

ACCA5029 - Assignment Two

by Curtis Louis Stephens

FILE	16954_CURTIS_LOUIS_STEPHENS_ACCA5029_-_ASSIGNMENT_TWO_314650_870485201.DOCX (454.81K)		
TIME SUBMITTED	14-JAN-2021 01:08AM (UTC+0000)	WORD COUNT	977
SUBMISSION ID	142009158	CHARACTER COUNT	4192



ACCA5029 – NETWORK PROGRAMMABILITY ASSIGNMENT 2



Curtis Stephens

UNIVERSITY OF WALES TRINITY SAINT DAVID 1903184

Contents

1. Introduction	2
2. More about my application	2
3. Run-through	3
3.1 Searching by name	3
3.2 Searching by criteria	4
3.3 ABV greater than or less than	5
3.4 IBU greater or less than	5
3.5. EBC greater or less than	6
3.5 Food pairing	7
3.6 Brewed before and after	7
3.7 Hops, yeast and malt	8
4. Evaluation	9

1. Introduction

In this document I will be giving details of the implementation of my chosen API for this project.

My chosen API is PunkAPI, which is a database of Brewdog-branded beers. The API allows the user to search for a beer either by name, or by criteria such as ABV percentage, the bitterness, colour or by food pairings.

Towards the end of this document, I will also be reflecting on the application which I have created and looking at how I could make it more user-friendly as well as features which could be added.

2. More about my application

As mentioned above, my application uses an API called PunkAPI. This API has a database of all the beers made by Brewdog, and even contains information such as how much of each hops are used in the drink, as well as basic information such as brew dates, ABV percentage and food pairings.

The application which I have created allows people to search this database, either by name or by criteria. The criteria are as follows:

- ABV greater or less than
- IBU (Bitterness) less or greater than
- EBC (Darkness) less or greater than
- Brewed before or after
- Food pairings
- Yeast used
- Hops used
- Barley used

3. Run-through

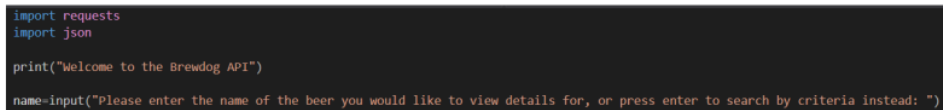
3.1 Searching by name

Below I will be posting screenshots from a code run-through, as well as snippets of the code that has been run.



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python7.5\python.exe
Welcome to the Brewdog API
Please enter the name of the beer you would like to view details for, or press enter to search by criteria instead: Clockwork
How many results would you like to see?: 2
{
  "id": 110,
  "name": "Clockwork Tangerine",
  "tagline": "Citrus IPA.",
  "first_brewed": "2018",
  "description": "Winner of the 2017 Prototype Challenge, Clockwork Tangerine is a session strength IPA, infused with tangerine, and appears as our new Q2 Seasonal, replacing Hop Fiction.",
  "image_url": null,
  "abv": 4.5,
  "ibu": 37,
  "target_fg": 1012,
  "target_ag": 1047,
  "ch": 20,
  "rm": 10,
  "ph": 4.2,
  "attenuation_level": 75,
  "volume": {
    "value": 20,
    "unit": "litres"
  },
  "bottle_volume": {
    "value": 20,
    "unit": "litres"
  },
  "method": {
    "mash_temp": {
      "temp": {
        "value": 66,
        "unit": "celsius"
      },
      "duration": 45
    }
  }
}
```

This is what the user sees when they first open the program. The user can either type the name of a drink, or press enter to search by criteria.

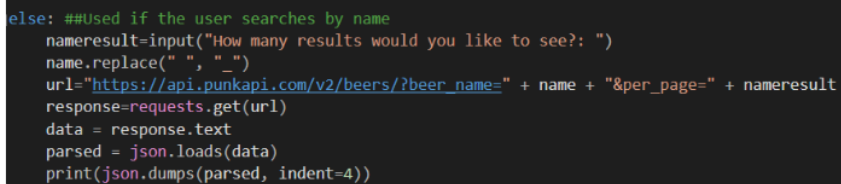


```
import requests
import json

print("Welcome to the Brewdog API")

name=input("Please enter the name of the beer you would like to view details for, or press enter to search by criteria instead: ")
```

The name search is in an else statement, the reason for which will be seen when I explain the criteria search.



```
else: ##Used if the user searches by name
    nameresult=input("How many results would you like to see?: ")
    name.replace(" ", "_")
    url="https://api.punkapi.com/v2/beers/?beer_name=" + name + "&per_page=" + nameresult
    response=requests.get(url)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
```

In the above code, the URL is set to simply search by name. This performs a fuzzy search, so even if the users search does not match a record exactly, it will show similar records to what has been entered. For example, if I search 'state' it will show beers with the word 'state' in the name. The closest match will appear at the top, however.

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python114\python.exe
Welcome to the brewing API.
Please enter the name of the beer you would like to view details for, or press enter to search by criteria instead: state
How many results would you like to see? 3
{
  "id": 11,
  "name": "Honey State",
  "tagline": "Brewed from Honey Ale.",
  "first_brewed": "06/2007",
  "description": "Brewing a full flavoured craft beer at 0.5% is no easy task. Packed with loads of Centennial, Amarillo, Columbus, Cascade and Simcoe hops, dry hopped to the brim and sitting at 15 IBU, Honey State is a force to be reckoned with. With a backbone of 8 different specialty malts, Honey State will tantalise your taste buds and leave you yearning for more.",
  "image_url": "https://images.punkapi.com/v2/11.png",
  "abv": 6.5,
  "ibu": 15,
  "target_iv": 1000,
  "target_ivg": 1000,
  "alc": 20,
  "wsg": 15,
  "ph": 4.4,
  "attenuation_level": 28.6,
  "volume": {
    "value": 20,
    "unit": "litres"
  },
  "serving_size": {
    "value": 25,
    "unit": "litres"
  },
  "serving": {
    "temp": {
      "value": 45,
      "unit": "celcius"
    },
    "duration": 30
  }
}
```

3.2 Searching by criteria

If the user presses enter at the first prompt, a list of options is displayed. The user must choose a criteria from this list to search by.

```
if name == "": #Used if the user presses enter to the first input.
    print("Please select a criteria to search by: ")
    print("1 = ABV greater than")
    print("2 = ABV less than")
    print("3 = IBU (Bitterness) greater than")
    print("4 = IBU (Bitterness) less than")
    print("5 = EBC (Darkness) greater than")
    print("6 = EBC (Darkness) less than")
    print("7 = Food Pairings")
    print("8 = Brewed Before")
    print("9 = Brewed After")
    print("10 = Hops used")
    print("11 = Malt used")
    print("12 = Yeast used")
```

The user is then prompted to choose a number from this list, and each number is linked to an if statement which is triggered when the relevant number is chosen.

3.3 ABV greater than or less than

```
8 = Brewed Before
9 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 1
Please input your desired ABV percentage to search higher than (Without % sign):
```

```
8 = Brewed Before
9 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 2
Please input your desired ABV percentage to search less than (Without % sign):
```

The code for these is very similar, with just the URL changing very slightly.

```
##&per_page= allows the user to select how many results they wish to see.
choice = input("Please select an item from the list: ") ## User enters a number from the list
if choice == "1": ##ABV higher than
    abvgt = input("Please input your desired ABV percentage to search higher than (Without % sign): ")
    abvgtresult = input("How many results would you like to see?: ")
    abvgturl="https://api.punkapi.com/v2/beers/?abv_gt=" + abvgt + "&per_page=" + abvgtresult
    response=requests.get(abvgturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
if choice == "2": ##ABV lower than
    abvlt = input("Please input your desired ABV percentage to search less than (Without % sign): ")
    abvltresult = input("How many results would you like to see?: ")
    abvlturl="https://api.punkapi.com/v2/beers/?abv_lt=" + abvlt + "&per_page=" + abvltresult
    response=requests.get(abvlturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
```

As can be seen, only the end of the URL as well as a couple of variable names change depending on which choice the user makes. The code prints in the same fashion as the name code, but only items with less or greater than the chosen ABV percentage will be displayed.

3.4 IBU greater or less than

This allows the user to search for beers by the level of bitterness, with lower numbers being less bitter than higher numbers.

The code for this is also very similar, with just the end of the URL being changed for each query.

```
if choice == "3": ##IBU higher than
    ibugt = input("Please input your desired IBU (bitterness level) to search greater than: ")
    ibugtresult = input("How many results would you like to see?: ")
    ibugturl="https://api.punkapi.com/v2/beers/?ibu_gt=" + ibugt + "&per_page=" + ibugtresult
    response=requests.get(ibugturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
if choice == "4": ##IBU lower than
    ibult = input("Please input your desired IBU (bitterness level) to search less than: ")
    ibultresult = input("How many results would you like to see?: ")
    ibulturl="https://api.punkapi.com/v2/beers/?ibu_lt=" + ibult + "&per_page=" + ibultresult
    response=requests.get(ibulturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
```

```

8 = Brewed Before
9 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 3
Please input your desired IBU (bitterness level) to search greater than:

```

```

8 = Brewed Before
9 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 4
Please input your desired IBU (bitterness level) to search less than:

```

3.5. EBC greater or less than

This allows the user to search by the lightness of a beer. The higher the EBC, the darker the beer is.

Again, this code is similar with just a few tweaks to the URL and variables.

```

if choice == "5": ##EBC/Darkness higher than
    ebcgt = input("Please input your desired EBC (Darkness) to search greater than: ")
    ebcgtresult = input("How many results would you like to see")
    ebcgturl="https://api.punkapi.com/v2/beers?ebc_gt=" + ebcgt + "&per_page=" + ebcgtresult
    response=requests.get(ebcgturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
if choice == "6": ##EBC/Darkness lower than
    ebclt = input("Please input your desired EBC (Darkness) to search less than: ")
    ebcltresult = input("How many results would you like to see?: ")
    ebclturl="https://api.punkapi.com/v2/beers?ebc_lt=" + ebclt + "&per_page=" + ebcltresult
    response=requests.get(ebclturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))

```

```

8 = Brewed Before
9 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 5
Please input your desired EBC (Darkness) to search greater than:

```

```

8 = Brewed Before
9 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 6
Please input your desired EBC (Darkness) to search less than:

```

3.5 Food pairing

This is like searching by name, just with a different field name. This once again makes a 'fuzzy' search which searches for matches similar to what the user has typed in.

For example, below I will search for beers that pair up well with ham, and the program will give me results for food pairings that are similar to ham.

In this code, there is also a line that changes spaces to underscores, as URLs cannot have spaces.

```
if choice == 7: ##Food pairing
    pairing = input("Please input an item of food to find a beer to pair it with: ")
    pairingresult = input("How many results would you like to see?: ")
    pairing.replace(" ", "_")
    pairingurl="https://api.punkapi.com/v2/beers/?food=" + pairing + "&per_page=" + pairingresult
    response=requests.get(pairingurl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
```

```
9 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 7
Please input an item of food to find a beer to pair it with: ham
How many results would you like to see?: 2
[
  {
    "id": 23
```

```
  },
  "food_pairing": [
    "Oysters",
    "Hickory smoked ham",
    "Rocky Road"
  ],
```

3.6 Brewed before and after

The code for these two are the same, with the user being prompted to input a year then a month. The program will then insert these into the URL and make a search for beers brewed before the selected date.

```
if choice == "8": ##first brew date before
    yearbefore = input("Please input a year (YYYY) to search: ")
    monthbefore = input("Please input a month (MM) to search: ")
    datebresult = input("How many results would you like to see?: ")
    dateburl="https://api.punkapi.com/v2/beers/?brewed_before=" + monthbefore + "-" + yearbefore + "&per_page=" + datebresult
    response=requests.get(dateburl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
if choice == "9": ##first brew date after
    yearafter = input("Please input a year (YYYY) to search: ")
    monthafter = input("Please input a month (MM) to search: ")
    datearesult = input("How many results would you like to see?: ")
    dateaurl="https://api.punkapi.com/v2/beers/?brewed_after=" + monthafter + "-" + yearafter + "&per_page=" + datearesult
    response=requests.get(dateaurl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
```

```

Please input a year (YYYY) to search: 2011
Please input a month (MM) to search: 08
How many results would you like to see?: 3
[
  {
    "id": 1,
    "name": "Buzz",
    "tagline": "A Real Bitter Experience.",
    "first_brewed": "09/2007",
    "description": "A light, crisp and bitter IPA brewed with English and American hops. A small batch brewed only o
    nce."
  }
]

```

```

12 = Yeast used
Please select an item from the list: 9
Please input a year (YYYY) to search: 2014
Please input a month (MM) to search: 12
How many results would you like to see?: 3
[
  {
    "id": 3,
    "name": "Berliner Weisse With Yuzu - B-Sides",
    "tagline": "Japanese Citrus Berliner Weisse.",
    "first_brewed": "11/2015",
  }
]

```

3.7 Hops, yeast, and malt

The code for these three is pretty much the same, again with just the variables and the URLs being changed. The code used is below.

```

if choice == "10": ##Hops
    hops = input("Please input hops to search for: ")
    hopsresult = input("How many results would you like to see?: ")
    hopsurl = "https://api.punkapi.com/v2/beers/?hops=" + hops + "&per_page=" + hopsresult
    response=requests.get(hopsurl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
if choice == "11": ##Malt
    malt = input("Please input malt to search for: ")
    maltresult = input("How many results would you like to see?: ")
    malturl = "https://api.punkapi.com/v2/beers/?malt=" + malt + "&per_page=" + maltresult
    response=requests.get(malturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
if choice == "12": ##Yeast
    yeast = input("Please input yeast to search for: ")
    yeastresult = input("How many results would you like to see?: ")
    yeasturl = "https://api.punkapi.com/v2/beers/?yeast=" + yeast + "&per_page=" + yeastresult
    response=requests.get(yeasturl)
    data = response.text
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))

```

```

9 = Brewed before
10 = Brewed After
10 = Hops used
11 = Malt used
12 = Yeast used
Please select an item from the list: 10
Please input hops to search for:

```

```

11 = Malt used
12 = Yeast used
Please select an item from the list: 11
Please input malt to search for:

```

```

11 = Malt used
12 = Yeast used
Please select an item from the list: 12
Please input yeast to search for:

```

4. Evaluation

I feel that I have created an application which is fairly user friendly. The application does everything that it is supposed to and has been tested to ensure that it works correctly which it does. However, there are some things that I would do different in the future.

- Allowing users to search by multiple criteria
- Outputting the result in a more user-friendly way

I would also learn more about how APIs work, and then add a way to add, delete and change records if required.

In addition, I think that a way to search by more criteria could be added, however I could not find the links for the respective criteria and as such it may not be possible.

I would also like to add a way for the user to make a second search if they need to, and add validation to the application, so if the user enters an input that is not valid, an error is thrown, and the user must enter the information that they wish to search for again.

ACCA5029 - Assignment Two

GRADEMARK REPORT

FINAL GRADE

72/100

GENERAL COMMENTS

Instructor

Implementation (70): 52

Report (30): 20

A very good effort - quite a bit of work clearly went in here. You could've used functions to make it a fair bit shorter (i.e. most of the internal code for each case is repeated) but nonetheless, some nice work.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10
