Final Project Requirements

Contents

Final Project

Overview

For the final project, you will be tasked with creating a recipe application that will retrieve and display meal recipes. For this application, you will be using the APIs provided by the website, www.themealdb.com/, which is a free service.

Requirements

- Create a console application.
- Display an Application Title and the Menu Options that the user can refer to when using the application.
- Allow the user to display a list of recipe categories (Beef, Chicken, Vegan, Side, Desert, etc.).
- Allow the user to display a list of meals based on a selected category.
- Allow the user to display meal information (instructions and ingredients) based on a selected meal.
- Allow the user to display a random meal using the URL for retrieving random meals from themealdb.com.
- Allow the user to display a list of areas (Chinese, French, Thai, etc.).
- Allow the user to display a list of meals based on a selected area.

Submitting

You will be required to submit your code in the form of attaching all your python (.py) files via Canvas. I will be running your application from these code files to validate program functionality.

Grading

The final project is worth 100 points (or 30% of your final grade). See the rubric in the Final project section of Canvas for the full point break down as well as required functionality details. There are no regrades on the final project so make sure it is ready before marking it ready for grading (see additional hints below).

Coding Specifications

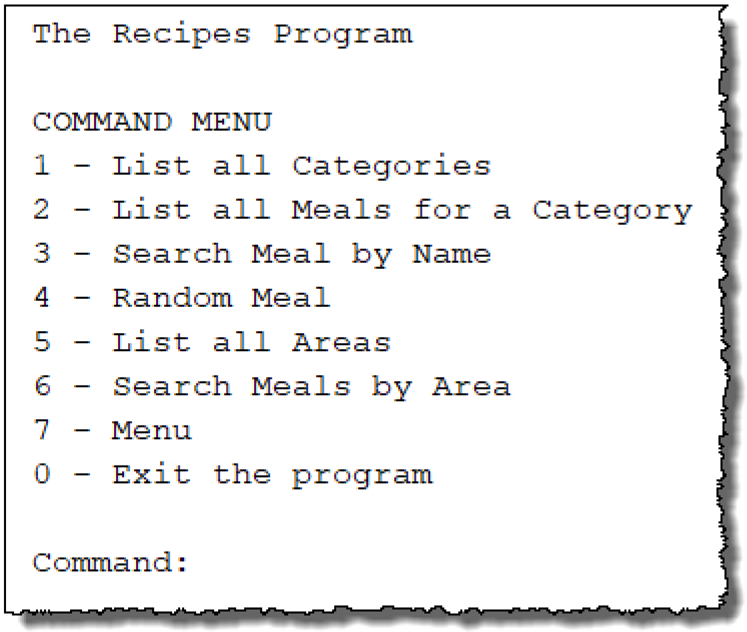
You will be required to follow OOP principals, separation of the data, business, and presentation layers. You should not be directly accessing the APIs from the presentation (recipes.py) or business (objects.) layers or have any “print” statements in the business (objects.py) or data (requests.py) layers. You should practice “clean” coding practices (comments and consistent use of white spacing, etc.). Additionally, you should be looking to use the most efficient code (reusing classes and functions) where possible.

Screen Specifications

The following are screen outputs of the functionality you will be required to provide. Any UI deviation from the following screen images must be pre-approved by the instructor.

Menu

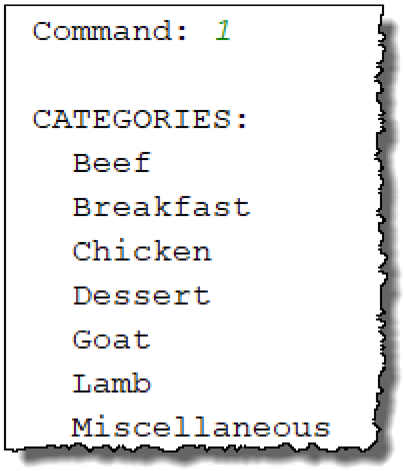
You will need to provide a menu that will allow the user to display the various options of the application.



```
The Recipes Program

COMMAND MENU
1 - List all Categories
2 - List all Meals for a Category
3 - Search Meal by Name
4 - Random Meal
5 - List all Areas
6 - Search Meals by Area
7 - Menu
0 - Exit the program

Command:
```

Category Listing

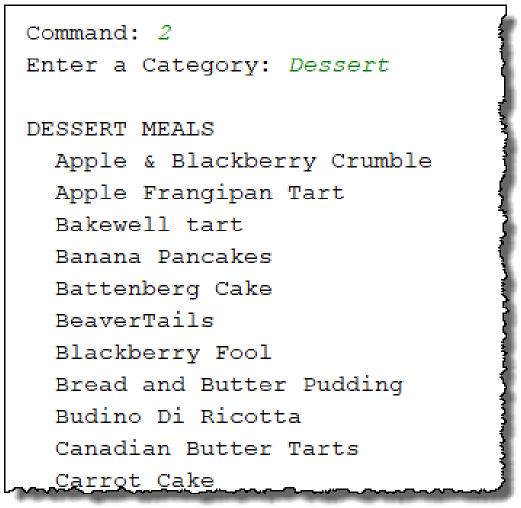
You will need to list all the available categories. The categories themselves should be indented to provide a better visual for the user.



```
Command:  1

CATEGORIES:
   Beef
   Breakfast
   Chicken
   Dessert
   Goat
   Lamb
   Miscellaneous
```

Meals for a Category

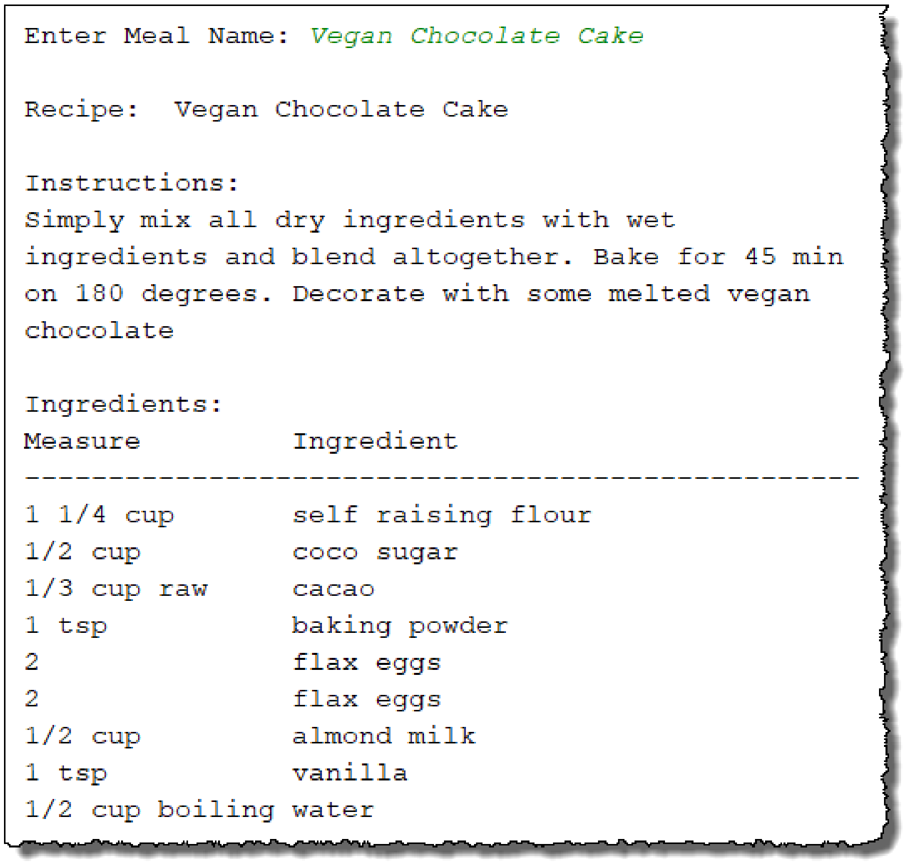
You will need to list all the meals for a category. The meals for the category should be indented to provide a better visual for the user.



```
Command: 2
Enter a Category: Dessert

DESSERT MEALS
  Apple & Blackberry Crumble
  Apple Frangipan Tart
  Bakewell tart
  Banana Pancakes
  Battenberg Cake
  BeaverTails
  Blackberry Fool
  Bread and Butter Pudding
  Budino Di Ricotta
  Canadian Butter Tarts
  Carrot Cake
```

Meal by Name

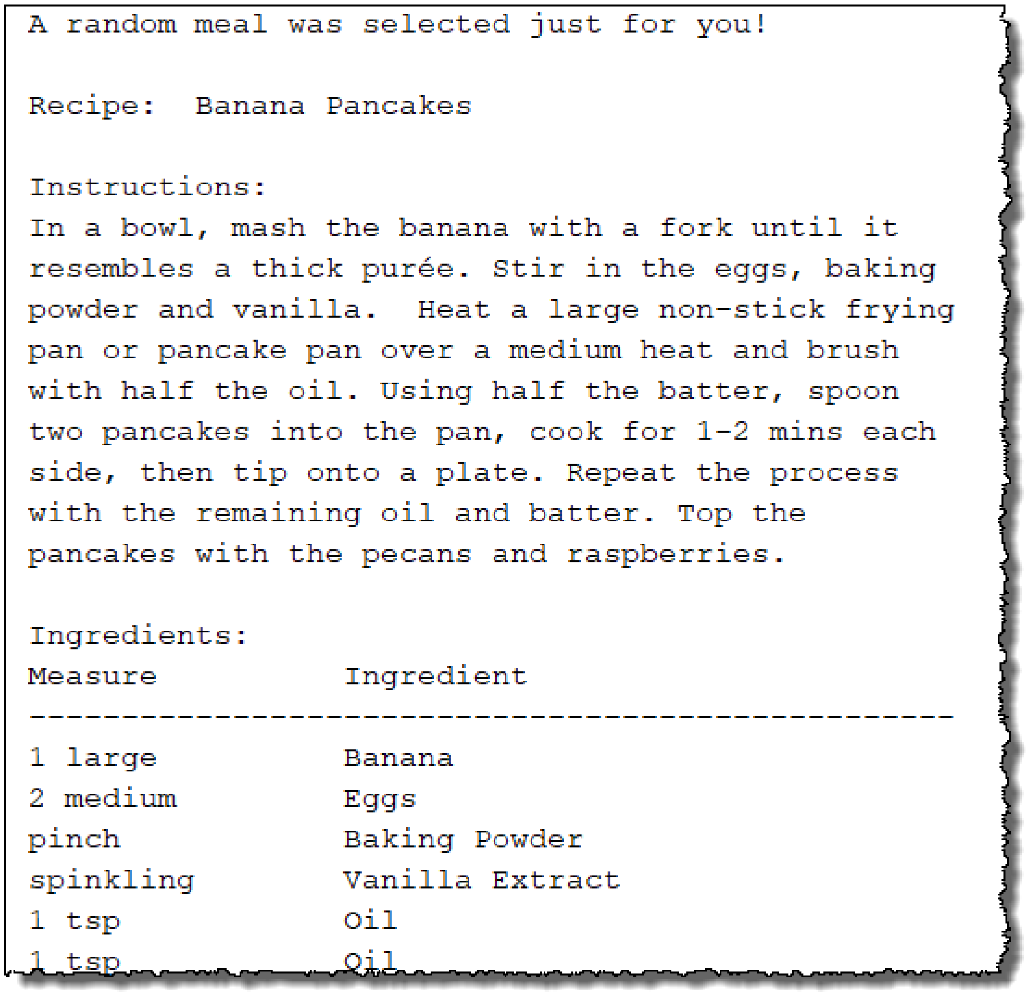
The instructions should word wrap at 80 characters.



```
Enter Meal Name: Vegan Chocolate Cake

Recipe:  Vegan Chocolate Cake

Instructions:
Simply mix all dry ingredients with wet
ingredients and blend altogether. Bake for 45 min
on 180 degrees. Decorate with some melted vegan
chocolate

Ingredients:
Measure         Ingredient
--------------------------------------------------
1 1/4 cup       self raising flour
1/2 cup         coco sugar
1/3 cup raw     cacao
1 tsp           baking powder
2               flax eggs
2               flax eggs
1/2 cup         almond milk
1 tsp           vanilla
1/2 cup boiling water
```

Random Meal

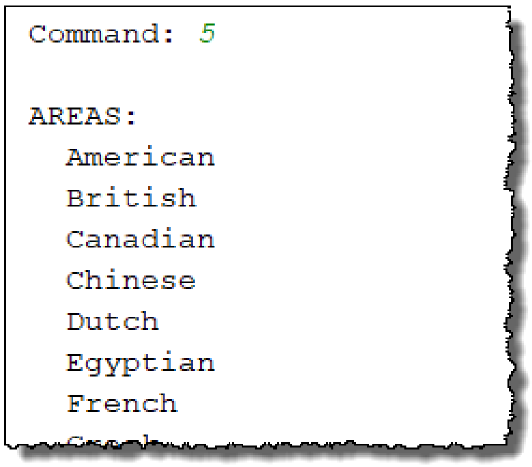
You will need to display a random meal. Include a display line to let the user know this is a random meal that is being displayed. *(There is an API you can use to retrieve a random meal).*



```
A random meal was selected just for you!

Recipe:  Banana Pancakes

Instructions:
In a bowl, mash the banana with a fork until it
resembles a thick purée. Stir in the eggs, baking
powder and vanilla.  Heat a large non-stick frying
pan or pancake pan over a medium heat and brush
with half the oil. Using half the batter, spoon
two pancakes into the pan, cook for 1-2 mins each
side, then tip onto a plate. Repeat the process
with the remaining oil and batter. Top the
pancakes with the pecans and raspberries.

Ingredients:
Measure          Ingredient
-------------------------------------------------
1 large          Banana
2 medium         Eggs
pinch            Baking Powder
spinkling        Vanilla Extract
1 tsp            Oil
1 tsp            Oil
```

Area Listing

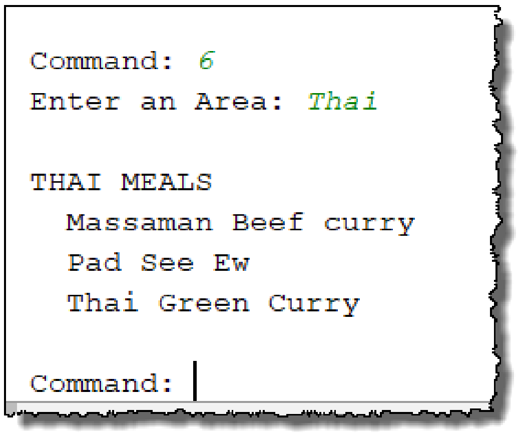
You will need to list all the available categories. The areas will need to be indented for visual clarity.



```
Command:  5

AREAS:
  American
  British
  Canadian
  Chinese
  Dutch
  Egyptian
  French
```

Meals for an Area

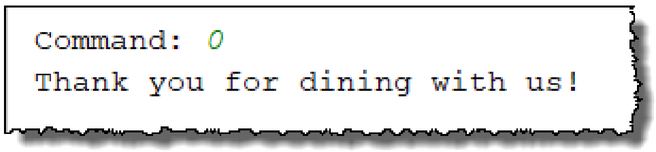
You will need to list all the available categories. The meals for the area will need to be indented for visual clarity.



```

Command:  6
Enter an Area:  Thai

THAI MEALS
  Massaman Beef curry
  Pad See Ew
  Thai Green Curry

Command:
```

End Application Message

You will need to display a message when the app ends.



```
Command:  0
Thank you for dining with us!
```

Organizing

This section will provide you with a high-level overview of how to organize the final project. If you need more help than what is provided here, see the Getting Started document

Review Requirements

Take a minute to review and understand all the requirements, the rubric is a good place to start. If you are not clear on any of the requirements, make sure you ask the instructor for clarification.

Also, make sure you review the *Hints* section at the end of this document, it might provide some valuable tips for organizing your thoughts on how to complete the final project.

Assets – The Data

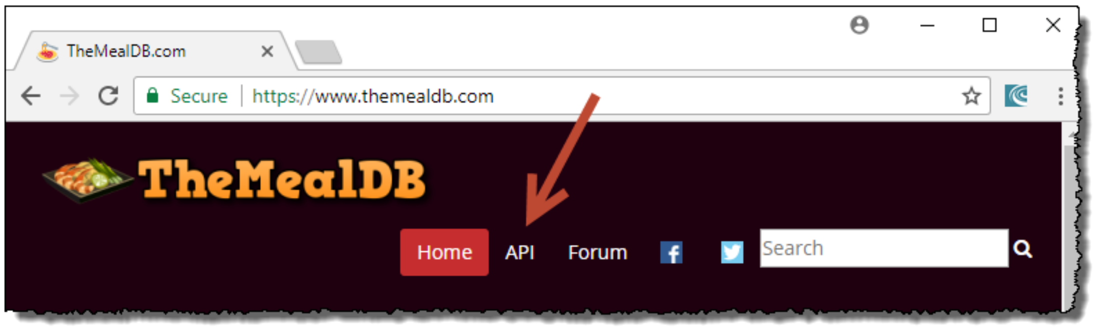
TheMealDB.com is an open-source database loaded with recipes that have been crowd sourced from around the world. The site provides free JSON APIs that you will be accessing to retrieve the various information that you will be displaying in your app.

Note: The site provides a developer's key ("1") that will return a small sampling of recipes which will be adequate for the final project. To gain full access to all the recipes, you will need to register with the site and replace the developer's key with the one assigned to you when you register. Additionally, while it is outside the scope of the final project, with the registered key, you will also be able to submit recipes to the site.

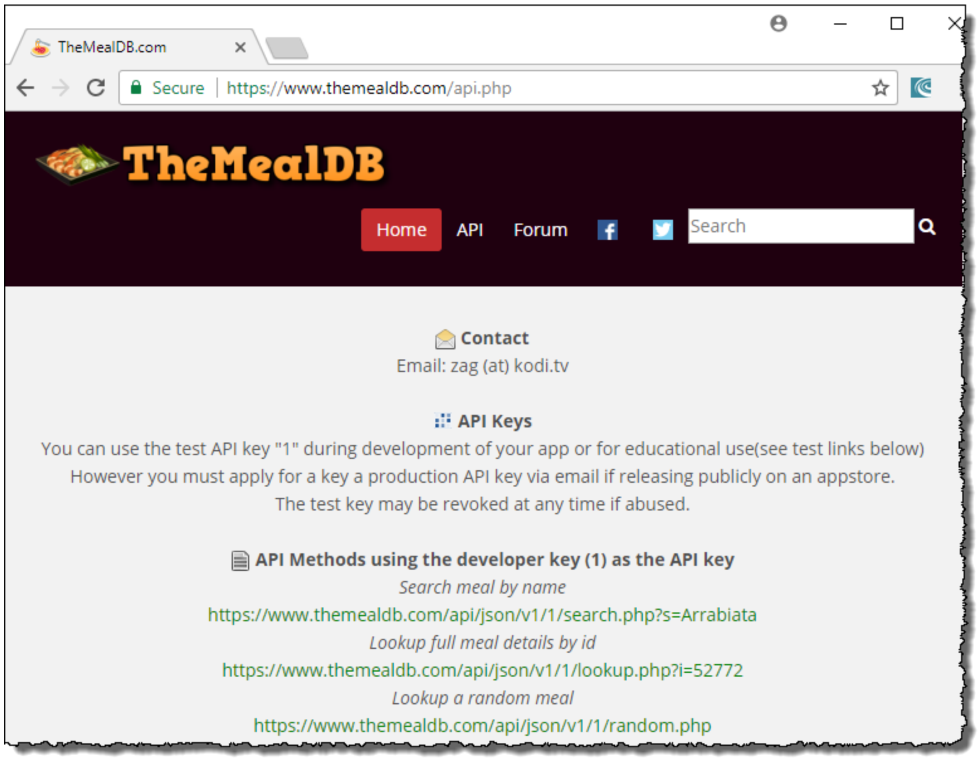
APIs

Review the APIs.

Open a web browser (such as Chrome or Firefox) and go to www.TheMealDB.com and select the API option in the page header.



From this page you will see the full list of APIs. You should familiarize yourself with the list so you know all the APIs available to you.



Contact
Email: zag (at) kodi.tv

API Keys
You can use the test API key "1" during development of your app or for educational use(see test links below)
However you must apply for a key a production API key via email if releasing publicly on an appstore.
The test key may be revoked at any time if abused.

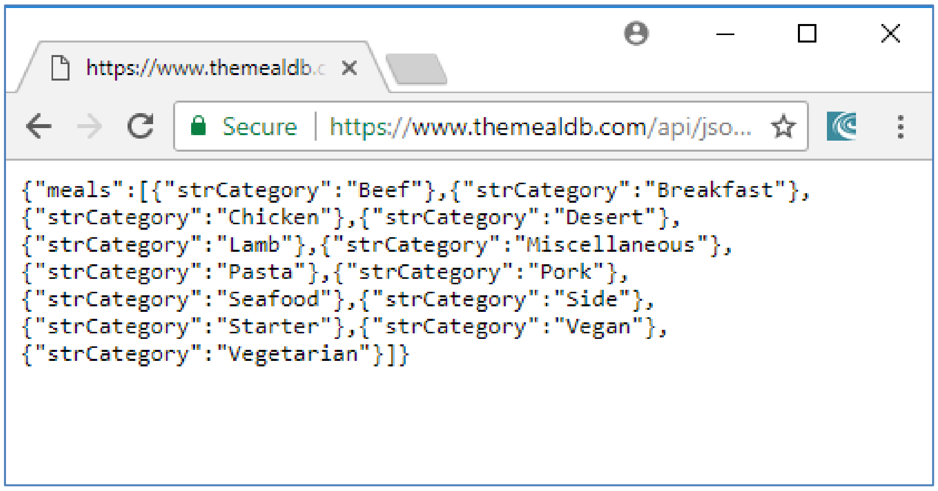
API Methods using the developer key (1) as the API key
Search meal by name
https://www.themealdb.com/api/json/v1/1/search.php?s=Arrabiata
Lookup full meal details by id
https://www.themealdb.com/api/json/v1/1/lookup.php?i=52772
Lookup a random meal
https://www.themealdb.com/api/json/v1/1/random.php

JSON Files

After determining which APIs, you will use for each requirement, review the JSON file to see the structure of the object that is being returned.

The first requirement for the final project is to display a list of categories. From the list of APIs (as described in the previous section), find the API to “List all Categories”. Note: There is a “List all meal categories” and a “List all Categories”. If you take a moment to view the “List all meal categories”, you will see it includes a lot of additional information that will not be required in the application whereas the “List all Categories” just includes a list of the category names.

If you copy the URL (https://www.themealdb.com/api/json/v1/1/list.php?c=list) and paste into a web browser. Note: the “1” after the “…/v1/”, this is the developer’s key previously mentioned, you will see what is returned and can determine the structure of the JSON file.



```
{"meals":[{"strCategory":"Beef"},{"strCategory":"Breakfast"},
{"strCategory":"Chicken"},{"strCategory":"Desert"},
{"strCategory":"Lamb"},{"strCategory":"Miscellaneous"},
{"strCategory":"Pasta"},{"strCategory":"Pork"},
{"strCategory":"Seafood"},{"strCategory":"Side"},
{"strCategory":"Starter"},{"strCategory":"Vegan"},
{"strCategory":"Vegetarian"}]}
```

Notice that if you run the “Filter by Category” https://www.themealdb.com/api/json/v1/1/filter.php?c=Seafood API, you can also replace the “Seafood” option with any category from the returned category list above (i.e. https://www.themealdb.com/api/json/v1/1/filter.php?c=Starter

Creating the Application

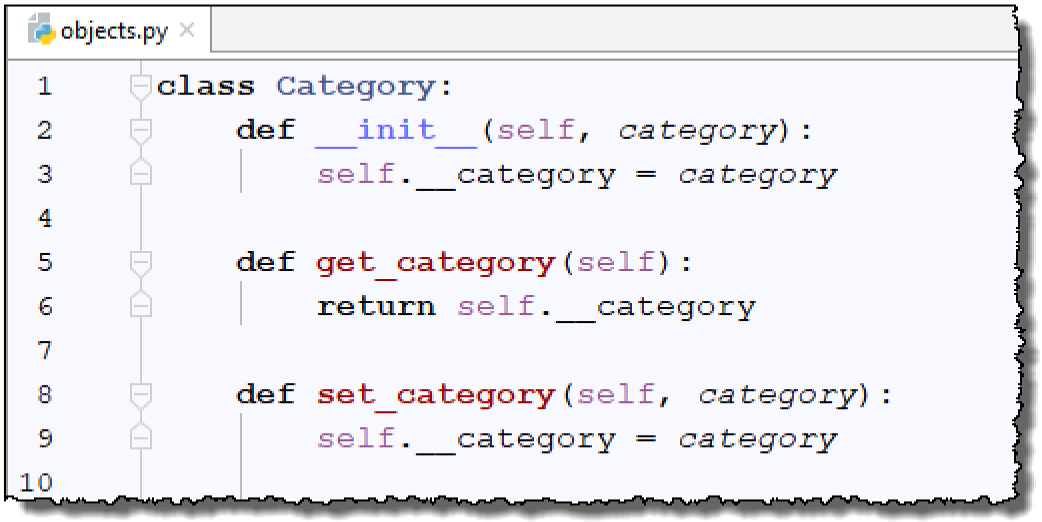
You will need to create a project for your application, I recommend **myRecipeApplication**.

Creating the Objects File

Each API will return data in the form of a JSON file, so you'll need to create data objects to make it easier to work with that data. Since these objects will be part of the business logic, you will want to create a single file to store all the data objects. I recommend naming this file **objects.py**.

Creating a Class

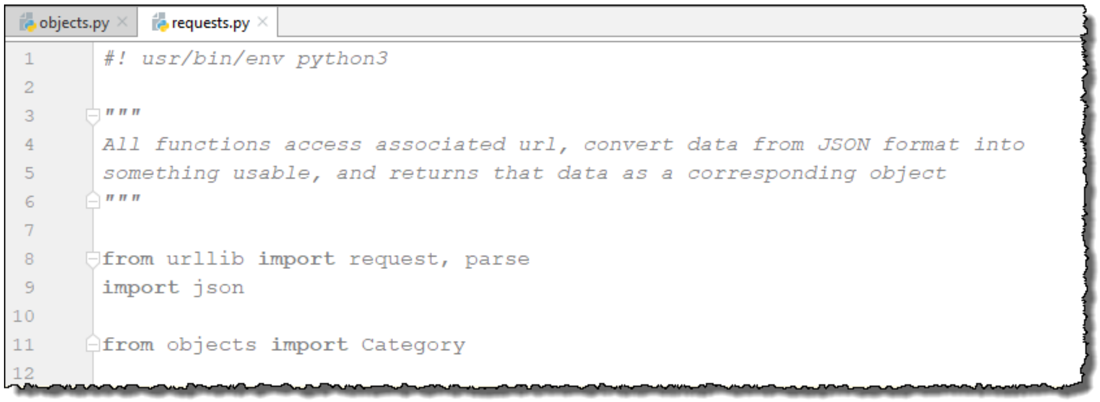
Continuing from the previous JSON section, the first class you will want to work on is the Category class. Since the category data only has one element "*strCategory" (see the JSON image above)*, the category class will only have one property (category). using the OOP methodology we covered during the semester, all objects should be encapsulated. Note: Since the app will only be displaying data, the setters are not really required.


objects.py

```python
class Category:
    def __init__(self, category):
        self.__category = category

    def get_category(self):
        return self.__category

    def set_category(self, category):
        self.__category = category

```

Creating the Request File

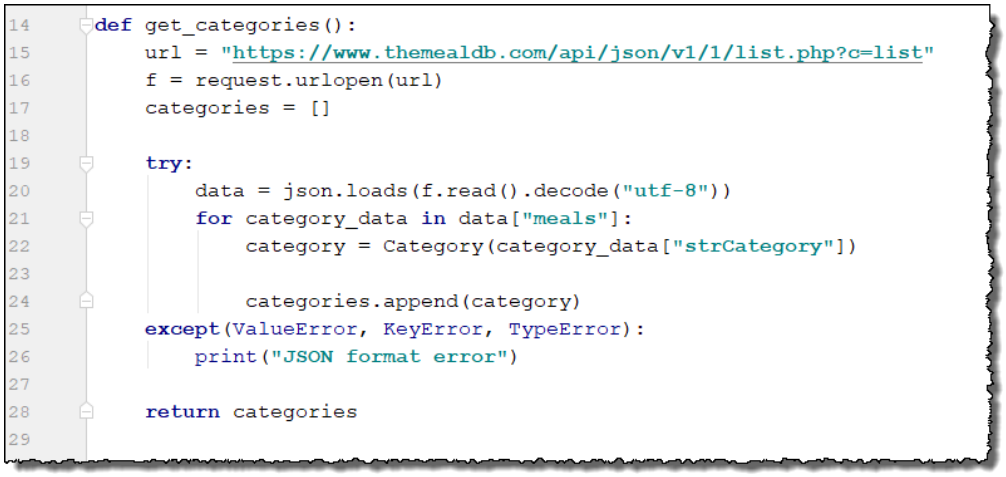
The data layer should contain all the functions that retrieves the data in the form of API request calls and should be contained in a single file, I recommend naming the file **requests.py**. For the API requests, you will need to include a few imports that you may not be familiar with, the urllib and json. The urllib import will allow you to make the API call and the json import will allow you to parse the returned “JSON data” into objects which you can use. Since you will be parsing the data into objects defined in the object file, you will also need to import the Category class from the objects file.



```
#! usr/bin/env python3

"""
All functions access associated url, convert data from JSON format into
something usable, and returns that data as a corresponding object
"""

from urllib import request, parse
import json

from objects import Category

```

Creating the API Calls

A few things to note here before you code this out. You imported the request and parse from urllib. The open method is from the request import which allows you to make the call and will do the work to retrieve the data from the API. You should note the try except block; this is for when you get bad data back from the API, you will be able to handle it gracefully and not crash the application.



```
14  def get_categories():
15      url = "https://www.themealdb.com/api/json/v1/1/list.php?c=list"
16      f = request.urlopen(url)
17      categories = []
18
19      try:
20          data = json.loads(f.read().decode("utf-8"))
21          for category_data in data["meals"]:
22              category = Category(category_data["strCategory"])
23
24              categories.append(category)
25      except(ValueError, KeyError, TypeError):
26          print("JSON format error")
27
28      return categories
29
```

Note the print statement (line 26), this is in the data layer and you will need to handle errors in a different manner as print statements in the data layer will break the OOP separation of layer rule.

Testing the API Calls

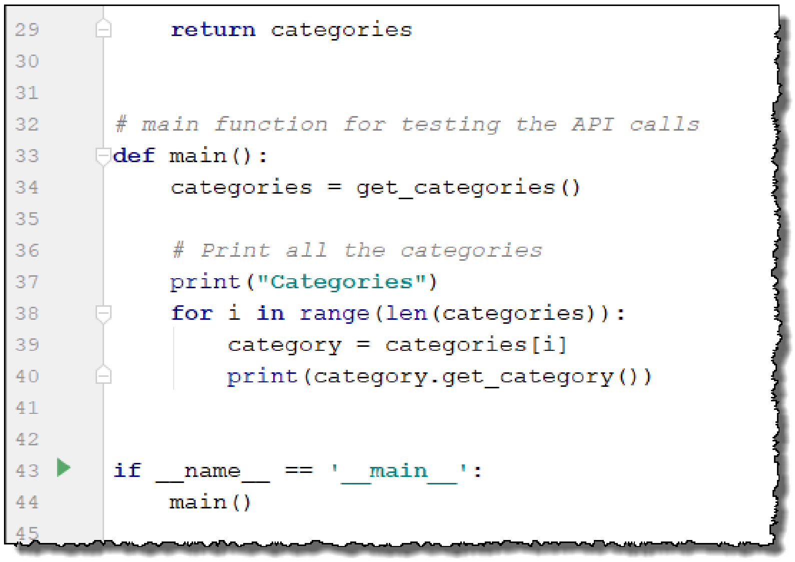
Mac User Notice: It appears later versions of Python installs do not come with certificates pre-installed. This will cause an error when running the urllib

Urllib.error.URLError: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verified failed: unable to get local issuer certificate.

To install the certificates, in terminal, locate the ***/Applications/Python/3.x*** folder (replace the x with the version of Python you have installed).

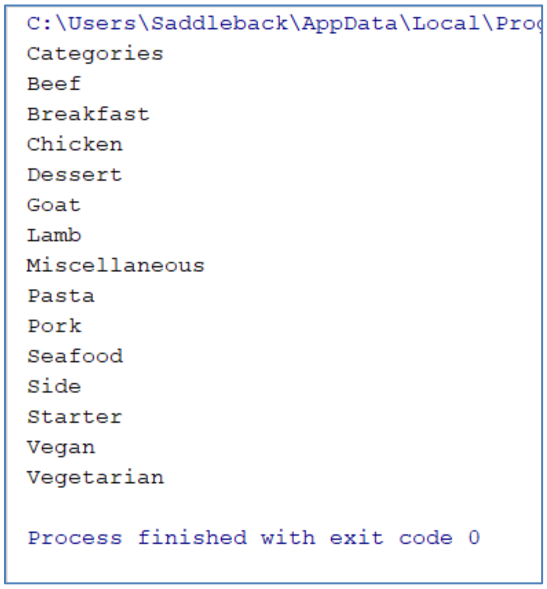
Locate and double click on the **Install Certificates.command**

You should create a simple console UI to validate your request call works as intended.



```
    return categories


# main function for testing the API calls
def main():
    categories = get_categories()

    # Print all the categories
    print("Categories")
    for i in range(len(categories)):
        category = categories[i]
        print(category.get_category())


if __name__ == '__main__':
    main()
```

And your output should look something like…

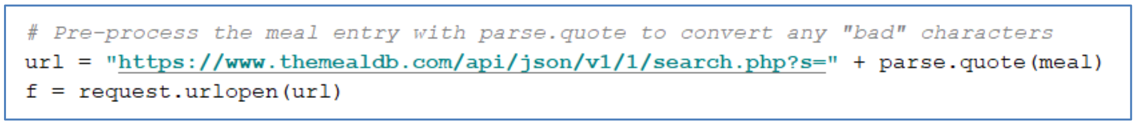


```
C:\Users\Saddleback\AppData\Local\Prog
Categories
Beef
Breakfast
Chicken
Dessert
Goat
Lamb
Miscellaneous
Pasta
Pork
Seafood
Side
Starter
Vegan
Vegetarian

Process finished with exit code 0
```

If you get an error, test the URL by coping and pasting the url (everything inside the “”) into a browser to make sure your URL is correct. If you get a JSON file back that looks correct, start checking the rest of the code for error.

Note on Meal by Name API

For searching a meal by name, since meals can have characters (i.e. apostrophes, etc.) that will cause issues, you will need to use the parse.quote() function to format the entry to convert any bad characters. Your code should look something like:



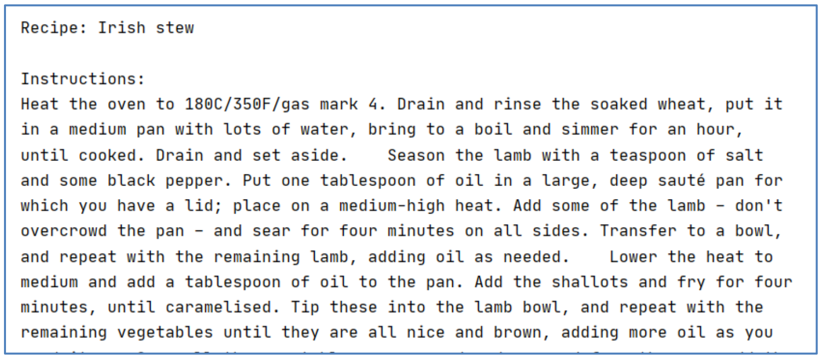
```python
# Pre-process the meal entry with parse.quote to convert any "bad" characters
url = "https://www.themealdb.com/api/json/v1/1/search.php?s=" + parse.quote(meal)
f = request.urlopen(url)
```

Creating the UI

The final python file you will need to create will be for the presentation layer. This will be the main file for the application and contain all the functionality to display the menu and all screens. I recommend naming this file **recipes.py**. From here you will need to create the menu, the main function to control the flow of the program and the remaining logic to complete the final project.

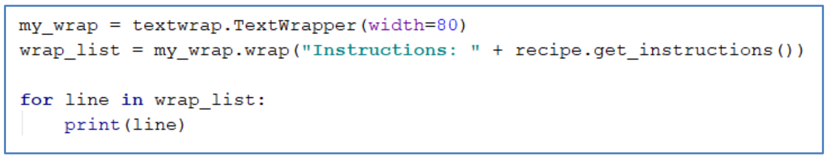
Formatting Meal by Name / Random Meal Notes:

The recipe instructions text should wrap at 80 characters.



```
Recipe: Irish stew

Instructions:
Heat the oven to 180C/350F/gas mark 4. Drain and rinse the soaked wheat, put it
in a medium pan with lots of water, bring to a boil and simmer for an hour,
until cooked. Drain and set aside.    Season the lamb with a teaspoon of salt
and some black pepper. Put one tablespoon of oil in a large, deep sauté pan for
which you have a lid; place on a medium-high heat. Add some of the lamb - don't
overcrowd the pan - and sear for four minutes on all sides. Transfer to a bowl,
and repeat with the remaining lamb, adding oil as needed.    Lower the heat to
medium and add a tablespoon of oil to the pan. Add the shallots and fry for four
minutes, until caramelised. Tip these into the lamb bowl, and repeat with the
remaining vegetables until they are all nice and brown, adding more oil as you
```

- Hint: For the text wrapping of the instructions, I imported the textwrap module and used the wrap function which breaks a string into a list of strings by the specified length. My code looked like:

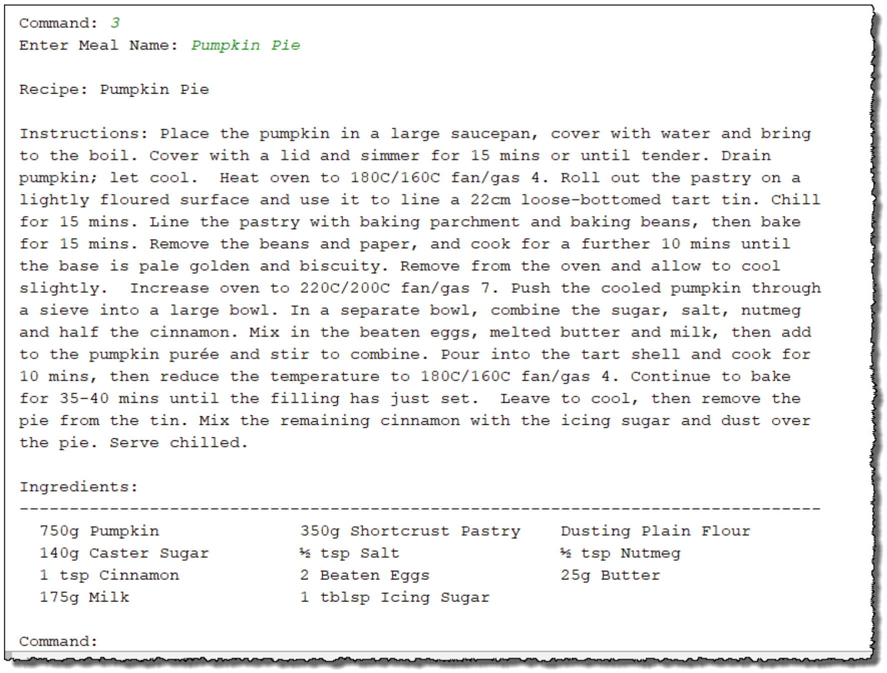


```python
my_wrap = textwrap.TextWrapper(width=80)
wrap_list = my_wrap.wrap("Instructions: " + recipe.get_instructions())

for line in wrap_list:
    print(line)
```

The ingredients section:

- You should only display as many lines as there is data.
- You should strip off any additional spaces that might have been entered before or after the ingredient/measurement.
- You should handle when the value entered is empty (“”), blank space(s), or None.

Meal by Name – Ingredient Formatting (Extra Credit)



```
Command: 3
Enter Meal Name: Pumpkin Pie

Recipe: Pumpkin Pie

Instructions: Place the pumpkin in a large saucepan, cover with water and bring
to the boil. Cover with a lid and simmer for 15 mins or until tender. Drain
pumpkin; let cool.  Heat oven to 180C/160C fan/gas 4. Roll out the pastry on a
lightly floured surface and use it to line a 22cm loose-bottomed tart tin. Chill
for 15 mins. Line the pastry with baking parchment and baking beans, then bake
for 15 mins. Remove the beans and paper, and cook for a further 10 mins until
the base is pale golden and biscuity. Remove from the oven and allow to cool
slightly.  Increase oven to 220C/200C fan/gas 7. Push the cooled pumpkin through
a sieve into a large bowl. In a separate bowl, combine the sugar, salt, nutmeg
and half the cinnamon. Mix in the beaten eggs, melted butter and milk, then add
to the pumpkin purée and stir to combine. Pour into the tart shell and cook for
10 mins, then reduce the temperature to 180C/160C fan/gas 4. Continue to bake
for 35-40 mins until the filling has just set.  Leave to cool, then remove the
pie from the tin. Mix the remaining cinnamon with the icing sugar and dust over
the pie. Serve chilled.

Ingredients:
--------------------------------------------------------------------------------
  750g Pumpkin              350g Shortcrust Pastry    Dusting Plain Flour
  140g Caster Sugar         ½ tsp Salt                ½ tsp Nutmeg
  1 tsp Cinnamon            2 Beaten Eggs             25g Butter
  175g Milk                 1 tblsp Icing Sugar

Command:
```

- The ingredients section should list the ingredients in 3 equal length columns and only displaying enough lines as there is data.

Hints

Do not procrastinate and use the lab time wisely, it's the only guaranteed time you will have to ask the instructor for assistance.

Break the final project into smaller "objectives", I recommend using the displays (List Categories, List Meals by Category, List Meal by Name, Random Meal, List Areas, List Meals by Area). When you start coding these objectives, make sure you complete and fully test the objective before moving on to the next.

If you don't know where to start, refer to the Getting Started document. It will help you by covering the List Categories and List Meals by Category objective. After that, move to the List Areas and List Meals by Area. These are very similar to the List Categories and List Meals by Category. Finally, finish with the Meal by Name and Display Random Meal.

For listing the meals by category, meals by area, and meal by name, make sure to account for when the user enters something that is not found in the database. In these cases, you should display a message letting the user know that the entry was not found.

Since the Meal by Name and Random Meal display the same information, both can use a common function to display the meal information. You can also use the same API request for retrieving the meal with just a minor code adjustment.

After you get to a point where the major functionality is working, I recommend submitting the files into Canvas. When you do this, add a note in the comments section letting me know you're still working it and I'll hold off on grading until you change the note to let me know your done of the due date.

Before final submission of your final project, make sure you review the final project rubric to validate your application is complete.