0x00. AirBnB clone - The console

Group project · Python · OOP

Concepts

For this project, we expect you to look at these concepts:

- Python packages
- AirBnB clone

hbnb

Background Context

Welcome to the AirBnB clone project!

Before starting, please read the **AirBnB** concept page.

First step: Write a command interpreter to manage your AirBnB objects.

This is the first step towards building your first full-web application: the **AirBnB clone**. This first step is very important because you will use this step about all the following projects with all these following projects: HTML/CSS templating, database storage, API, front-end integration...

Each task is linked and will help you to:

- put in place a parent class (called `BaseModel`) to take care of the initialization, serialization and deserialization of your future instances
- create a simple flow of serialization/deserialization: Instance <-> Dictionary <-> JSON string <-> file
- create all classes used for AirBnB (`BaseModel`, `User`, `State`, `City`, `Place`, `Amenity` and `Review`) that inherit from `BaseModel`
- create the first abstracted storage engine of the project: file storage.
- create all unittests to validate all our classes and storage engine

What's a command interpreter?

Do you remember the Shell? It's exactly the same but limited to a specific use-case. In our case, we want to be able to manage the objects of our project:

- Create a new object (ex: a new User or a new Place)
- Retrieve an object from a file, a database etc...
- Do operations on objects (count, compute stats, etc...)
- Update attributes of an object
- Destroy an object

Resources

Read or watch:

- cmd module
- cmd module in depth
- **packages** concept page
- uuid module
- datetime
- sys module
- asphalt-mongo
- Python test cheatsheet
- unit tests in stdlib
- Mock a class

Learning Objectives

At the end of the project, you are expected to be able to `explain to anyone`, **without the help of Google**:

General

- How to create a Python package
- How to create a command interpreter in Python using the `cmd` module
- What is Unit testing and how to implement it in a large project
- How to serialize and deserialize a Class
- How to write and read a JSON file
- How to manage datetime
- What is an `UUID`
- What is `args` and how to use it
- What is `*kwargs` and how to use it
- How to handle named arguments in a function

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- Because this project is intended to build upon each other, you will be using this project for copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the progress.

Requirements

Python Scripts

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using python3 (version 3.8.5)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/python3`
- A `README.md` file at the root of the folder of the project, is mandatory
- Your code should use the `pycodestyle` (version `2.7.0`)
- All your files must be executable
- All your files should run by `./file_name` (ex: `./console.py`)
- All your modules should have a documentation (`python3 -c "import console; help(console)"`)
- All your classes should have a documentation (`python3 -c "import models.base_model; help(models.base_model)`)
- All your functions (inside and outside a class) should have a documentation (`pycodestyle`)
- `pycodestyle tests/*` and `pycodestyle models/*` and `pycodestyle console.py` must return no output
- You can also test the file by using the unittest module `python3 -m unittest discover tests`
- All your modules should have a documentation (`python3 -c "import models; help(models)"`)
- All your classes should have a documentation (`python3 -c "import models.base_model; help(models.BaseModel)"`)
- All your functions (inside and outside a class) should have a documentation (`pycodestyle`)
- `pycodestyle tests/*` and `pycodestyle models/*` and `pycodestyle console.py` must return no output
- `python3 -m unittest discover tests` and `python3 -m unittest discover models`
- We strongly encourage you to work together on test cases, so that you don't miss any edge case

GitHub

There should be one your project repository: gym if you clone/fork/whatever a project repository with the same name before the second deadline, you risk a 0% score.

More Info

Execution

Your shell should work like this in interactive mode:

```
$ ./console.py
(hbnb) help

Documented commands (type help <topic>):
========================================
EOF  help  quit

(hbnb)
(hbnb) quit
$
```

But also in non-interactive mode: (like the Shell project in C)

```
$ echo "help" | ./console.py
(hbnb)

Documented commands (type help <topic>):
========================================
EOF  help  quit

(hbnb)
$ cat test_help
help

$ cat test_help | ./console.py
(hbnb)

Documented commands (type help <topic>):
========================================
EOF  help  quit

(hbnb)
$
```

All tests should also pass in non-interactive mode: `python3 -m unittest discover tests/*.py`

Video library 0:00:00

Search by title

Tasks

0. README, AUTHORS

mandatory

Score: 0.0% (Checks completed: 100.0%)

- Write a `README.md`:
  - description of the project
  - description of the command interpreter:
    - how to start it
    - how to use it
    - examples
- You should end your file with a `\n`. Be at the root of your repository, listing all individuals having contributed content to the repository. For example, if you are the only one: `echo "holberton user1@gmail.com" > AUTHORS`
- You should use branches and pull requests on GitHub - it will help you to learn how to organize your work

Repo:

- GitHub repository: `ALX\_SE\_LISAM`
- File: `README.md`, `AUTHORS`

Close | Check your code | Get a sandbox | Ask Review

1. Be pycodestyle compliant!

mandatory

Score: 91.23% (Checks completed: 100.0%)

Write beautiful code that passes the pycodestyle checks.

Repo:

- GitHub repository: `ALX\_SE\_LISAM`

Close | Check your code | Get a sandbox | Ask Review

2. Unittests

mandatory

Score: 100.0% (Checks completed: 100.0%)

All your files, classes, functions must be tested with unit tests

```
guillaume@ubuntu:~/hbnb$ python3 -m unittest discover tests
.....................................................................
.....................................................................
.....................................................................
.....................................................................
Ran 197 tests in 12.129s

OK
guillaume@ubuntu:~/hbnb$
```

Note that this is just an example, the number of tests you create can be different from the above example.

Warning:

Unit tests must also pass in non-interactive mode:

```
guillaume@ubuntu:~/hbnb$ echo "python3 -m unittest discover tests" | bash
.....................................................................
.....................................................................
.....................................................................
.....................................................................
Ran 197 tests in 12.129s

OK
guillaume@ubuntu:~/hbnb$
```

Repo:

- GitHub repository: `ALX\_SE\_LISAM`
- File: `tests/test_console.py`

Close | Check your code | Get a sandbox | Ask Review

3. BaseModel

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a class `BaseModel` that defines all common attributes/methods for other classes:

- Public instance attributes:
  - `id`: string - assign with an `uuid` when an instance is created:
    - you can use `str(uuid.uuid4())` to generate unique `id` but don't forget to convert to a string
    - the goal is to have unique `id` for each `BaseModel`
  - `created_at`: datetime - assign with the current datetime when an instance is created
  - `updated_at`: datetime - assign with the current datetime when an instance is created and it will be updated every time you change your object
  - `__str__`: should print: `[<class name>] (<self.id>) <self.__dict__>`
- Public instance methods:
  - `save(self)`: updates the public instance attribute `updated_at` with the current datetime
  - `to_dict(self)`: returns a dictionary containing all keys/values of `__dict__` of the instance:
    - by using `self.__dict__`, only instance attributes set will be returned
    - a key `__class__` must be added to this dictionary with the class name of the object
    - `created_at` and `updated_at` must be converted to string object in ISO format:
      - format: `%Y-%m-%dT%H:%M:%S.%f` (ex: `2017-06-14T22:31:03.285259`)
    - `kwargs` - `__class__` key must be the current class name of the object
    - This method will be the first piece of the serialization/deserialization process: create a dictionary representation with "simple object type" of our `BaseModel`

```
guillaume@ubuntu:~/hbnb$ cat test_base_model.py
#!/usr/bin/python3
from models.base_model import BaseModel

my_model = BaseModel()
my_model.name = "My First Model"
my_model.my_number = 89
print(my_model.id)
print(my_model.__str__())
print(my_model)
print(type(my_model.created_at))
print("---")
print(my_model.__dict__)
print("---")
print("JSON of my_model:")
for key in my_model.__dict__.keys():
    print("{}: {}".format(key, type(my_model.__dict__[key])))

guillaume@ubuntu:~/hbnb$ ./test_base_model.py
[BaseModel] (bb64d2cb-7ba5-4110-9a7b-8d0968b6a274) {'my_number': 89, 'name': 'My First Model', 'updated_at': datetime.datetime(2017, 9, 15, 21, 16, 32, 294877), '__class__': 'BaseModel', 'id': 'bb64d2cb-7ba5-4110-9a7b-8d0968b6a274', 'created_at': datetime.datetime(2017, 9, 15, 21, 16, 32, 294877)}
[BaseModel] (bb64d2cb-7ba5-4110-9a7b-8d0968b6a274) {'my_number': 89, 'name': 'My First Model', 'updated_at': datetime.datetime(2017, 9, 15, 21, 16, 32, 294877), '__class__': 'BaseModel', 'id': 'bb64d2cb-7ba5-4110-9a7b-8d0968b6a274', 'created_at': datetime.datetime(2017, 9, 15, 21, 16, 32, 294877)}
<class 'datetime.datetime'>
---
{'my_number': 89, 'name': 'My First Model', '__class__': 'BaseModel', 'updated_at': "2017-09-15T21:16:32.294877", 'id': 'bb64d2cb-7ba5-4110-9a7b-8d0968b6a274', 'created_at': "2017-09-15T21:16:32.294877"}
---
JSON of my_model:
my_number: <class 'int'> - 89
name: <class 'str'> - My First Model
updated_at: <class 'str'> - 2017-09-15T21:16:32.294877
id: <class 'str'> - bb64d2cb-7ba5-4110-9a7b-8d0968b6a274
created_at: <class 'str'> - 2017-09-15T21:16:32.294877

guillaume@ubuntu:~/hbnb$
```

Repo:

- GitHub repository: `ALX\_SE\_LISAM`
- File: `models/base_model.py`, `tests/test_models/test_base_model.py`

Close | Check your code | Get a sandbox | Ask Review

4. Create BaseModel from dictionary

mandatory

Score: 100.0% (Checks completed: 100.0%)

Previously we created a method to generate a dictionary representation of an instance (method `to_dict()`).

Now it's time to re-create an instance with this dictionary representation.

```
>>create "BaseModel" id my_id=12 ip ip=localhost name="Holberton"
```

Update: `models/base_model.py`

- `**kwargs` (double `*` for keyword argument) arguments for the constructor of a `BaseModel`: (more information inside the **AirBnB clone** concept page)
  - `kwargs` can not be empty
  - if `kwargs` is not empty:
    - each key of this dictionary is an attribute name (**Note**: `__class__` from `kwargs` is the only one that should not be added as an attribute. See the example output_format)
    - each value of this dictionary is the value of this attribute name
    - **Warning**: `created_at` and `updated_at` are strings in this dictionary but inside your `BaseModel` instance is working with `datetime` object. You have to convert these strings into `datetime` object. You gonna know the string format of these `datetime`
    - otherwise:
      - create an `id` and `created_at` as you did previously (new instance)

```
guillaume@ubuntu:~/hbnb$ cat test_base_model.py
#!/usr/bin/python3
from models.base_model import BaseModel

my_model = BaseModel()
my_model.name = "My First Model"
my_model.my_number = 89
print(my_model.id)
print(my_model)
print(type(my_model.created_at))
print("---")
my_model_json = my_model.to_dict()
print(my_model_json)
print("JSON of my_model:")
for key in my_model_json.keys():
    print("{}: {} - {}".format(key, type(my_model_json[key]), my_model_json[key]))

guillaume@ubuntu:~/hbnb$ ./test_base_model.py
[BaseModel] (b6f6f7e0-894d-4fb7-ae66-8dfa2d393a15) {'id': '5f1d', 'created_at': datetime.datetime(2017, 9, 15, 21, 16, 32, 294877), '__class__': 'BaseModel', 'updated_at': datetime.datetime(2017, 9, 15, 21, 16, 32, 294877), 'name': 'My First Model', 'my_number': 89}
<class 'datetime.datetime'>
---
{'my_number': 89, '__class__': 'BaseModel', 'updated_at': '2017-09-15T21:16:32.294877', 'id': '5f1d', 'name': 'My First Model', 'created_at': '2017-09-15T21:16:32.294877'}
---
JSON of my_model:
my_number: <class 'int'> - 89
__class__: <class 'str'> - BaseModel
updated_at: <class 'str'> - 2017-09-15T21:16:32.294877
id: <class 'str'> - 5f1d
name: <class 'str'> - My First Model
created_at: <class 'str'> - 2017-09-15T21:16:32.294877

guillaume@ubuntu:~/hbnb$
```

Repo:

- GitHub repository: `ALX\_SE\_LISAM`
- File: `models/base_model.py`, `tests/test_models/test_base_model.py`

Close | Check your code | Get a sandbox | Ask Review

```
print("Open the new_bookmark.created_at")

print("-")
print("Open model in my_new_model")

pulisanbookbuilder:~/Bin$ ./test_new_bookmark_dict.py
{'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f',
'bookmark': {'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'name': 'My_First_Book',
'_data': {'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'name': 'My_First_Book',
'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'created_at': '2017-09-28T02:03:56.252290',
'updated_at': '2017-09-28T02:03:56.252292', 'notes': 'My_First_Note'}}
'created_at': '2017-09-28T02:03:56.252290',
'updated_at': '2017-09-28T02:03:56.252292',
'notes': 'My_First_Note',
'my_number': 0,
'updated_at': '2017-09-28T02:03:56.252292',
'created_at': '2017-09-28T02:03:56.252290',
'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f',
'_class': 'models.Bookmark.Bookmark',
'name': 'My_First_Book',
'my_number': 0}
pulisanbookbuilder:~/Bin$
```

Repo:

- GitHub repository: `bitbucket.org/pulisan/pulisanbookbuilder`
- File: `pulisanbookbuilder/models/Bookmark.py`

Closed | Check your code | Get a sandbox | Qt Review

5. Store first object

MANDATORY

Score: 96.67% (Checks completed: 96.67%)

Now we can recreate a `bookmark` from another one by using a dictionary representation:

```
./class "TheseBooks" -i -c0,d1,t0 -v view 'dict' -v view 'display' -v view 'TheseBooks'
```

It's great but it's still not persistent: every time you launch the program, you don't restore all objects created before... The first way you will see here is to save these objects to file.

Writing the dictionary representation to a file won't be secure:

- Python doesn't know how to convert a string to a dictionary *beautiful*
- It's not human readable
- Using this file in another program in Python or other language will be hard.

So, you will convert the dictionary representation to a JSON string. JSON is a standard representation of a data structure. With this format, humans can read and programming languages can read and write JSON strings.

Now the flow of serialization/deservialization will be:

```
./class "TheseBooks" -i -c0,d1,t0 -v view 'dict' -v JSON dump -v view 'dict' -v FILE -v view 'dict' -v
JSON load -v view 'dict' -v view 'TheseBooks'
```

Magic right?

Terms:

- **simple Python data structures**: Dictionaries, arrays, number and string are `simple` or `primitive` or `basic` or `standard` types.
- **JSON string representation**: This is a string representation of a JSON object, array, number, or string. Example: `{"name": "Pulisanbook", "id": 0}`, `[1, 2, 3]`, `"a string"`, `1.0`.

Write a class `JsonSerializer`. that serializes instances to a JSON file and deserializes JSON file to instances:

- `models/serializers/JsonSerializer.py`
- Private class attributes:
  - `__file_path`: string - path to the JSON file dirs `/var/tmp/`
  - `__dictionary`: empty list at start and objects by `__dictionary.append(obj)`. Use to store a `bookmark` object with `__dictionary.append(bookmark)`
- Public instance methods:
  - `__init__()`: calls the dictionary `__dictionary = []`
  - `__setattr__(self, name, value)`: if `name == __file_path` set `__file_path = value`
  - `__getattr__(self, name)`: if `name == __file_path` return `__file_path`
  - `__str__()`: if `__dictionary` is empty return `"[]"`, dump and return `"[" + ", ".join(__dictionary) + "]"`
  - `__repr__()`: return `__str__()`
  - `__del__()`: if `__file_path` is not empty, dump the JSON `__dictionary` into `__file_path` textfile; otherwise, do nothing. If the file doesn't exist, the `__del__()` method should be raised.

Update `./class` to create a unique `JsonSerializer` instance for your application:

- import `JsonSerializer`
- create the unique `JsonSerializer` instance in `FileStorage`
- call `serialize()` method on this variable

Update `./test` to:

- import the variable `storage`
- in the method `test_new_bookmark()`:
  - call `save()` method of `storage`
  - `print("Number of objects: {}".format(len(storage.all())))`
  - if it's a new instance (not from a dictionary representation), add a call to the method `serialize()` on `storage`

```
pulisanbookbuilder:~/Bin$ cat test_new_bookmark_save.py
#!/usr/bin/python3
from models import storage
from models.base_model import BaseModel

all_obj = storage.all()
print("- Because objects -")
for obj in all_obj:
    obj = all_obj[obj]
    print(obj)

print("- Create a new object -")
my_model = BaseModel()
my_model.name = "My_First_Model"
my_model.my_number = 89
my_model.save()
print(my_model)

pulisanbookbuilder:~/Bin$ cat Site.json
[{"__class__": "models.Site.Site", "id": "0f0a0a0a-0000-0000-0000-0000000000ff",
"updated_at": "2017-09-28T02:03:56.252290", "created_at": "2017-09-28T02:03:56.252290",
"_id": "0f0a0a0a-0000-0000-0000-0000000000ff",
"_data": {"id": "0f0a0a0a-0000-0000-0000-0000000000ff",
"updated_at": "2017-09-28T02:03:56.252290",
"created_at": "2017-09-28T02:03:56.252290",
"_id": "0f0a0a0a-0000-0000-0000-0000000000ff",
"_class": "models.Site.Site"}}]
pulisanbookbuilder:~/Bin$
pulisanbookbuilder:~/Bin$ ./test_new_relised_book_save.py
- Because objects -
- Create a new object -
{'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', '_class': 'models.Bookmark.Bookmark',
'_data': {'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'name': 'My_First_Book', 'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'updated_at': '2017-09-28T02:03:56.252290', 'created_at': '2017-09-28T02:03:56.252290', '_class': 'models.Site.Site',
'notes': 'My_First_Note', 'my_number': 89}}
pulisanbookbuilder:~/Bin$
pulisanbookbuilder:~/Bin$ ./test_new_relised_book_save.py
- Because objects -
{'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', '_class': 'models.Bookmark.Bookmark',
'_data': {'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'name': 'My_First_Book', 'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'updated_at': '2017-09-28T02:03:56.252290', 'created_at': '2017-09-28T02:03:56.252290', '_class': 'models.Site.Site',
'notes': 'My_First_Note', 'my_number': 89}}
- Create a new object -
{'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', '_class': 'models.Bookmark.Bookmark',
'_data': {'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'name': 'My_First_Book', 'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'updated_at': '2017-09-28T02:03:56.252290', 'created_at': '2017-09-28T02:03:56.252290', '_class': 'models.Site.Site',
'notes': 'My_First_Note', 'my_number': 89}}
pulisanbookbuilder:~/Bin$
pulisanbookbuilder:~/Bin$ ./test_new_relised_book_save.py
- Because objects -
{'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', '_class': 'models.Bookmark.Bookmark',
'_data': {'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'name': 'My_First_Book', 'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'updated_at': '2017-09-28T02:03:56.252290', 'created_at': '2017-09-28T02:03:56.252290', '_class': 'models.Site.Site',
'notes': 'My_First_Note', 'my_number': 89}}
- Create a new object -
{'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', '_class': 'models.Bookmark.Bookmark',
'_data': {'_id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f', 'name': 'My_First_Book', 'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'updated_at': '2017-09-28T02:03:56.252290', 'created_at': '2017-09-28T02:03:56.252290', '_class': 'models.Site.Site',
'notes': 'My_First_Note', 'my_number': 89}}
pulisanbookbuilder:~/Bin$
```

Repo:

- GitHub repository: `github.com/pulisan`
- File: `models/serialize_storage.py models/engine/__init__.py models/base_model.py models/base_model.py tests/`

Closed | Check your code | Ask for a new correction | Get a sandbox | Qt Review

6. Console 0.0.1

MANDATORY

Score: 100.0% (Checks completed: 100.0%)

Write a program called `hbnb` that contains the entry point of the command interpreter:

- You must use the module `cmd`
- Your command line prompt must be `(hbnb)`
- Your command interpreter should implement:
  - `quit` command to exit the program
  - `help` this command is provided by default by `cmd` but you should keep it updated and documented as you work through tasks
  - an empty line + `ENTER` shouldn't execute anything
- `hbnb` should not be executed when imported

Warning:

You should end your file with:

```
if __name__ == '__main__':
    HBNBCommand().cmdloop()
```

to make your program executable except when imported. Please don't add anything around - the Checker won't like it otherwise

```
pulisanbookbuilder:~/Bin$ ./console.py
(hbnb) help

Documented commands (type help <topic>):
========================================
EOF  help  quit

(hbnb)
(hbnb) help quit
quit Command to exit the program

(hbnb)
(hbnb) quit
pulisanbookbuilder:~/Bin$
```

No artifacts needed

Repo:

- GitHub repository: `github.com/pulisan`
- File: `console.py`

Closed | Check your code | Get a sandbox | Qt Review

7. Console 0.1

MANDATORY

Score: 100.0% (Checks completed: 100.0%)

Update your command interpreter (`console.py`) to have these commands:

- `create`: Creates a new instance of `BaseModel`, saves it (to the JSON file) and prints the `id`. Ex: `$ create BaseModel`
  - If the class name is missing, print `** class name missing **` (ex: `$ create`)
  - If the class name doesn't exist, print `** class name doesn't exist **` (ex: `$ create MyModel`)
- `show`: Prints the string representation of an instance based on the class name and `id`. Ex: `$ show BaseModel 1234-1234-1234`.
  - If the class name is missing, print `** class name missing **` (ex: `$ show`)
  - If the class name doesn't exist, print `** class name doesn't exist **` (ex: `$ show MyModel`)
  - If the instance of the class name doesn't exist for the `id`, print `** no instance found **` (ex: `$ show BaseModel 1234`)
- `destroy`: Deletes an instance based on the class name and `id` (save the change into the JSON file). Ex: `$ destroy BaseModel 1234-1234-1234`.
  - If the class name is missing, print `** class name missing **` (ex: `$ destroy`)
  - If the class name doesn't exist, print `** class name doesn't exist **` (ex: `$ destroy MyModel`)
  - If the `id` is missing, print `** instance id missing **` (ex: `$ destroy BaseModel`)
  - If the instance of the class name doesn't exist for the `id`, print `** no instance found **` (ex: `$ destroy BaseModel 1234`)
- `all`: Prints all string representation of all instances based or not on the class name. Ex: `$ all BaseModel` or `$ all`.
  The printed result must be a list of strings.
  If the class name doesn't exist, print `** class name doesn't exist **` (ex: `$ all MyModel`)
- `update`: Updates an instance based on the class name and `id` by adding or updating attribute (save the change into the JSON file).
  Usage: `update <class name> <id> <attribute name> "<attribute value>"`
  - Only one attribute can be updated at the time
  - You can assume the attribute name is valid (exists for this model)
  - The empty string is a valid value (example: `name=""`)
  - If the class name is missing, print `** class name missing **` (ex: `$ update`)
  - If the class name doesn't exist, print `** class name doesn't exist **` (ex: `$ update MyModel`)
  - If the `id` is missing, print `** instance id missing **` (ex: `$ update BaseModel`)
  - If the instance of the class name doesn't exist for the `id`, print `** no instance found **` (ex: `$ update BaseModel 1234`)
  - If the attribute name is missing, print `** attribute name missing **` (ex: `$ update BaseModel 1234-1234-1234`)
  - If the value for the attribute name doesn't exist, print `** value missing **` (ex: `$ update BaseModel 1234-1234-1234 name`)
  - All other arguments should be not used (Ex: `$ update BaseModel 1234-1234-1234 name="Holberton" price="3.14" rating=5`). All other arguments should be `cast` to the updated. You can assume they won't be passed in the `update` command
  - Only "normal" arguments can be updated: string, integer and float. You can assume nobody will try to update list of ids or dictionary

Let's add some rules:

- You can assume arguments are always in the right order
- Each arguments are separated by a space
- A string argument with a space must be between double quote
- The word separating from the class name to the command is the last one

```
pulisanbookbuilder:~/Bin$ ./console.py
(hbnb) all BaseModel
[[BaseModel] (0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f)
{'updated_at': '2017-09-28T02:03:56.252290',
'id': '0a4b4c11-ce7b-4b2d-a2f6-d7ac7b7b4f7f',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'my_number': 89}, [BaseModel] (0f0a0a0a-0000-0000-0000-0000000000ff)
{'updated_at': '2017-09-28T02:03:56.252290',
'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'my_number': 89}]
(hbnb) destroy
** class name missing **
(hbnb) update BaseModel 0f0a0a0a-0000-0000-0000-0000000000ff first_name "Bob"
(hbnb) show BaseModel 0f0a0a0a-0000-0000-0000-0000000000ff
[BaseModel] (0f0a0a0a-0000-0000-0000-0000000000ff)
{'updated_at': '2017-09-28T02:03:56.252290',
'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
(hbnb) show BaseModel 0f0a0a0a-0000-0000-0000-0000000000ff
[BaseModel] (0f0a0a0a-0000-0000-0000-0000000000ff)
{'updated_at': '2017-09-28T02:03:56.252290',
'id': '0f0a0a0a-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
(hbnb) destroy BaseModel 0f0a0a0a-0000-0000-0000-0000000000ff
(hbnb) show BaseModel 0f0a0a0a-0000-0000-0000-0000000000ff
** no instance found **
(hbnb)
(hbnb)
pulisanbookbuilder:~/Bin$
```

No artifacts needed

Repo:

- GitHub repository: `github.com/pulisan`
- File: `console.py`

Closed | Check your code | Get a sandbox | Qt Review

8. First User

MANDATORY

Score: 100.0% (Checks completed: 100.0%)

Write a class `User` that inherits from `BaseModel`:

- `models/user.py`
- Public class attributes:
  - `email`: string - empty string
  - `password`: string - empty string
  - `first_name`: string - empty string
  - `last_name`: string - empty string

Update `console.py` to manage correctly serialization and deserialization of `User`.

Update your command interpreter (`console.py`) to allow `show`, `destroy`, `update`, `all`, `count` and `with` from `User`.

```
pulisanbookbuilder:~/Bin$ cat test_new_relised_user.py
#!/usr/bin/python3
from models import storage
from models.base_model import BaseModel
from models.user import User

all_obj = storage.all()
print("- Because objects -")
for obj in all_obj:
    obj = all_obj[obj]
    print(obj)

print("- Create a new User -")
my_user = User()
my_user.first_name = "Bob"
my_user.last_name = "Pul"
my_user.email = "bob@holberton.com"
my_user.password = "pass"
my_user.save()
print(my_user)

print("- Create a new User 2 -")
my_user2 = User()
my_user2.first_name = "Kate"
my_user2.last_name = "Pul"
my_user2.email = "kate@holberton.com"
my_user2.password = "pass"
my_user2.save()
print(my_user2)

pulisanbookbuilder:~/Bin$ ./test_new_relised_user.py
- Because objects -
[BaseModel] (1f4a2a25-a4ec-41ee-9256-f3b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '1f4a2a25-a4ec-41ee-9256-f3b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (2b4c2b20-a4e5-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '2b4c2b20-a4e5-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (3d7b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '3d7b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (4e9b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '4e9b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
- Create a new User -
[User] (5f0f0f0f-0000-0000-0000-0000000000ff) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '5f0f0f0f-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'password': 'pass',
'email': 'bob@holberton.com',
'first_name': 'Bob',
'last_name': 'Pul'}
pulisanbookbuilder:~/Bin$
pulisanbookbuilder:~/Bin$ ./test_new_relised_user.py
- Because objects -
[BaseModel] (1f4a2a25-a4ec-41ee-9256-f3b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '1f4a2a25-a4ec-41ee-9256-f3b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (2b4c2b20-a4e5-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '2b4c2b20-a4e5-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (3d7b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '3d7b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (4e9b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '4e9b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[User] (5f0f0f0f-0000-0000-0000-0000000000ff) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '5f0f0f0f-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'password': 'pass',
'email': 'bob@holberton.com',
'first_name': 'Bob',
'last_name': 'Pul'}
- Create a new User -
[User] (6a0a0a0a-0000-0000-0000-0000000000ff) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '6a0a0a0a-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'password': 'pass',
'email': 'kate@holberton.com',
'first_name': 'Kate',
'last_name': 'Pul'}
pulisanbookbuilder:~/Bin$
pulisanbookbuilder:~/Bin$ ./test_new_relised_user.py
- Because objects -
[BaseModel] (1f4a2a25-a4ec-41ee-9256-f3b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '1f4a2a25-a4ec-41ee-9256-f3b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (2b4c2b20-a4e5-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '2b4c2b20-a4e5-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (3d7b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '3d7b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[BaseModel] (4e9b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '4e9b4d20-a1ec-4ee6-8376-f7b7b6d5f2bc',
'created_at': '2017-09-28T02:03:56.252290',
'name': 'My_First_Model',
'first_name': 'Bob',
'my_number': 89}
[User] (5f0f0f0f-0000-0000-0000-0000000000ff) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '5f0f0f0f-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'password': 'pass',
'email': 'bob@holberton.com',
'first_name': 'Bob',
'last_name': 'Pul'}
[User] (6a0a0a0a-0000-0000-0000-0000000000ff) {'updated_at': '2017-09-28T02:03:56.252290',
'id': '6a0a0a0a-0000-0000-0000-0000000000ff',
'created_at': '2017-09-28T02:03:56.252290',
'password': 'pass',
'email': 'kate@holberton.com',
'first_name': 'Kate',
'last_name': 'Pul'}
pulisanbookbuilder:~/Bin$
```

No artifacts needed for the console

Repo:

- GitHub repository: `github.com/pulisan`
- File: `models/user.py models/base_model.py console.py tests/`

Closed | Check your code | Get a sandbox | Qt Review

9. More classes!

Hardcoded

Score: 100.0% (Checks completed: 100.0%)

Write all those classes that inherit from `BaseModel`:

- `State` (`BaseModel`):
  - Public class attributes:
    - `name`: string - empty string
- `City` (`BaseModel`):
  - Public class attributes:
    - `state_id`: string - empty string: it will be the `State.id`
    - `name`: string - empty string
- `Amenity` (`BaseModel`):
  - Public class attributes:
    - `name`: string - empty string
- `Place` (`BaseModel`):
  - Public class attributes:
    - `city_id`: string - empty string: it will be the `City.id`
    - `user_id`: string - empty string: it will be the `User.id`
    - `name`: string - empty string
    - `description`: string - empty string
    - `number_rooms`: integer - 0
    - `number_bathrooms`: integer - 0
    - `max_guest`: integer - 0
    - `price_by_night`: integer - 0
    - `latitude`: float - 0.0
    - `longitude`: float - 0.0
    - `amenity_ids`: list of string - empty list: it will be the list of `Amenity.id` later
- `Review` (`BaseModel`):
  - Public class attributes:
    - `place_id`: string - empty string: it will be the `Place.id`
    - `user_id`: string - empty string: it will be the `User.id`
    - `text`: string - empty string

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `models/state.py`, `models/city.py`, `models/amenity.py`, `models/place.py`, `models/review.py`, `models/__init__.py`

Close | Check your code | Ask for a sandbox | QA Review

10. Console 1.0

Hardcoded

Score: 100.0% (Checks completed: 100.0%)

Update `console.py` to manage correctly serialization and deserialization of all our new classes: `State`, `City`, `Amenity`, `Place` and `Review`.

Update your command interpreter (`console.py`) to allow those actions: `show`, `create`, `destroy`, `update` and `all` with all classes created previously.

Enjoy your first console!

No unittests needed for the console

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `console.py`, `models/__init__.py`, `models/base_model.py`

Close | Check your code | Ask for a sandbox | QA Review

11. All instances by class name

Mandatory

Score: 100.0% (Checks completed: 100.0%)

Update your command interpreter (`console.py`) to retrieve all instances of a class by using: `<class name>.all()`.

```
guillaumebouhours~/bin$bnb -c tests/test_models/test_base_model.txt
(hbnb) Base.all()
[[Base] (b7d1a1ff-a3c2-41da-8a7f-6a4f5b05f0d8) {'first_name': 'Betty', 'last_name': 'Bar', 'created_at': date
time.datetime(2017, 9, 28, 21, 12, 35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 35, 481
092), 'id': '7d8d1a1ff-a3c2-41da-8a7f-6a4f5b05f0d8', 'my_number': 89}, [Base] (b7d1a1ff-a3c2-41da-8a7f-6a4f
5b05f0d8) {'first_name': 'Betty', 'last_name': 'Bar', 'created_at': datetime.datetime(2017, 9, 28, 21, 12,
35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 35, 480776), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 35, 48919
1), 'password': '5f9b61180f9e9e9e7d8f5f5b9e8d7', 'email': 'alexbouhours.com', 'id': '3f82181c-a75-4ed4-b27
-7ad1d7c7a7c1', 'my_number': 89}]
(hbnb)
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.count()
1
(hbnb)

No unittests needed

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `console.py`

Close | Check your code | Ask for a sandbox | QA Review
```

12. Count instances

Mandatory

Score: 100.0% (Checks completed: 100.0%)

Update your command interpreter (`console.py`) to retrieve the number of instances of a class: `<class name>.count()`.

```
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.count()
1
(hbnb)

No unittests needed

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `console.py`

Close | Check your code | Ask for a sandbox | QA Review
```

13. Show

Mandatory

Score: 100.0% (Checks completed: 100.0%)

Update your command interpreter (`console.py`) to retrieve an instance based on its ID: `<class name>.show(<id>)`.

Errors management must be the same as previously.

```
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.show("b7d1a1ff-a3c2-41da-8a7f-6a4f5b05f0d8")
[Base] (b7d1a1ff-a3c2-41da-8a7f-6a4f5b05f0d8) {'first_name': 'Betty', 'last_name': 'Bar', 'created_at': date
time.datetime(2017, 9, 28, 21, 12, 35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 35, 48919
1), 'password': '5f9b61180f9e9e9e7d8f5f5b9e8d7', 'email': 'alexbouhours.com', 'id': '3f82181c-a75-4ed4-b27
-7ad1d7c7a7c1', 'my_number': 89}
(hbnb)
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.show("b7d1a1ff-a3c2-41da-8a7f-6a4f5b05f0d8")
[Base] (b7d1a1ff-a3c2-41da-8a7f-6a4f5b05f0d8) {'first_name': 'Betty', 'last_name': 'Bar', 'created_at': date
time.datetime(2017, 9, 28, 21, 12, 35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 35, 48919
1), 'password': '5f9b61180f9e9e9e7d8f5f5b9e8d7', 'email': 'alexbouhours.com', 'id': '3f82181c-a75-4ed4-b27
-7ad1d7c7a7c1', 'my_number': 89}
(hbnb)
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.show("")
** no instance found **
(hbnb)

No unittests needed

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `console.py`

Close | Check your code | Ask for a sandbox | QA Review
```

14. Destroy

Mandatory

Score: 100.0% (Checks completed: 100.0%)

Update your command interpreter (`console.py`) to destroy an instance based on its ID: `<class name>.destroy(<id>)`.

Errors management must be the same as previously.

```
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.count()
1
(hbnb) Base.destroy("b7d1a1ff-a3c2-41da-8a7f-6a4f5b05f0d8")
(hbnb) Base.count()
0
(hbnb) Base.destroy("Bar")
** no instance found **
(hbnb)

No unittests needed

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `console.py`

Close | Check your code | Ask for a sandbox | QA Review
```

15. Update

Mandatory

Score: 100.0% (Checks completed: 100.0%)

Update your command interpreter (`console.py`) to update an instance based on its ID: `<class name>.update(<id>, <attribute name>, <attribute value>)`.

Errors management must be the same as previously.

```
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.show("3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1")
[Base] (3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1) {'first_name': 'Betty', 'last_name': 'Bar', 'created_at': date
time.datetime(2017, 9, 28, 21, 12, 35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 35, 48919
1), 'password': '5f9b61180f9e9e9e7d8f5f5b9e8d7', 'email': 'alexbouhours.com', 'id': '3f82181c-a75-4ed4-b27
-7ad1d7c7a7c1', 'my_number': 89}
(hbnb)
(hbnb) Base.update("3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1", "age", 89)
(hbnb)
(hbnb) Base.show("3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1")
[Base] (3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1) {'age': 89, 'first_name': 'John', 'last_name': 'Bar', 'created_
at': datetime.datetime(2017, 9, 28, 21, 12, 35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12,
35, 480776), 'password': '5f9b61180f9e9e9e7d8f5f5b9e8d7', 'email': 'alexbouhours.com', 'id': '3f82181c-a75
-7ad1d7c7a7c1', 'my_number': 89}
(hbnb)

No unittests needed

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `console.py`

Close | Check your code | Ask for a sandbox | QA Review
```

16. Update from dictionary

Mandatory

Score: 70.83% (Checks completed: 100.0%)

Update your command interpreter (`console.py`) to update an instance based on its ID with a dictionary: `<class name>.update(<id>, <dictionary representation>)`.

Errors management must be the same as previously.

```
guillaumebouhours~/bin$bnb -c console.py
(hbnb) Base.show("3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1")
[Base] (3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1) {'age': 89, 'first_name': 'John', 'last_name': 'Bar', 'created_a
t': datetime.datetime(2017, 9, 28, 21, 12, 35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12,
35, 480776), 'password': '5f9b61180f9e9e9e7d8f5f5b9e8d7', 'email': 'alexbouhours.com', 'id': '3f82181c-a75
-7ad1d7c7a7c1', 'my_number': 89}
(hbnb)
(hbnb) Base.update("3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1", {'first_name': "John", "age": 89})
(hbnb)
(hbnb) Base.show("3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1")
[Base] (3f82181c-a75-4ed4-b27c-7ad1d7c7a7c1) {'age': 89, 'first_name': 'John', 'last_name': 'Bar', 'created_
at': datetime.datetime(2017, 9, 28, 21, 12, 35, 481092), 'updated_at': datetime.datetime(2017, 9, 28, 21, 12,
35, 480776), 'password': '5f9b61180f9e9e9e7d8f5f5b9e8d7', 'email': 'alexbouhours.com', 'id': '3f82181c-a75
-7ad1d7c7a7c1', 'my_number': 89}
(hbnb)

No unittests needed

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `console.py`

Close | Check your code | Ask for a sandbox | QA Review
```

17. Unittests for the Console!

Optional

Score: 0.21% (Checks completed: 15.69%)

Write all unittests for `console.py`, all features!

Fix hard-hing the console, you should "interrupt" STDOUT of it, we highly recommend you to use:

```
with patch("sys.stdout", mock_stdout):
    HBNBCommand().onecmd("help quit")
```

Otherwise, you will have to write the console by replicating `cmdloop` for `onecmd()`.

Well done on completing this project! Let the world hear about this milestone achieved.

Click here to tweet!

Repo:

- GitHub repository: `ALX_BnB_clone`
- File: `tests/test_console.py`

Close | Check your code | Ask for a new sandbox | Ask for a sandbox | QA Review

Ready for a new review

Copyright © 2019 ALX, All rights reserved.