

__init__ HBNB – Console (v1)

BY CHRIS STAMPER

PyPack

Python packages help us stay organized! This way, similar tools, files, features, etc. can be grouped together.

Packages are distinguished from directories by their `__init__.py` files; these files can either contain initialization code or be blank.

Packages can be used in multiple separate scripts simultaneously.

```
Demo_Console> Exiting...
● grazingtatanka@PC:~/evilcode/console$ tree
.
├── README.md
├── demo_console.py
├── package_demo
│   ├── __init__.py
│   ├── are.py
│   ├── package.py
│   └── we.py
├── package_demo.egg-info
│   ├── PKG-INFO
│   ├── SOURCES.txt
│   ├── dependency_links.txt
│   └── top_level.txt
├── setup.py
├── super_serial
│   ├── console.py
│   ├── models
│   │   ├── __init__.py
│   │   ├── base.py
│   │   └── cereal.py
│   ├── engine
│   │   ├── __init__.py
│   │   └── file_storage.py
│   └── tests
│       ├── test_console.py
│       ├── test_models
│       │   ├── __init__.py
│       │   ├── test_base.py
│       │   ├── test_cereal.py
│       │   └── test_engine
│       │       ├── __init__.py
│       │       └── test_file_storage.py
└── 8 directories, 23 files
○ grazingtatanka@PC:~/evilcode/console$
```

cmd Module

1. First import the cmd module
2. Create a class that inherits from the cmd.Cmd
3. Create a .cmdloop() of said class
(class_here.cmdloop())
4. Set the prompt to be displayed based on tty mode
5. Returning True breaks us out of the loop!
6. EOF (End of File) signal shortcut remains Ctrl+D (do_EOF)
7. You now have a CLI similar to Simple Shell >:D

```
evilcode > console > demo_console.py > Demo_Console
1  #!/usr/bin/env python3
2  """Demo console application"""
3
4  import cmd
5
6  class Demo_Console(cmd.Cmd):
7      prompt = 'Demo_Console> '
8
9      def do_greet(self, name):
10         """greet [name] - Greet the named person"""
11         if name:
12             print(f"Hello, {name}!")
13         else:
14             print("Hello!")
15
16     def do_exit(self, arg):
17         """exit - Exit the interpreter"""
18         print("Exiting...")
19         return True
20
21     def do_EOF(self, line):
22         return True

TERMINAL  DEBUG CONSOLE  PROBLEMS  OUTPUT  COMMENTS
o grazingtatanka@PC:~/evilcode/console$ ./demo_console.py
Demo_Console> greet c20
Hello, c20!
Demo_Console> 
```

Testing test tests

- ▶ Your 'tests' directory usually goes at the base of your project repo.
- ▶ Tests directory files mirror project architecture (e.g. /models/base.py is tested in /tests/test_models/test_base.py)
- ▶ **REMEMBER** to make a 'tests/test_console.py' even if it's empty
- ▶ Utilize setup && teardown methods to govern behavior before and/or following a test (e.g. reset storage)
- ▶ Have fun! Testing for test tests can be frustrating, so try to enjoy it.
- ▶ `python3 -m unittest discover tests`

Meet Jason!

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data-interchange format
- It is easy for humans to read and write
- It's easy for machines to parse and generate
- JSON is a text format that is completely language independent
- Python has built in json support (import json)

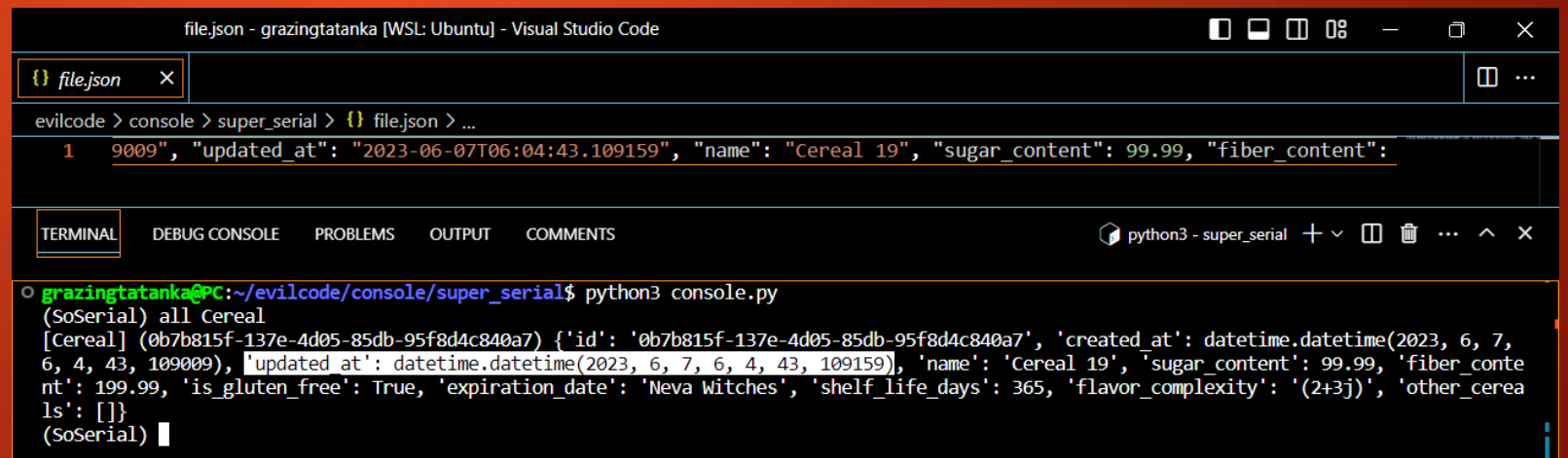
I'm Super Serial-izing

<class 'BaseModel'> -> to_dict() -> <class 'dict'> -> JSON dump -> <class 'str'> -> FILE -
> <class 'str'> -> JSON load -> <class 'dict'> -> <class 'BaseModel'>

```
{"Cereal.0b7b815f-137e-4d05-85db-95f8d4c840a7":  
  {"id": "0b7b815f-137e-4d05-85db-95f8d4c840a7",  
    "created_at": "2023-06-07T06:04:43.109009", "updated_at": "2023-06-07T06:04:43.109159",  
    "name": "Cereal 19", "sugar_content": 99.99,  
    "fiber_content": 199.99, "is_gluten_free": true,  
    "expiration_date": "Neva Witches", "shelf_life_days": 365,  
    "flavor_complexity": "(2+3j)", "other_cereals": [], "__class__": "Cereal"}}
```

What datetime is it?

- ▶ Time data is present in nearly all data sets
- ▶ Enables tracking changes over time, analyzing trends
- ▶ Datetime object can be created with datetime class built in `.now()` method
- ▶ They can be converted / saved to string using `.isoformat()`
- ▶ To make new datetimes on reload, you can use `.strptime(...)`



The screenshot shows a Visual Studio Code window titled "file.json - grazingtatanka [WSL: Ubuntu] - Visual Studio Code". The editor displays a JSON file named "file.json" with the following content:

```
1 9009", "updated_at": "2023-06-07T06:04:43.109159", "name": "Cereal 19", "sugar_content": 99.99, "fiber_content":
```

Below the editor, the "TERMINAL" panel is active, showing the output of a Python script. The prompt is `grazingtatanka@PC:~/evilcode/console/super_serial$ python3 console.py`. The output is:

```
(SoSerial) all Cereal  
[Cereal] (0b7b815f-137e-4d05-85db-95f8d4c840a7) {'id': '0b7b815f-137e-4d05-85db-95f8d4c840a7', 'created_at': datetime.datetime(2023, 6, 7, 6, 4, 43, 109009), 'updated_at': datetime.datetime(2023, 6, 7, 6, 4, 43, 109159), 'name': 'Cereal 19', 'sugar_content': 99.99, 'fiber_content': 199.99, 'is_gluten_free': True, 'expiration_date': 'Neva Witches', 'shelf_life_days': 365, 'flavor_complexity': '(2+3j)', 'other_cereals': []}  
(SoSerial)
```

UUID

- ▶ UUID === 'Universally Unique Identifier'
- ▶ Format: 8-4-4-4-12 for a total of 36 characters
- ▶ From uuid import uuid4 ← random number based

Helps us differentiate similar, but not the same, class instances

e.g. [Cereal] (0b7b815f-137e-4d05-85db-95f8d4c840a7) {'id': '0b7b815f-137e-4d05-85db-95f8d4c840a7', 'created_at': datetime.datetime(2023, 6, 7, 6, 4, 43, 109009), 'updated_at': datetime.datetime(2023, 6, 7, 6, 4, 43, 109159), 'name': 'Cereal 19', 'sugar_content': 99.99, 'fiber_content': 199.99, 'is_gluten_free': True, 'expiration_date': 'Neva Witches', 'shelf_life_days': 365, 'flavor_complexity': '(2+3j)', 'other_cereals': []}

[Cereal] (f1f8ea78-d3ed-4b5f-8e88-ec31d9fcafba) {'id': 'f1f8ea78-d3ed-4b5f-8e88-ec31d9fcafba', 'created_at': datetime.datetime(2023, 6, 7, 6, 25, 10, 97766), 'updated_at': datetime.datetime(2023, 6, 7, 6, 25, 10, 98069), 'name': 'Cereal 19', 'sugar_content': 99.99, 'fiber_content': 199.99, 'is_gluten_free': True, 'expiration_date': 'Neva Witches', 'shelf_life_days': 365, 'flavor_complexity': (2+3j), 'other_cereals': []}