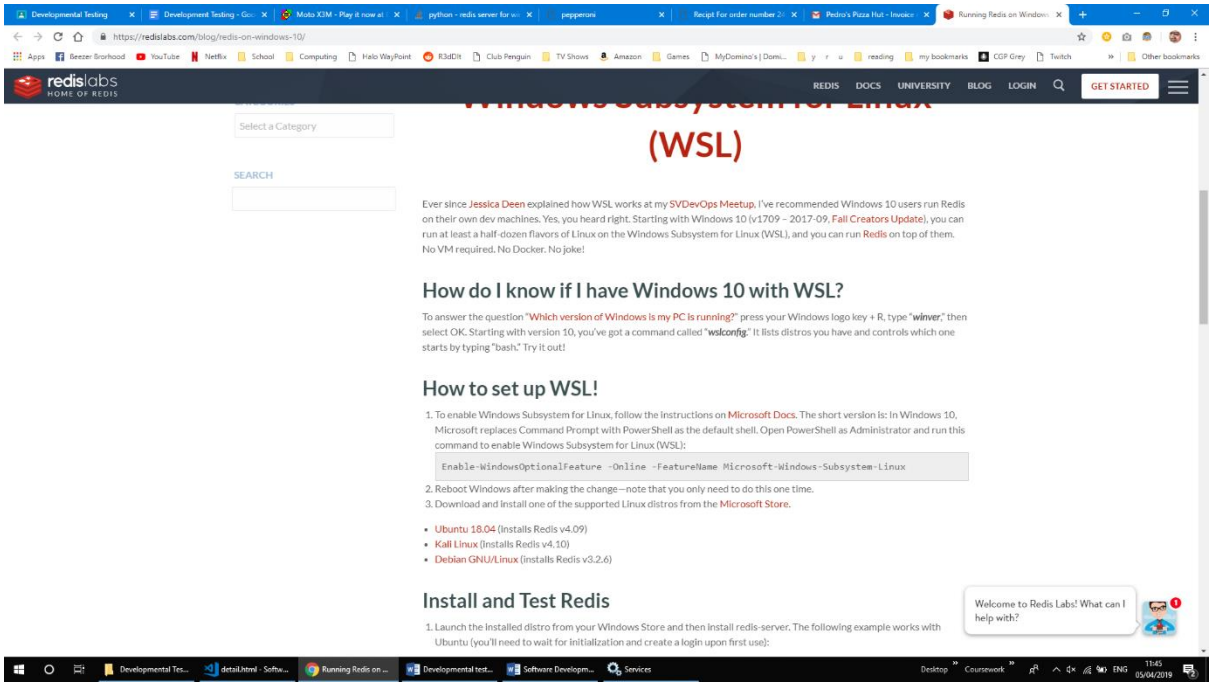
I found a stack overflow question with a similar problem, https://stackoverflow.com/questions/24129640/redis-server-for-windows-with-use-for-python3

This directed me to a different install of redis on github which I then installed.

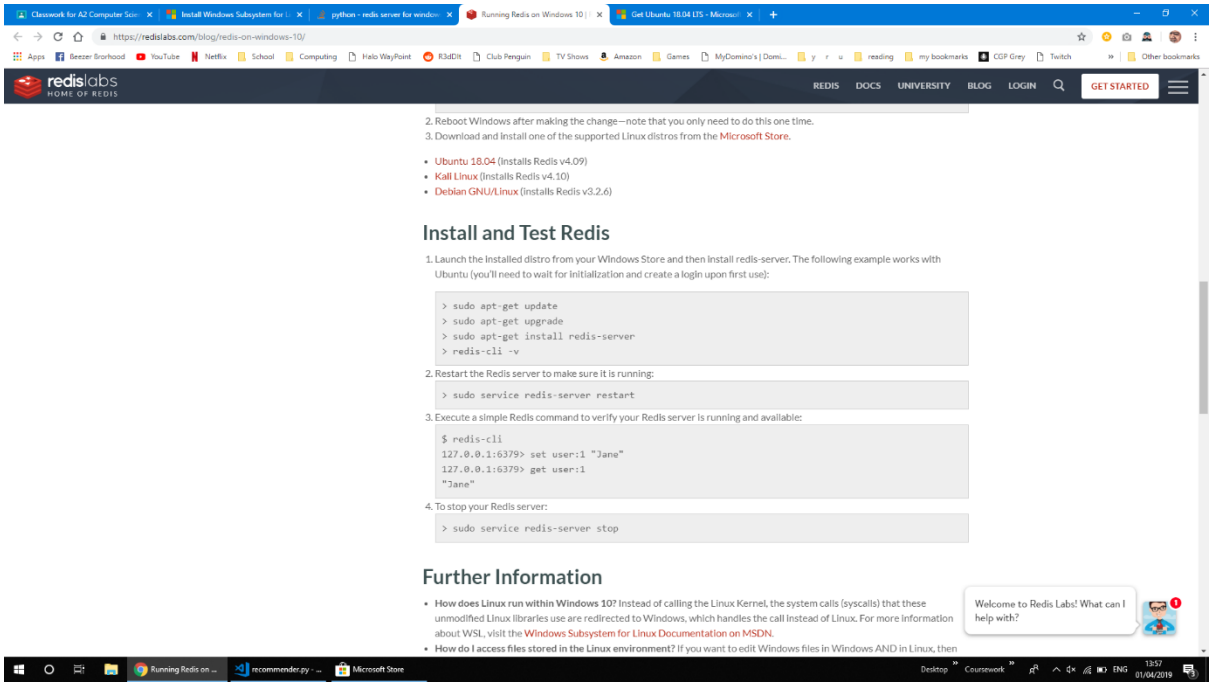


Redis is designed for Linux and because of that it is far easier to use on Linux. To run a redis server I found out that you need to run at least the server on Linux. I do not have a Linux machine so I was worried that I would have to find another way to make the recommendation engine.

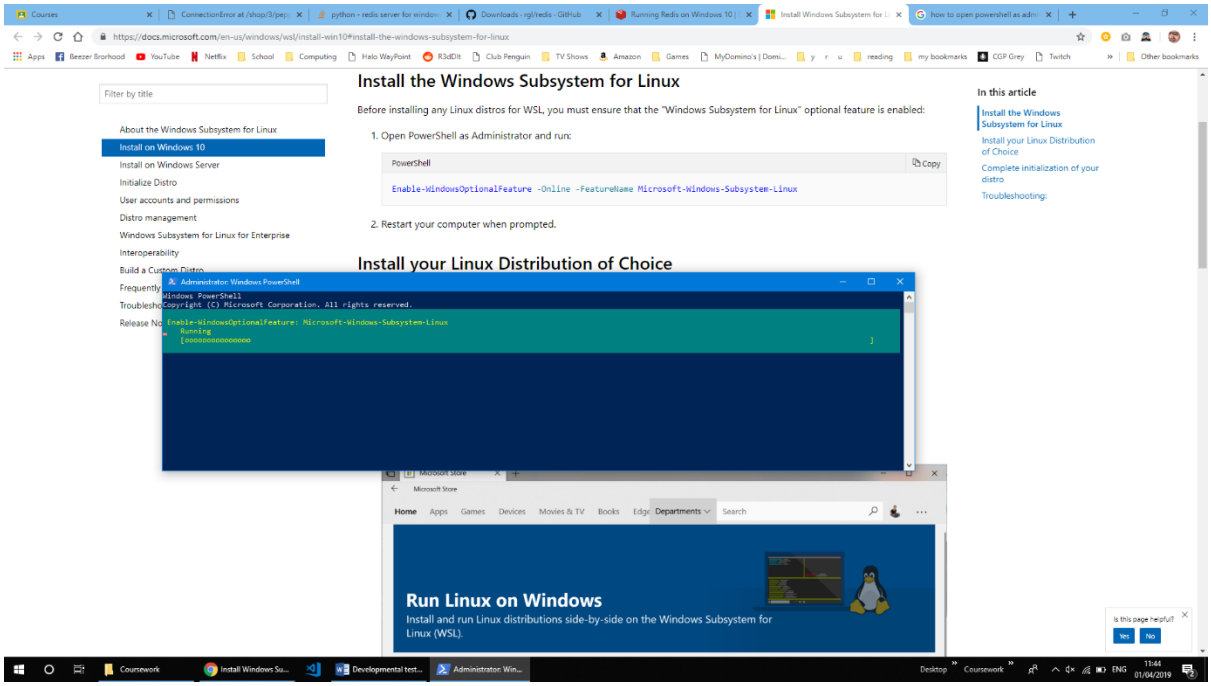
I found a way to run a redis server on my windows 10 pc, by installing a Windows Subsystem for Linux or a WSL. I did this by following this article: https://redislabs.com/blog/redis-on-windows-10/



The above and below screenshots are of that article.

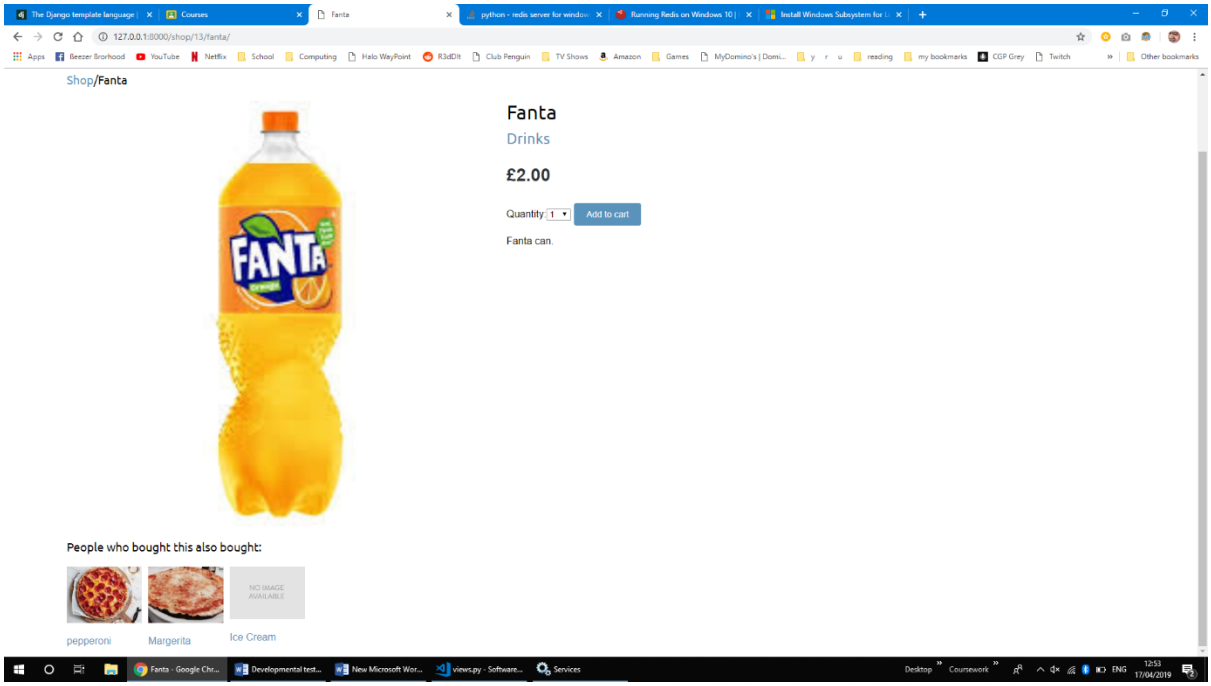


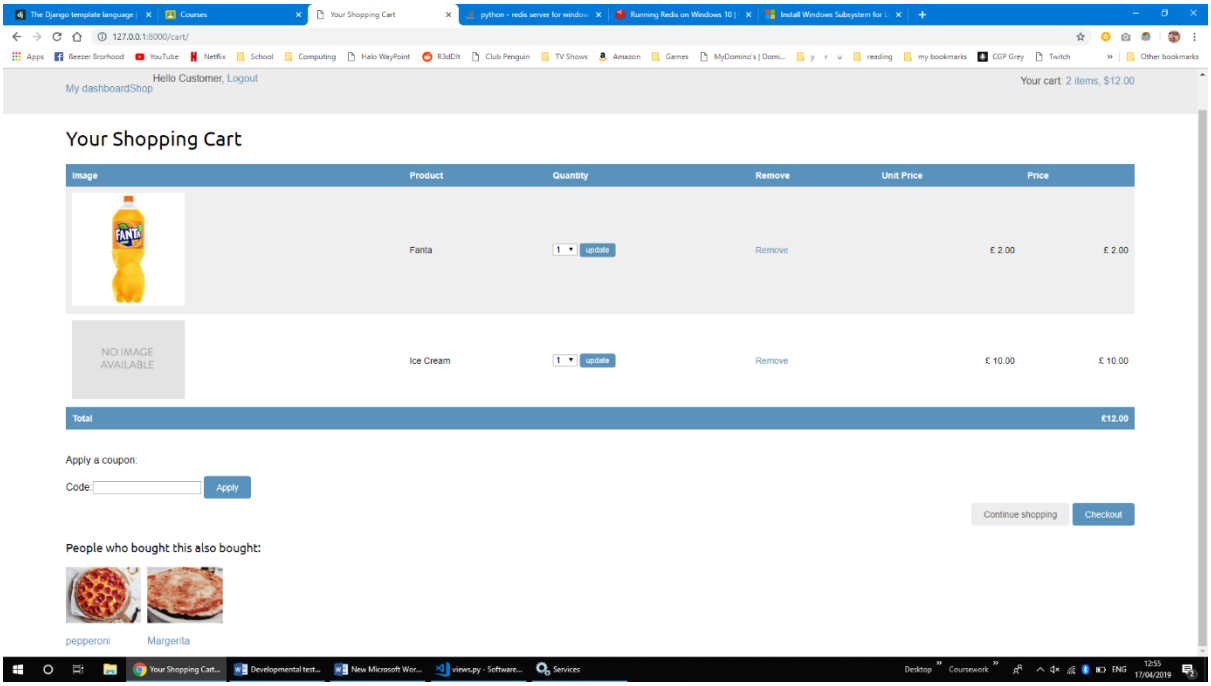
But I had to enable the optional feature to have a WSL. I found out how to do this through the Microsoft website: https://docs.microsoft.com/en-us/windows/wsl/install-win10#install-the-windows-subsystem-for-linux



That fixed my problem, and allowed me to run a redis server on my laptop.

Below are screenshots showing redis displaying the recommended products in a product detail page and the cart detail page.

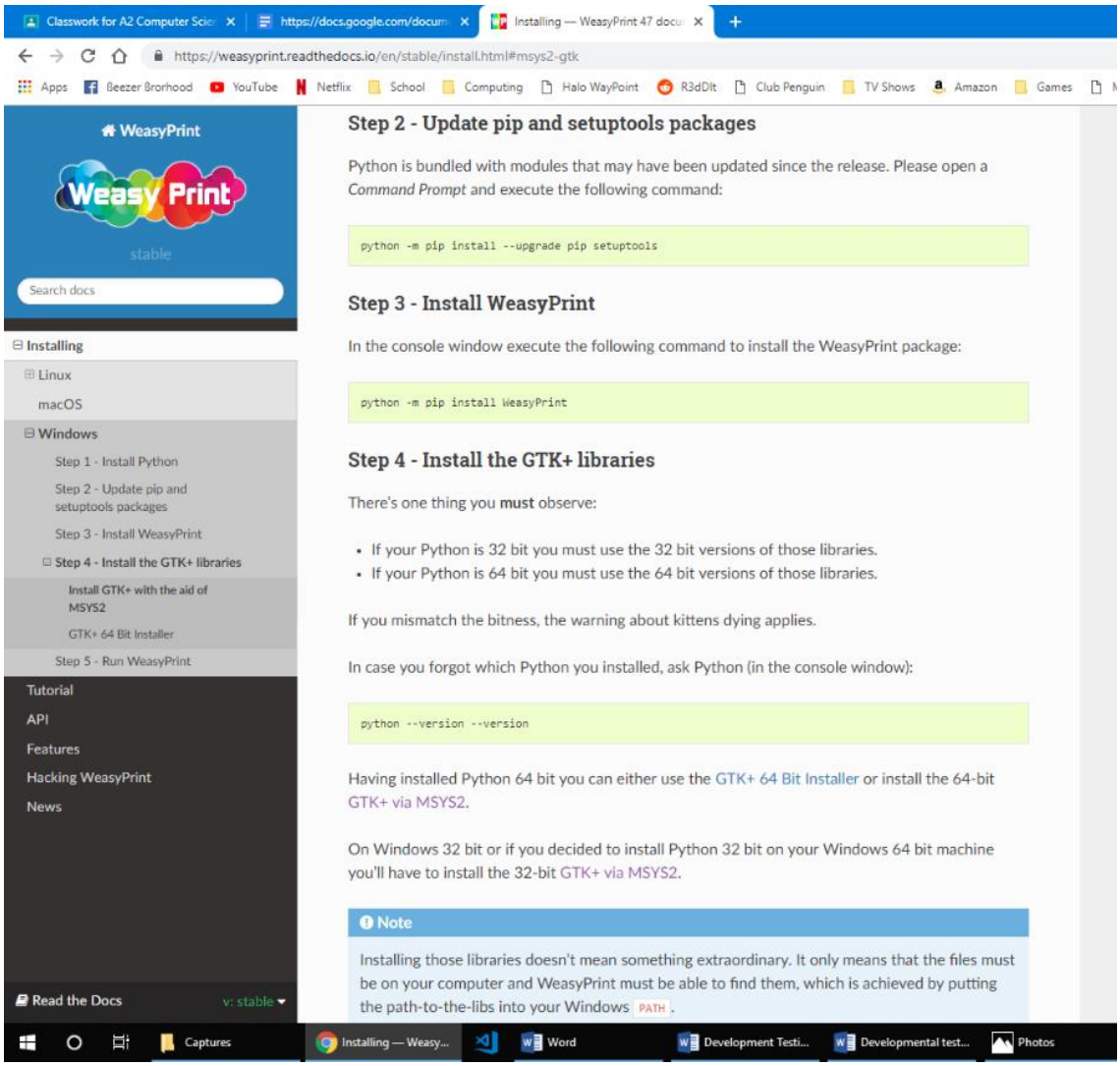

Shop/Fanta
Fanta
Drinks
£2.00
Quantity: 1
Add to cart
Fanta can.
People who bought this also bought:
NO IMAGE AVAILABLE
pepperoni
Margerita
Ice Cream


Hello Customer, Logout
My dashboardShop
Your cart: 2 items, $12.00
Your Shopping Cart
Image
Product
Quantity
Remove
Unit Price
Price
Fanta
1
update
Remove
£ 2.00
£ 2.00
NO IMAGE AVAILABLE
Ice Cream
1
update
Remove
£ 10.00
£ 10.00
Total
£12.00
Apply a coupon:
Code:
Apply
Continue shopping
Checkout
People who bought this also bought:
pepperoni
Margerita

Creating PDFs

This problem took me multiple days to complete, to create pdfs I chose weasyprint. Weasyprint required a much more complex install before using pip install, I had to follow this page https://weasyprint.readthedocs.io/en/stable/install.html#msys2-gtk in order to do so.

Below are screenshots of the main steps I had to follow, for example I didn't screenshot installing python.



I had to install the GTX+ libraries because WeasyPrint uses them to create the pdf files.

Classwork for A2 Computer Scie | https://docs.google.com/docum | Installing — WeasyPrint 47 docu

https://weasyprint.readthedocs.io/en/stable/install.html#msys2-gtk

Apps | Beezer Brorhood | YouTube | Netflix | School | Computing | Halo WayPoint | R3dDit | Club Penguin | TV Shows | Amazon | Games

WeasyPrint

stable

Search docs

- Installing
  - Linux
  - macOS
  - Windows
    - Step 1 - Install Python
    - Step 2 - Update pip and setuptools packages
    - Step 3 - Install WeasyPrint
    - Step 4 - Install the GTK+ libraries
      - Install GTK+ with the aid of MSYS2
      - GTK+ 64 Bit Installer
    - Step 5 - Run WeasyPrint
- Tutorial
- API
- Features
- Hacking WeasyPrint
- News

Read the Docs v: stable

Install GTK+ with the aid of MSYS2

Sadly the GTK+ Runtime for 32 bit Windows was discontinued in April 2017. Since then developers are advised to either bundle GTK+ with their software (which is beyond the capacities of the WeasyPrint maintainers) or install it through the MSYS2 project.

With the help of MSYS2, both the 32 bit as well as the 64 bit GTK+ can be installed. If you installed the 64 bit Python and don't want to bother with MSYS2, then go ahead and use the GTK+ 64 Bit Installer.

MSYS2 is a development environment. We (somehow) mis-use it to only supply the up-to-date GTK+ runtime library files in a subfolder we can inject into our `PATH`. But maybe you get interested in the full powers of MSYS2. It's the perfect tool for experimenting with MinGW and cross-platform development – look at its wiki.

Ok, let's install GTK3+.

- Download and run the MSYS2 installer
  - On 32 bit Windows: "msys2-**i686**-xxxxxxxx.exe"
  - On 64 bit Windows: "msys2-**x86_64**-xxxxxxxx.exe"

  You alternatively may download a zipped archive, unpack it and run `msys2_shell.cmd` as described in the MSYS2 wiki.
- Update the MSYS2 shell with

```
pacman -Syuu
```

  Close the shell by clicking the close button in the upper right corner of the window.
- Restart the MSYS2 shell. Repeat the command

```
pacman -Su
```

  until it says that there are no more packages to update.
- Install the GTK+ package and its dependencies.

  To install the 32 bit (**i686**) GTK run the following command:

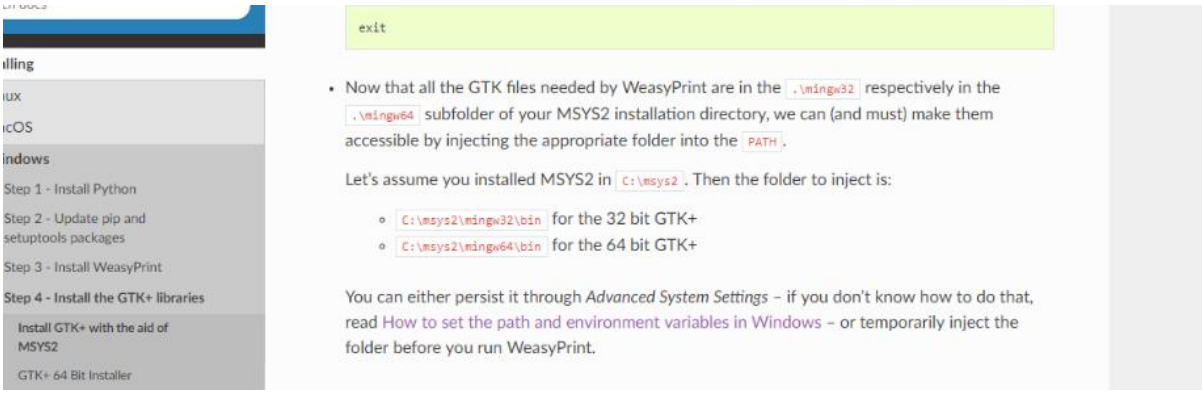
```
pacman -S mingw-w64-i686-gtk3
```

Captures | Installing — Weasy... | Developmental test... | Development Testi... | Photos

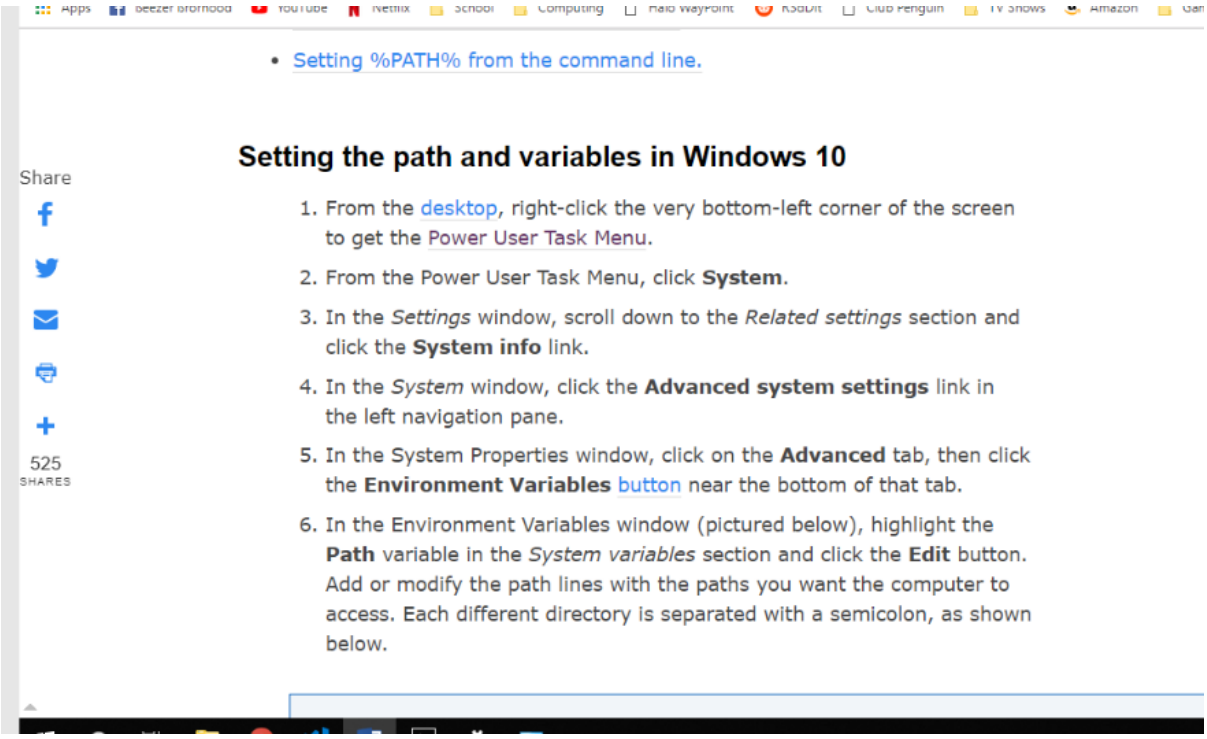
These steps installed called MSYS2. MSYS2 is software distribution and a building platform for Windows. It provides a Unix-like environment, a command-line interface and a software repository making it easier to install, use, build and port software on Windows.

```
Olly_@OliverPC MSYS ~
$ pacman -S mingw-w64-i686-gtk3
warning: mingw-w64-i686-gtk3-3.22.26-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) mingw-w64-i686-gtk3-3.22.26-1

Total Installed Size:  73.27 MiB
Net Upgrade Size:       0.00 MiB

:: Proceed with installation? [Y/n]
```

After completing that, I had finally installed all of the libraries that I needed. I just had one more step to do from the tutorial which was to inject the folder that contained the GTK files into the PATH. At the time I had no idea what this was so I had to go and learn about it.



I found out that injecting the folder into the PATH would allow weasyprint to access it.



Once I had injected the folder, I then had to pip install weasyprint again in the command line, but I had another problem. One of the libraries that I had installed, which weasyprint was dependant was unable to be installed.

Below is the screenshot of the error that I received. Once I received this error after days of trying to get weasyprint to work I decided to simply cut my losses with weasyprint and restart pdfs in Django from the ground up as weasyprint was taking too long.

Terminal Help

Welcome - prototype - Visual Studio Code

Welcome

Start

Customize

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: cmd

```
  building 'cairo._cairo' extension
  creating build\temp.win32-3.6
  creating build\temp.win32-3.6\Release
  creating build\temp.win32-3.6\Release\cairo
  C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\bin\HostX86\x86\cl.exe /c /nologo /Ox /W3 /GL /DNDEBUG /MT -DPYCAIRO_VERSION_MAJOR=1 -DPYCAIRO_VERSION_MINOR=18 -DPYCAIRO_VERSION_
MICRO=0 -Ic:\users\olly_\appdata\local\programs\python\python36-32\include -Ic:\users\olly_\appdata\local\programs\python\python36-32\include "-IC:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC
\14.16.27023\Include" "-IC:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\ATLMFC\Include" "-IC:\Program Files (x86)\Windows Kits\10\include\10.0.17763.0\shared" "-IC:\Program Files
(x86)\Windows Kits\10\include\10.0.17763.0\um" "-IC:\Program Files (x86)\Windows Kits\10\include\10.0.17763.0\winrt" "-IC:\Program Files (x86)\Windows Kits\10\include\10.0.17763.0\ucrt" "-IC:\Program Files (x86)\Windows
Kits\NETFXSDK\4.6.1\include\um" /Tccairo/device.c /Fobuild\temp.win32-3.6\Release\cairo/device.obj
  device.c
  c:\users\olly_\appdata\local\temp\pip-install-lkgrdejh\pycairo\cairo\pycairo.h(37): fatal error C1083: Cannot open include file: 'cairo.h': No such file or directory
  error: command 'C:\\Program Files (x86)\\Microsoft Visual Studio\\2017\\Community\\VC\\Tools\\MSVC\\14.16.27023\\bin\\HostX86\\x86\\cl.exe' failed with exit status 2

  ----------------------------------------
  Failed building wheel for pycairo
  Running setup.py clean for pycairo
Failed to build pycairo
Installing collected packages: pycairo
  Running setup.py install for pycairo ... error
    Complete output from command c:\users\olly_\appdata\local\programs\python\python36-32\python.exe -u -c "import setuptools, tokenize;__file__='C:\\Users\\Olly_\\AppData\\Local\\Temp\\pip-install-lkgrdejh\\pycairo\\set
up.py';f=getattr(tokenize, 'open', open)(__file__);code=f.read().replace('\r\n', '\n');f.close();exec(compile(code, __file__, 'exec'))" install --record C:\Users\Olly_\AppData\Local\Temp\pip-record-4_0tt4cv\install-recor
d.txt --single-version-externally-managed --compile:
    running install
    running build
    running build_py
    creating build
    creating build\lib.win32-3.6
    creating build\lib.win32-3.6\cairo
    copying cairo\__init__.py -> build\lib.win32-3.6\cairo
    copying cairo\__init__.pyi -> build\lib.win32-3.6\cairo
    copying cairo\py.typed -> build\lib.win32-3.6\cairo
    running build_ext
    building 'cairo._cairo' extension
    creating build\temp.win32-3.6
    creating build\temp.win32-3.6\Release
    creating build\temp.win32-3.6\Release\cairo
    C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\bin\HostX86\x86\cl.exe /c /nologo /Ox /W3 /GL /DNDEBUG /MT -DPYCAIRO_VERSION_MAJOR=1 -DPYCAIRO_VERSION_MINOR=18 -DPYCAIRO_VERSIO
N_MICRO=0 -Ic:\users\olly_\appdata\local\programs\python\python36-32\include -Ic:\users\olly_\appdata\local\programs\python\python36-32\include "-IC:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MS
VC\14.16.27023\Include" "-IC:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\ATLMFC\Include" "-IC:\Program Files (x86)\Windows Kits\10\include\10.0.17763.0\shared" "-IC:\Program File
s (x86)\Windows Kits\10\include\10.0.17763.0\um" "-IC:\Program Files (x86)\Windows Kits\10\include\10.0.17763.0\winrt" "-IC:\Program Files (x86)\Windows Kits\10\include\10.0.17763.0\ucrt" "-IC:\Program Files (x86)\Window
s Kits\NETFXSDK\4.6.1\include\um" /Tccairo/device.c /Fobuild\temp.win32-3.6\Release\cairo/device.obj
    device.c
    c:\users\olly_\appdata\local\temp\pip-install-lkgrdejh\pycairo\cairo\pycairo.h(37): fatal error C1083: Cannot open include file: 'cairo.h': No such file or directory
    error: command 'C:\\Program Files (x86)\\Microsoft Visual Studio\\2017\\Community\\VC\\Tools\\MSVC\\14.16.27023\\bin\\HostX86\\x86\\cl.exe' failed with exit status 2

    ----------------------------------------
Command "c:\users\olly_\appdata\local\programs\python\python36-32\python.exe -u -c "import setuptools, tokenize;__file__='C:\\Users\\Olly_\\AppData\\Local\\Temp\\pip-install-lkgrdejh\\pycairo\\setup.py';f=getattr(tokeniz
e, 'open', open)(__file__);code=f.read().replace('\r\n', '\n');f.close();exec(compile(code, __file__, 'exec'))" install --record C:\Users\Olly_\AppData\Local\Temp\pip-record-4_0tt4cv\install-record.txt --single-version-e
xternally-managed --compile" failed with error code 1 in C:\Users\Olly_\AppData\Local\Temp\pip-install-lkgrdejh\pycairo\

C:\Users\Olly_\Desktop\Oliver\A-Levels\Computing\Coursework\Prototype\Prototype2\prototype>
```

After much searching, I found a new library called xhtml2pdf which uses html 4 instead. This method of creating a pdf file I Django uses a different approach from weasyprint. It uses one library, xhtml2pdf and then you, the programmer create the function render_to_pdf function yourself in a new file called utils.py and that function is called whenever you need to create a pdf. Below is my render_to_pdf function:

```
from io import BytesIO
from django.http import HttpResponse
from django.template.loader import get_template

from xhtml2pdf import pisa

def render_to_pdf(template_src, context_dict={}):
    template = get_template(template_src)
    html  = template.render(context_dict)
    result = BytesIO()
    pdf = pisa.pisaDocument(BytesIO(html.encode("ISO-8859-1")), result)
    if not pdf.err:
        return HttpResponse(result.getvalue(), content_type='application/pdf')
    return None
```

This was a much simpler path as it was far easier to use & interact with this third-party library.

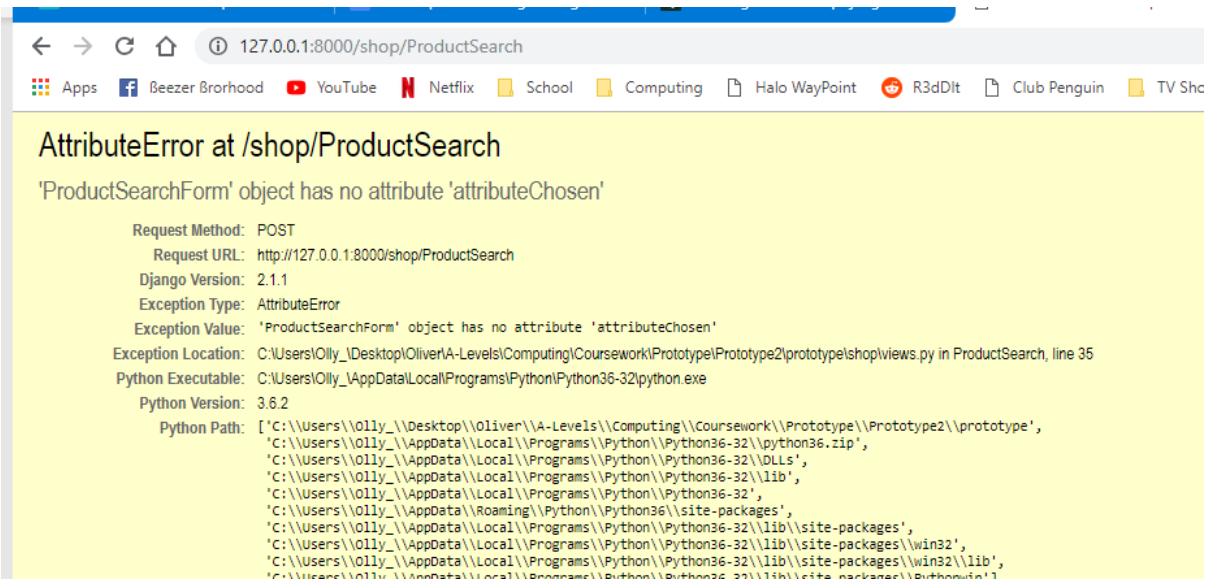
For each of the html template files that held the pdf templates, I had to have the css for that file to be infile otherwise it would not execute properly once rendered as a pdf. This is only a minor problem that can be ignored as it will only potentially cause problems whenever you want to edit the css, and have to do it for each file. I also had to specify which html version I was using in the DOCTYPE tag to make sure it was using HTML4 because HTML5 does not support this library.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Obtaining user Input

```
def ProductSearch(request, category_slug=None):
        if request.method == 'POST':
                form = ProductSearchForm(request.POST)
                if form.is_valid():
                        if form.attributeChosen == '0':
                                products = Product.objects.filter(available = True, id = form.searchCriteria)
                        elif form.attributeChosen == '1':
                                products = Product.objects.filter(available = True, name = form.searchCriteria)
                        form = ProductSearchForm(request.POST)
                        category = None
                        categories = Category.objects.all()
                        if category_slug:
```

When I first started using Django I struggled with obtaining user input. This was because it required a complex process of passing through the user data through html and Django forms which I did not fully understand. I first tried the mthod above but I received an error:



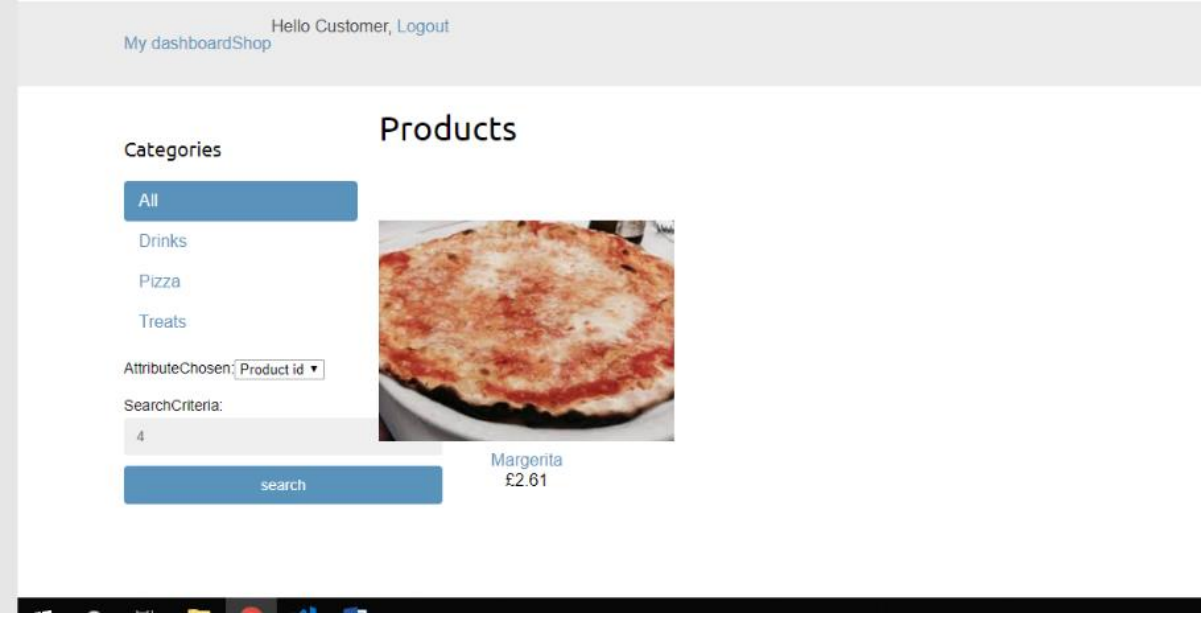
I learnt that you cannot access the data in forms directly, which I thought you could do as it is possible to save a form to a database directly, '{objectName} = form.save()'. Each form normalises the data entered by the user into a consistent format, relative to the format of the field, this always data of a field to be entered in multiple different ways but always resulting in a consistent output. This data is called the clean data. So because of this you access the data of a form differently.

Each form has a 'cleaned_data' attribute and it is through this that you can access user input. Below is the new code that works.

```
def ProductSearch(request, category_slug=None):
        if request.method == 'POST':
                form = ProductSearchForm(request.POST)
                if form.is_valid():
                        cd = form.cleaned_data#creates a dictionary of the
form's cleaned data
                        if cd["attributeChosen"] == '0':
                                products = Product.objects.filter(available =
True, id = cd['searchCriteria'])
                        elif cd["attributeChosen"] == '1':
```

```
                                products = Product.objects.filter(available =
True, name = cd['searchCriteria'])
                            form = ProductSearchForm(request.POST)
```

What this code does differently is that it creates a dictionary of the form's cleaned data. This allows me to access the clean data of the form which I can then actually use. Below is a screenshot showing that the data of the form was able to be used to output what the user searched for.



Using different objects in each other's methods

Whenever I was creating methods for data processing for my objects, I ran into some problems. The situation was that I in order to make a method for object A, I needed to import object B. However, some of object B's methods required me to import object A. This resulted in the error below when I tried to import object B for object A.

```
    raise _exception[1]
  File "C:\Users\Olly_\AppData\Local\Programs\Python\Python36-32\lib\site-packages\django\core\management\__init__.py", line 337, in execute
    autoreload.check_errors(django.setup)()
  File "C:\Users\Olly_\AppData\Local\Programs\Python\Python36-32\lib\site-packages\django\utils\autoreload.py", line 225, in wrapper
    fn(*args, **kwargs)
  File "C:\Users\Olly_\AppData\Local\Programs\Python\Python36-32\lib\site-packages\django\__init__.py", line 24, in setup
    apps.populate(settings.INSTALLED_APPS)
  File "C:\Users\Olly_\AppData\Local\Programs\Python\Python36-32\lib\site-packages\django\apps\registry.py", line 112, in populate
    app_config.import_models()
  File "C:\Users\Olly_\AppData\Local\Programs\Python\Python36-32\lib\site-packages\django\apps\config.py", line 198, in import_models
    self.models_module = import_module(models_module_name)
  File "C:\Users\Olly_\AppData\Local\Programs\Python\Python36-32\lib\importlib\__init__.py", line 126, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 978, in _gcd_import
  File "<frozen importlib._bootstrap>", line 961, in _find_and_load
  File "<frozen importlib._bootstrap>", line 950, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 655, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 678, in exec_module
  File "<frozen importlib._bootstrap>", line 205, in _call_with_frames_removed
  File "C:\Users\Olly_\Desktop\Oliver\A-Levels\Computing\Coursework\Software Development\account\models.py", line 7, in <module>
    from orders.models import Order
  File "C:\Users\Olly_\Desktop\Oliver\A-Levels\Computing\Coursework\Software Development\orders\models.py", line 3, in <module>
    from account.models import MyUser
ImportError: cannot import name 'MyUser'
```

I found out that the reason I could not import object B for object A to use was that I was creating a circular import. At first I looked at merging the two modules together to fix this circular dependency. I started to plan it out but quickly realised that this would be a major change to my code, and design of my system so I decided to find another way.

I researched more into circular imports and found this:

When Python imports a module, it checks the module registry to see if the module was already imported. If the module was already registered, Python uses that existing object from cache. The module registry is a table of modules that have been initialized and indexed by module name. This table can be accessed through `sys.modules`.

If it was not registered, Python finds the module, initializes it if necessary, and executes it in the new module's namespace.

In our example, when Python reaches `import module2`, it loads and executes it. However, module2 also calls for module1, which in turn defines `function1()`.

The problem occurs when `function2()` tries to call module1's `function3()`. Since module1 was loaded first, and in turn loaded module2 before it could reach `function3()`, that function isn't yet defined and throws an error when called:

```
$ python __init__.py
```

This made me realise that if, for example, object B did not import object A whenever object B was being imported, as the circular import only occurred whenever the module being import was initialised, that I could solve this problem by importing object A into object B **after** it had been initialised.

Here is the code I had before:

```
from django.db import models
from django.contrib.auth.models import AbstractUser
import datetime
from django.urls import reverse
#used in accountAge method
import arrow
from orders.models import Order

class MyUser(AbstractUser):
    gender = models.BooleanField(default = True)
    password = models.CharField(max_length=50)
```

Since the methods of an object are not executed when the object is imported, I move the 'form orders.models import Oder' line to a method, so that it will not be executed until it is called meaning there will be no import error.

Here is my new code:

```
def getCusOrders(self, cusOrderType = 0):
        from orders.models import Order#importing Order class here because
when I
        #imported it at the top i had a circular import error, solved using
link that is now bookmarked
        if cusOrderType == 0:
            orders = Order.objects.filter(customer_id = self.id)
        else:
            orders = Order.objects.filter(customer_id = self.id, orderType =
cusOrderType)

        return orders
```

To avoid duplication of my code in each object that I needed to do this, I have a method that does this and contains the import line, for example in this object that I have shown I have re used this method multiple times:

```
def averageDiffInOrdersDays(self):
        orders = self.getCusOrders()
```

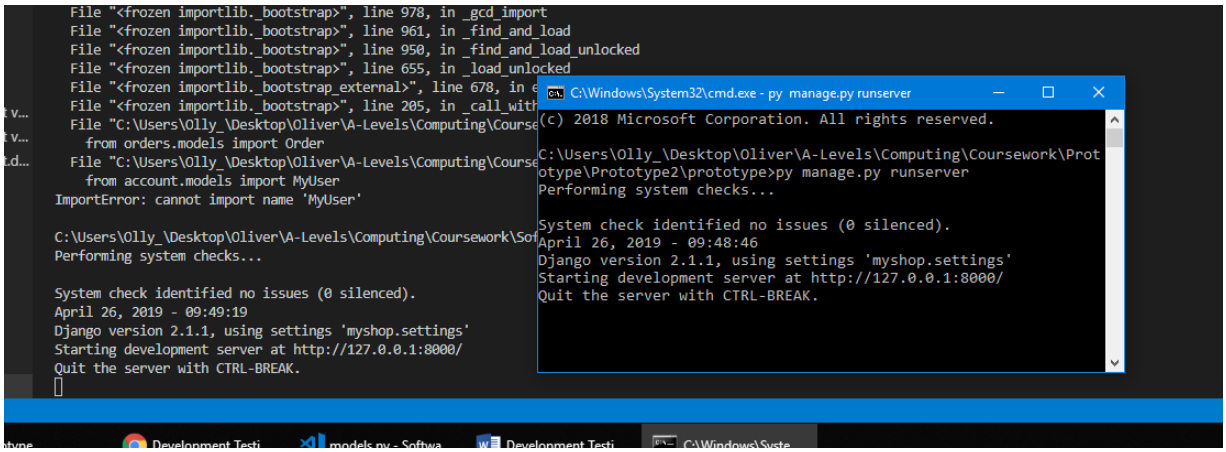
```
def totalSpent(self):
        costOfOrders = 0
        orders = self.getCusOrders()
```

```
def checkIfTimeSinceLastOrderIsAboveAverage(self):
        orders = self.getCusOrders()
```

Access Levels

In Django the standard for creating a site with different access levels is to have the code that you write as the standard user site, and then use Django's built in background admin site for the admin access level. This however was not good enough for my system because I had multiple different access levels, customer, staff, driver, cook and manager.

My original plan was to create different account apps for each different account so that the user would go to a different url to login. I thought that this would be a relatively simple solution to the problem as this would keep the account types completely separate.



This turned out to be a much more complex than I originally though as I would need to interact with different servers early on while I was still learning how to use Django, so I decided to find another way.

My new approach was to use Boolean flags in the user model & if statements in the html templates, and to make use of Django decorators. A decorator checks user roles and permissions.

Below I shall show examples of how I used these:

```
@login_required
def dashboard(request):
    prevorders = Order.objects.filter(customer_id = request.user.id)
    savedorders = Order.objects.filter(customer_id = request.user.id, saved =
True)

    return render(request,
            'account/dashboard.html',
            {'section': 'dashboard',
            'prevorders': prevorders,
            'savedorders': savedorders})
```

'@Login_required' uses the login_required decorator of the Django authentication framework. This decorator checks whether the current user is authenticated. If the user is authenticated, it executes the decorated view and if the user is not authenticated, the decorator redirects the user to the login url with the originally requested url they were trying to access after they successfully log in.

```
{% if request.user.is_authenticated %}
```

```
{% block content %}
    <h1>Dashboard</h1>

    {% if user.is_manager == True %}
        <h1>manager app<a href = "manager/managerBase">here</a></h1>
    {% else %}
    {% endif %}

    {% if user.is_staff == True %}
        <h1>staff app <a href = "staff/staffBase">here</a></h1>
    {% else %}
    {% endif %}

    {% if user.is_driver == True %}
        <h1>driver app <a href = "driver/driverBase">here</a></h1>
    {% else %}
    {% endif %}

    {% if user.is_cook == True %}
        <h1>cook app <a href = "cook/cookBase">here</a></h1>
    {% else %}
    {% endif %}
    <p>Welcome to your Dashboard.</p>
    {% if user.is_customer == True %}
        <h2>Your Saved Orders:</h2>
```

```
class MyUser(AbstractUser):
    gender = models.BooleanField(default = True)
    password = models.CharField(max_length=50)
    username = models.CharField(max_length=50, unique = True)
    #boolean flags for access levels
    is_superuser = models.BooleanField(default = False)
    is_staff = models.BooleanField(default = False)
    is_cook = models.BooleanField(default = False)
    is_driver = models.BooleanField(default = False)
    is_manager = models.BooleanField(default = False)

    is_active = models.BooleanField(default = False)
    date_joined = models.DateField(auto_now=True)
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    email = models.EmailField()
```

Here is a sample of the Boolean flags and their use. This stops users from having any access to links to an app or page that they should not have. This stops users from being able to access these pages, or even see them if they know the url.

Creating a Popularity finding class

One of my key objectives was for different users to be able to sort different objects/records by their popularity or other options such as cost for the orders class for example. This data was not stored in the database, so I would have to obtain the data myself and then order it.

I started by writing the methods that would obtain the data. I did this first as I needed to know what type of data structure I would be sorting and how the data would be stored in it.

```
def totalQuantity(self):
                orderItems = self.getOrderItems()
                totalQuantityValue = 0
                for orderItem in orderItems:
                        totalQuantityValue = totalQuantityValue +
(orderItem.quantity)
                return totalQuantityValue
```

My original plan was to use a method like that which would return a value, in this case the popularity. I would use this to create a 2D array. Each array in this array would consists of two elements, the id and the numerical value that the objects/records were being sorted by. I then implement this and wrote a view function for obtaining the data:

```
#type is either total number ordered or number of times it appears in the db,
#e.g: 2 customers can order a product 10 times and 2 times
#type 1 will return a total of 12, type 2 will return a total of 2
def productPopularity(products, type):
        popularity = []

        for product in products:
                if type == 0:
                        productPopularity = [product.id,
product.totalQuantity()]
                else:
                        productPopularity = [product.id,
product.amountOftimesOrdered()]
                popularity.append(productPopularity)
        outerBubbleSortLoop(popularity, 0)#had lots of errors by having 'array
= ' instead of just calling the recursive function, none was returned
        products = sortedIDSToSortedArray(popularity, 0)


        return products
```

I now had to write the functions that would sort the 2D array and then instead of returning the sorted 2D array, use the ids to return the actual objects/records sorted.

I did not know where to start with this so I wrote some pseudocode to get started.

0 1 2 3 4

4 3 2 1 0

~~def bubbleSort(array, position~~
~~for i in range~~

swapped = False
bubbleSort(array)
~~if swa~~

i=0
def bubbleSort(array):
    i++
    if i != len(array):
        for j in range(len(array)-1):
            if array[j] > array[j+1]:
                temp = array[j+1]
                array[j+1] = array[j]
                array[j] = temp
    bubbleSort(array)

swapped = False
for i in range(len(array)):
    for j in range(len(array)-1):
        ~~if array[i] > array[j]:~~
            ~~temp = array[j]~~
            ~~array[j] = array[i]~~
            ~~array[i] = temp~~
        ~~if array[j+1] > array[j]:~~
            ~~temp = array[j]~~
            ~~array[j] = array[j+1]~~
        if array[j] > array[j+1]:
            temp = array[j+1]
            array[j+1] = array[j]
            array[j] = temp
            swapped = True

i=0
j=0

---

def innerBubbleSortLoop(array):
    j++
    if j != len(array)-1:
        if array[j] > array[j+1]:
            temp = array[j+1]
            array[j+1] = array[j]
            array[j] = temp
        innerBubbleSortLoop(array)
    else:
        return(array)

Recursive Bubble Sort

~~def outerBubbleSortLoop(array):~~
~~i++~~
~~if i != len(array):~~
~~innerBubbleSortLoop(array)~~
~~outerBubbleSortLoop(array)~~

def outerBubbleSortLoop(array):
    i++
    if i != len(array):
        innerBubbleSortLoop(array)
        outerBubbleSortLoop(array)
    else:
        return array

Category — Products — OrdersItems — Orders — Customers

Sort Categories based off popularity:

> popularity total sale of total sales count of Products
> total of amount of times it is ordered.

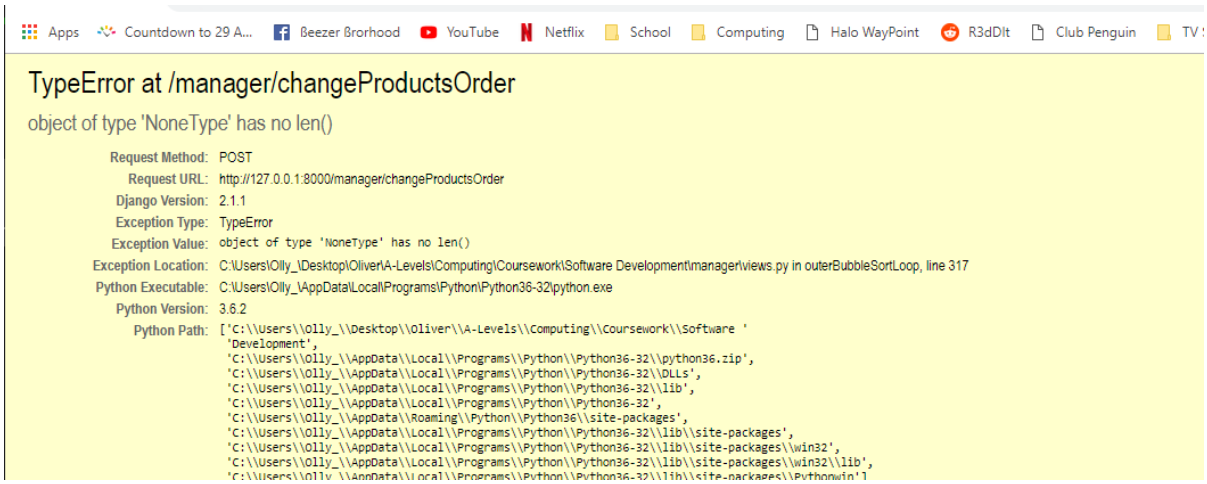
```
class Category(self):
    def getProducts(self):
        products = Product.objects.filter(category_id = self.id)
        return products

    def getTotalQuantityOfCusProducts(self):
        products = self.getProducts(self)
        total = 0
        for product in products:
            total += product.getTotalQuantity()
        return total

    def getTotalOfCusProducts(self):

    def getTotal(self, type):
        products = self.getProducts()
        total = 0
        for product in products:
            if type = 0:
                total += product.getTotalQuantity()
            else:
                total += product.getTotalOrdered()
        return total
```

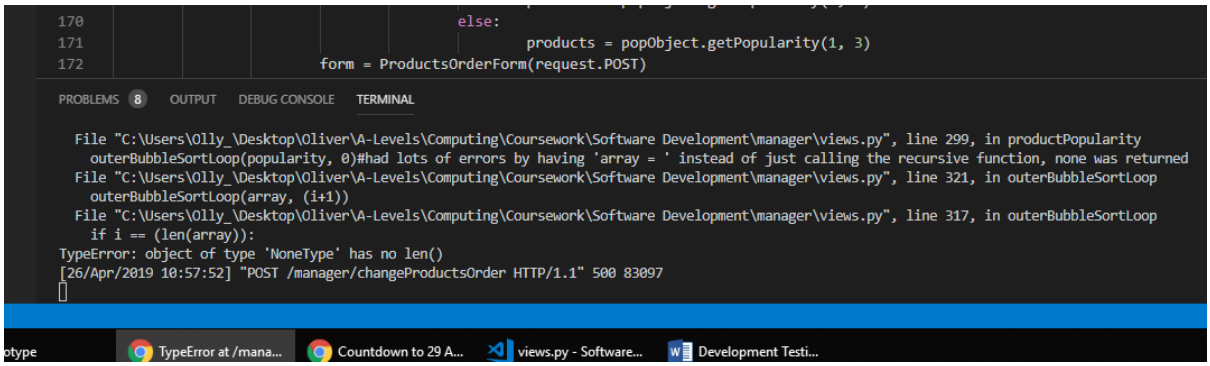
Views file:

```
def findPopCat():
    categories = getCategories()
    popularity = []
    for category in categories:
        popularity.append(category.id,
                          category.getTotal(0))
    outerBubbleSortLoop(popularity)
```

In this pseudocode I wrote the sorting functions. I chose to use a recursive bubble sort approach. On the second page I wrote how the data for the categories would be obtained and how it would all come together in the views file.

However when I implemented this I ran into some problems. Firstly the sorting functions did not work. I received this error:





Here is the code of those functions at the time:

```
#RECURSIVE
#recursive bubble sort
def outerBubbleSortLoop(array, i):

        if i == (len(array)):
                return(array)
        else:
                array = innerBubbleSortLoop(array, 0)
                array = outerBubbleSortLoop(array, (i+1))

#RECURSIVE
def innerBubbleSortLoop(array, j):
        if j == (len(array)-1):
                return(array)
        else:
                if array[j][1] > array[j+1][1]:
```

```
                        temp = array[j+1]
                        array[j+1] = array[j]
                        array[j] = temp
                array = innerBubbleSortLoop(array, (j+1))
```

The problem was that when I tried to return the array, e.g.: ‘array = innerBubbleSortLoop(array , 0) none was returned. I only found out that after lots and lots of debugging. This problemwas caused because I didn’t fully understand how recursive functions work with regards to memory of variables. After researching further I found out that there was no need to have

```
                array = innerBubbleSortLoop(array, 0)
                array = outerBubbleSortLoop(array, (i+1))
```

Instead I just needed to call the recursive function, and then it would work. Below is the code that worked.

```
#RECURSIVE
#recursive bubble sort
def outerBubbleSortLoop(array, i):

        if i == (len(array)):
                return(array)
        else:
                innerBubbleSortLoop(array, 0)
                outerBubbleSortLoop(array, (i+1))

#RECURSIVE
def innerBubbleSortLoop(array, j):
        if j == (len(array)-1):
                return(array)
        else:
                if array[j][1] > array[j+1][1]:
                        temp = array[j+1]
                        array[j+1] = array[j]
                        array[j] = temp
                innerBubbleSortLoop(array, (j+1))

#this function takes an array of sorted ids, and uses that to output a sorted
list of objects
def sortedIDSToSortedArray(array, type):
        sortedArray = []
        for element in array:
                if type == 0:
                        elementObject = Product.objects.get(id = element[0])
                elif type == 1:
                        elementObject = Category.objects.get(id = element[0])
                sortedArray.append(elementObject)
        return sortedArray
```

On the next screen is a screenshot showing the products being successfully sorted

Pedro's Pizzeria

Hello manger, Logout

My dashboard

Manager Menu

Product List

AttributeChosen: Popularity of total quantity orderded

OrderOfProducts: Ascending

update

Add a New Product

Product ID	Name	Image	Description	Price	Available	Created	Updated	Category	total amount of times ordered	total Quantity	Delete
13	Fanta		Fanta can.	2.00	True Change Availablity	March 29, 2019, 11:50 a.m.	March 29, 2019, 11:50 a.m.	Drinks	7	8	Delete
3	pepperoni		01101100 01101101 01100001 01101111 00100000 01101101 01100001 01100100 01100101 00100000 01111001 01101111 01110101 00100000 01101100 01101111 01101111 01101011 00100000 01101110 01101111 01101111 01100010 01111010	6.90	True Change Availablity	Nov. 20, 2018, 2:38 p.m.	Feb. 22, 2019, 1:58 p.m.	Pizza	45	117	Delete
4	Margerita		http://127.0.0.1:8000	2.61	True Change Availablity	Nov. 20, 2018, 2:44 p.m.	Feb. 22, 2019, 1:58 p.m.	Pizza	68	123	Delete
12	Ice Cream		This a an ice cream	10.00	True Change Availablity	March 1, 2019, 2:53 p.m.	March 1, 2019, 2:53 p.m.	Treats	53	196	Delete

The next problem that I ran into was that this approach was pretty messy and not very easy to make use elsewhere in my code especially in other apps. This was because I would have to import and call many different functions from the views file of a specific app which would be impractical.

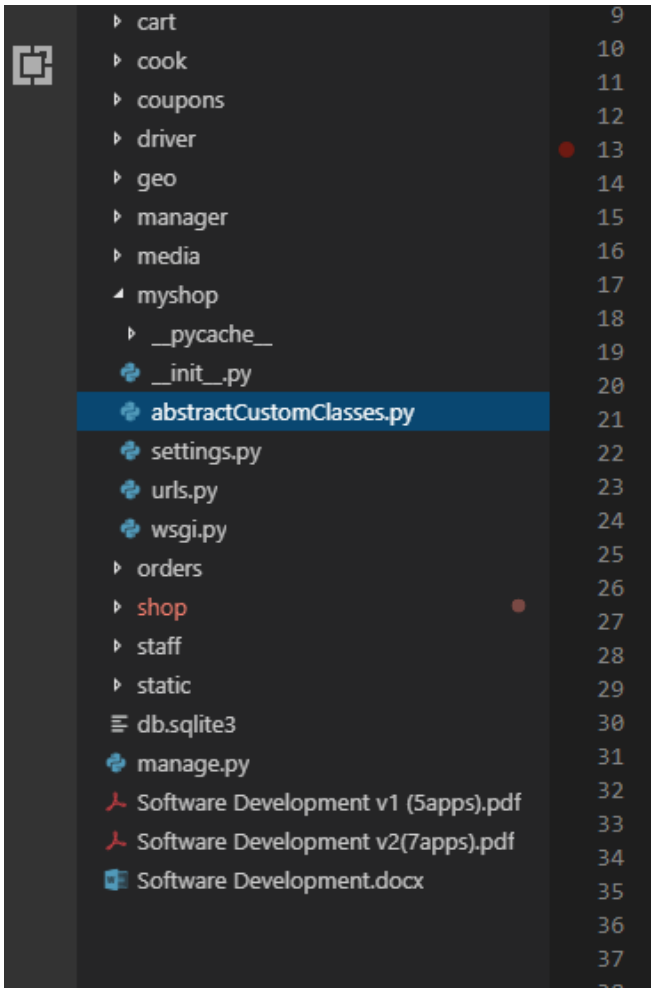
I decided to change my approach from a functional one, to using classes in a dedicated file that could then be used anywhere else in my project with ease.

Again, I started this by writing some pseudocode which is on the next page.

In the folder that holds the settings of my project, I created a new python file called abstractCustomClasses.py. this file contains two classes, one that gets the data and another that sorts it. The class that sorts the data is a child class of the class that sorts the data. I did this so that I could make use of inheritance and use the methods in the sorting class in the class that obtains the data.

The reason I did this was so that I would only need to import and interact one class, meaning that if I did need to make changes it would be easier.

The reason why I split the two classes up was so that the sorting class could be used elsewhere if needed, making my code more reusable which is a large part of why we use OOP.



```
def sortit(array, order, database)
    self.outerBubbleSortLoop(array, 0)
    sortedIDsToSortedObjects(array, database)
    if order == 1:
        array.reverse
    return array
```

class abstractObjectSort(array, order, database

```
    self.array = array
    self.order = order
    self.database = database
```

```
def outerBubbleSortLoop(array, i):
    if i != len(array):
        innerBubbleSortLoop(array, j=0)
        i++
        outerBubbleSortLoop(array, i)
    else:
        return array

def innerBubbleSortLoop(array, j):
    if j != len(array)-1:
        if array[j][0] > array[j+1][0]:
            temp = array[j+1][0]
            array[j+1][0] = array[j][0]
            array[j][0] = temp
        j++
        innerBubbleSortLoop(array, j)
    else:
        return(array)
```

```
def sortedIDsToSortedObjects(array, database):
    if database == "products":
        from shop.models import Product
        arrayOfObjects = []
        for element in array:
            elementObject = Product.objects.get(id = element[1])
            arrayOfObjects.append(elementObject)

    return(arrayOfObjects)
```

```
sortArray = abstractObjectSort(array)
sortArray.sortit()
```

```
def sortit(array, order, database):
    self.outerBubbleSortLoop(array, 0)
    array = sortedIDsToSortedObjects(array, database)
    if order == 1:
        array.reverse
    return(array)
```

```
sortArray = abstractObjectSort()
array = sortArray.sortit(array, order, database).
```

arr = [shop.models]

arr[0]

```python
from shop.models import Product, Category
from coupons.models import Coupon
from account.models import MyUser

class abstractObjectSort(object):
    def sortIt(self, array, order, datatype):
        self.outerBubbleSortLoop(array, 0)
        array = self.sortedIDSToSortedArray(array, datatype)
        if order == 1:
            array.reverse()
        return array


#recursive bubble sort
#recursive outer loop
    def outerBubbleSortLoop(self, array, i):

        if i == (len(array)):
                return(array)
        else:
                self.innerBubbleSortLoop(array, 0)
                self.outerBubbleSortLoop(array, (i+1))
#recursive inner loop
    def innerBubbleSortLoop(self, array, j):
        if j == (len(array)-1):
                return(array)
        else:
                if array[j][1] > array[j+1][1]:
                        temp = array[j+1]
                        array[j+1] = array[j]
                        array[j] = temp
                self.innerBubbleSortLoop(array, (j+1))

    def sortedIDSToSortedArray(self, array, datatype):
        sortedArray = []
        for element in array:
                if datatype == 0:
                        elementObject = Product.objects.get(id = element[0])
                elif datatype == 1:
                        elementObject = Category.objects.get(id = element[0])
                elif datatype == 2:
                        elementObject = Coupon.objects.get(id = element[0])
                elif datatype == 3:
                        elementObject = Product.objects.get(id = element[0])

                elif datatype == 4:
                        elementObject = MyUser.objects.get(id = element[0])
                elif datatype == 5:
                        elementObject = MyUser.objects.get(id = element[0])
```

```
                elif datatype == 6:
                        elementObject = MyUser.objects.get(id = element[0])
                elif datatype == 7:
                        elementObject = MyUser.objects.get(id = element[0])

                sortedArray.append(elementObject)
        return sortedArray

class abstractPopularity(abstractObjectSort):
    def getPopularity(self, order, datatype):
        popularity = []
        if datatype == 0:
            products = Product.objects.filter()
            for product in products:
                popularity.append([product.id, product.totalQuantity()])
        elif datatype == 1:
            categories = Category.objects.filter()
            for category in categories:
                popularity.append([category.id, category.getTotal(0)])
        elif datatype == 2:
            coupons = Coupon.objects.filter()
            for coupon in coupons:
                popularity.append([coupon.id, coupon.findTotalUse()])
        elif datatype == 3:
            products = Product.objects.filter()
            for product in products:
                popularity.append([product.id,
product.amountOftimesOrdered()])

        elif datatype == 4:
            customers = MyUser.objects.filter()
            for customer in customers:
                popularity.append([customer.id, customer.numOfOrdersOfCus()])
        elif datatype == 5:
            customers = MyUser.objects.filter()
            for customer in customers:
                popularity.append([customer.id,
customer.numOfDeliveriesOfCus()])
        elif datatype == 6:
            customers = MyUser.objects.filter()
            for customer in customers:
                popularity.append([customer.id,
customer.numOfCollectionsOfCus()])
        elif datatype == 7:
            customers = MyUser.objects.filter()
            for customer in customers:
                popularity.append([customer.id, customer.numOfInStoreOfCus()])
```

```
        elif datatype == 8:
            customers = MyUser.objects.filter()
            for customer in customers:
                popularity.append([customer.id, customer.totalSpent()])
        elif datatype == 9:
            customers = MyUser.objects.filter()
            for customer in customers:
                popularity.append([customer.id, customer.averageSpent()])

        return self.sortIt(popularity, order, datatype)
```