0x00. AirBnB clone - The console

alx intranet.alxswe.com/projects/263

Group projectPythonOOP

By: Guillaume
Weight: 5
Project to be done in teams of 2 people (your team: Jonah Asante, Jason Jasiel Quist)
Project will start Jul 10, 2023 6:00 AM, must end by Jul 17, 2023 6:00 AM
Checker will be released at Jul 15, 2023 12:00 PM
Manual QA review must be done (request it when you are done with the project)
An auto review will be launched at the deadline

Concepts

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

- Python packages
- AirBnB clone

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 what you build during this project with all other 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 (`User`, `State`, `City`, `Place`…) 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
- unittest module
- args/kwargs
- Python test cheatsheet
- cmd module wiki page
- python unittest

Learning Objectives

At the end of this 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.
- You will not be able to meet the objectives of this or any following project by 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 program.

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 `2.8.*`)
- All your files must be executable
- The length of your files will be tested using `wc`
- All your modules should have a documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)
- All your classes should have a documentation (`python3 -c 'print(__import__("my_module").MyClass.__doc__)'`)

- All your functions (inside and outside a class) should have a documentation (`python3 -c 'print(__import__("my_module").my_function.__doc__)'` and `python3 -c 'print(__import__("my_module").MyClass.my_function.__doc__)'`)
- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module, class or method (the length of it will be verified)

Python Unit Tests

- Allowed editors: `vi`, `vim`, `emacs`
- All your files should end with a new line
- All your test files should be inside a folder `tests`
- You have to use the unittest module
- All your test files should be python files (extension: `.py`)
- All your test files and folders should start by `test_`
- Your file organization in the tests folder should be the same as your project
- e.g., For `models/base_model.py`, unit tests must be in: `tests/test_models/test_base_model.py`
- e.g., For `models/user.py`, unit tests must be in: `tests/test_models/test_user.py`
- All your tests should be executed by using this command: `python3 -m unittest discover tests`
- You can also test file by file by using this command: `python3 -m unittest tests/test_models/test_base_model.py`
- All your modules should have a documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)
- All your classes should have a documentation (`python3 -c 'print(__import__("my_module").MyClass.__doc__)'`)
- All your functions (inside and outside a class) should have a documentation (`python3 -c 'print(__import__("my_module").my_function.__doc__)'` and `python3 -c 'print(__import__("my_module").MyClass.my_function.__doc__)'`)
- We strongly encourage you to work together on test cases, so that you don't miss any edge case

GitHub

There should be one project repository per group. 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)
(hbnb) quit
$
```

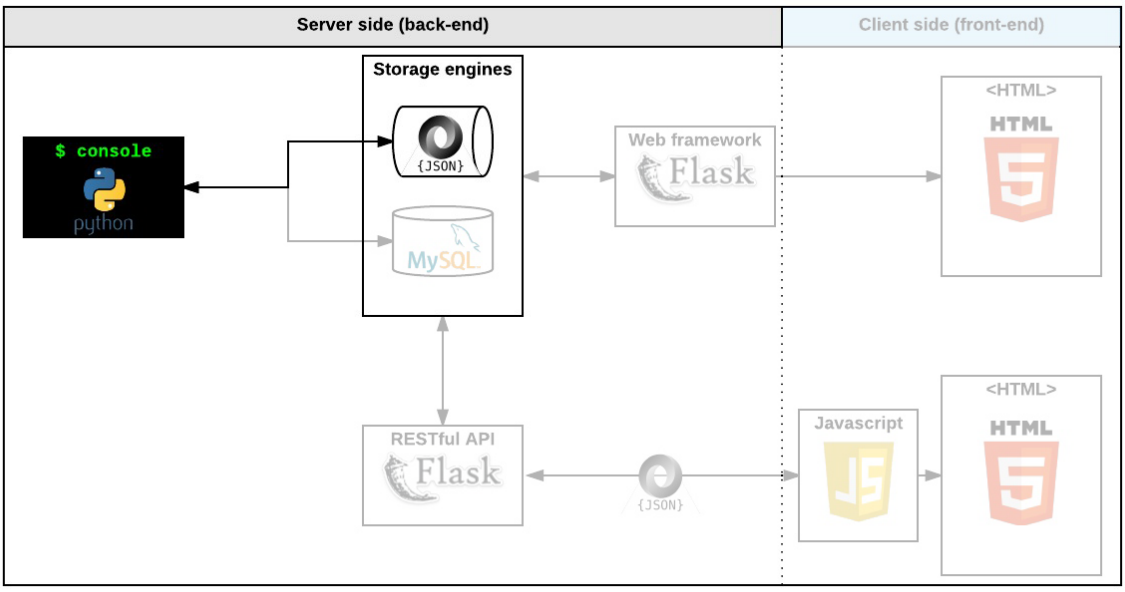
But also in non-interactive mode: (like the Shell project in C)

```
$ 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: `$ echo "python3 -m unittest discover tests" | bash`



Video library(8 total)

HBNB project overview

HBNB - the console

Python: Unique Identifier

Python: Unittests

Python: BaseModel and inheritance

Code consistency

Python: Modules and Packages

HBNB - storage abstraction

Tasks

0. README, AUTHORS

mandatory

- Write a `README.md`:
  - description of the project
  - description of the command interpreter:
    - how to start it
    - how to use it
    - examples
- You should have an `AUTHORS` file at the root of your repository, listing all individuals having contributed content to the repository. For format, reference Docker's AUTHORS page
- You should use branches and pull requests on GitHub - it will help you as team to organize your work

Repo:

- GitHub repository: `AirBnB_clone`
- File: `README.md, AUTHORS`

1. Be pycodestyle compliant!

mandatory

Write beautiful code that passes the pycodestyle checks.

Repo:

GitHub repository: `AirBnB_clone`

2. Unittests

mandatory

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

```
guillaume@ubuntu:~/AirBnB$ python3 -m unittest discover tests
.....................................................................................
..
.....................................................................................
..
......................
----------------------------------------------------------------------
Ran 189 tests in 13.135s

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

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:~/AirBnB$ echo "python3 -m unittest discover tests" | bash
.....................................................................................
..
.....................................................................................
..
......................
----------------------------------------------------------------------
Ran 189 tests in 13.135s

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

Repo:

- GitHub repository: `AirBnB_clone`
- File: `tests/`

3. BaseModel

mandatory

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

- `models/base_model.py`
- Public instance attributes:
  - `id`: string - assign with an `uuid` when an instance is created:
    - you can use `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`)
      - you can use `isoformat()` of `datetime` 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:~/AirBnB$ 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)
my_model.save()
print(my_model)
my_model_json = my_model.to_dict()
print(my_model_json)
print("JSON of my_model:")
for key in my_model_json.keys():
    print("\t{}: ({}) - {}".format(key, type(my_model_json[key]),
my_model_json[key]))

guillaume@ubuntu:~/AirBnB$ ./test_base_model.py
[BaseModel] (b6a6e15c-c67d-4312-9a75-9d084935e579) {'my_number': 89, 'name': 'My
First Model', 'updated_at': datetime.datetime(2017, 9, 28, 21, 5, 54, 119434),
'id': 'b6a6e15c-c67d-4312-9a75-9d084935e579', 'created_at':
datetime.datetime(2017, 9, 28, 21, 5, 54, 119427)}
[BaseModel] (b6a6e15c-c67d-4312-9a75-9d084935e579) {'my_number': 89, 'name': 'My
First Model', 'updated_at': datetime.datetime(2017, 9, 28, 21, 5, 54, 119572),
'id': 'b6a6e15c-c67d-4312-9a75-9d084935e579', 'created_at':
datetime.datetime(2017, 9, 28, 21, 5, 54, 119427)}
{'my_number': 89, 'name': 'My First Model', '__class__': 'BaseModel',
'updated_at': '2017-09-28T21:05:54.119572', 'id': 'b6a6e15c-c67d-4312-9a75-
9d084935e579', 'created_at': '2017-09-28T21:05:54.119427'}
JSON of my_model:
    my_number: (<class 'int'>) - 89
    name: (<class 'str'>) - My First Model
    __class__: (<class 'str'>) - BaseModel
    updated_at: (<class 'str'>) - 2017-09-28T21:05:54.119572
    id: (<class 'str'>) - b6a6e15c-c67d-4312-9a75-9d084935e579
    created_at: (<class 'str'>) - 2017-09-28T21:05:54.119427

guillaume@ubuntu:~/AirBnB$
```

Repo:

- GitHub repository: `AirBnB_clone`
- File: `models/base_model.py, models/__init__.py, tests/`

4. Create BaseModel from dictionary

mandatory

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.

```
<class 'BaseModel'> -> to_dict() -> <class 'dict'> -> <class 'BaseModel'>
```

Update `models/base_model.py`:

- `__init__(self, *args, **kwargs)`:
  - you will use `*args, **kwargs` arguments for the constructor of a `BaseModel`. (more information inside the **AirBnB clone** concept page)
  - `*args` won't be used
  - 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, below)
    - 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. Tip: you know the string format of these datetime
  - otherwise:
    create `id` and `created_at` as you did previously (new instance)

```
guillaume@ubuntu:~/AirBnB$ cat test_base_model_dict.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("\t{}: ({}) - {}".format(key, type(my_model_json[key]),
my_model_json[key]))

print("--")
my_new_model = BaseModel(**my_model_json)
print(my_new_model.id)
print(my_new_model)
print(type(my_new_model.created_at))

print("--")
print(my_model is my_new_model)

guillaume@ubuntu:~/AirBnB$ ./test_base_model_dict.py
56d43177-cc5f-4d6c-a0c1-e167f8c27337
[BaseModel] (56d43177-cc5f-4d6c-a0c1-e167f8c27337) {'id': '56d43177-cc5f-4d6c-
a0c1-e167f8c27337', 'created_at': datetime.datetime(2017, 9, 28, 21, 3, 54,
52298), 'my_number': 89, 'updated_at': datetime.datetime(2017, 9, 28, 21, 3, 54,
52302), 'name': 'My_First_Model'}
<class 'datetime.datetime'>
--
{'id': '56d43177-cc5f-4d6c-a0c1-e167f8c27337', 'created_at': '2017-09-
28T21:03:54.052298', '__class__': 'BaseModel', 'my_number': 89, 'updated_at':
'2017-09-28T21:03:54.052302', 'name': 'My_First_Model'}
JSON of my_model:
    id: (<class 'str'>) - 56d43177-cc5f-4d6c-a0c1-e167f8c27337
    created_at: (<class 'str'>) - 2017-09-28T21:03:54.052298
    __class__: (<class 'str'>) - BaseModel
    my_number: (<class 'int'>) - 89
    updated_at: (<class 'str'>) - 2017-09-28T21:03:54.052302
    name: (<class 'str'>) - My_First_Model
--
56d43177-cc5f-4d6c-a0c1-e167f8c27337
[BaseModel] (56d43177-cc5f-4d6c-a0c1-e167f8c27337) {'id': '56d43177-cc5f-4d6c-
a0c1-e167f8c27337', 'created_at': datetime.datetime(2017, 9, 28, 21, 3, 54,
52298), 'my_number': 89, 'updated_at': datetime.datetime(2017, 9, 28, 21, 3, 54,
52302), 'name': 'My_First_Model'}
<class 'datetime.datetime'>
--
False
guillaume@ubuntu:~/AirBnB$
```

Repo:

- GitHub repository: `AirBnB_clone`
- File: `models/base_model.py, tests/`

5. Store first object

mandatory

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

```
<class 'BaseModel'> -> to_dict() -> <class 'dict'> -> <class 'BaseModel'>
```

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 a file.

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

- Python doesn't know how to convert a string to a dictionary (easily)
- It's not human readable
- Using this file with 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 all programming languages have a JSON reader and writer.

Now the flow of serialization-deserialization will be:

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

Magic right?

Terms:

- **simple Python data structure**: Dictionaries, arrays, number and string. ex: `{ '12': { 'numbers': [1, 2, 3], 'name': "John" } }`
- **JSON string representation**: String representing a simple data structure in JSON format. ex: `'{ "12": { "numbers": [1, 2, 3], "name": "John" } }'`

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

- `models/engine/file_storage.py`

- Private class attributes:
  - `__file_path`: string - path to the JSON file (ex: `file.json`)
  - `__objects`: dictionary - empty but will store all objects by `<class name>.id` (ex: to store a `BaseModel` object with `id=12121212`, the key will be `BaseModel.12121212`)
- Public instance methods:
  - `all(self)`: returns the dictionary `__objects`
  - `new(self, obj)`: sets in `__objects` the `obj` with key `<obj class name>.id`
  - `save(self)`: serializes `__objects` to the JSON file (path: `__file_path`)
  - `reload(self)`: deserializes the JSON file to `__objects` (only if the JSON file (`__file_path`) exists ; otherwise, do nothing. If the file doesn't exist, no exception should be raised)

Update `models/__init__.py`: to create a unique `FileStorage` instance for your application

- import `file_storage.py`
- create the variable `storage`, an instance of `FileStorage`
- call `reload()` method on this variable

Update `models/base_model.py`: to link your `BaseModel` to `FileStorage` by using the variable `storage`

- import the variable `storage`
- in the method `save(self)`:
  call `save(self)` method of `storage`
- `__init__(self, *args, **kwargs)`:
  if it's a new instance (not from a dictionary representation), add a call to the method `new(self)` on `storage`

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

all_objs = storage.all()
print("-- Reloaded objects --")
for obj_id in all_objs.keys():
    obj = all_objs[obj_id]
    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)

guillaume@ubuntu:~/AirBnB$ cat file.json
cat: file.json: No such file or directory
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ ./test_save_reload_base_model.py
-- Reloaded objects --
-- Create a new object --
[BaseModel] (ee49c413-023a-4b49-bd28-f2936c95460d) {'my_number': 89,
'updated_at': datetime.datetime(2017, 9, 28, 21, 7, 25, 47381), 'created_at':
datetime.datetime(2017, 9, 28, 21, 7, 25, 47372), 'name': 'My_First_Model', 'id':
'ee49c413-023a-4b49-bd28-f2936c95460d'}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ cat file.json ; echo ""
{"BaseModel.ee49c413-023a-4b49-bd28-f2936c95460d": {"my_number": 89, "__class__":
"BaseModel", "updated_at": "2017-09-28T21:07:25.047381", "created_at": "2017-09-
28T21:07:25.047372", "name": "My_First_Model", "id": "ee49c413-023a-4b49-bd28-
f2936c95460d"}}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ ./test_save_reload_base_model.py
-- Reloaded objects --
[BaseModel] (ee49c413-023a-4b49-bd28-f2936c95460d) {'name': 'My_First_Model',
'id': 'ee49c413-023a-4b49-bd28-f2936c95460d', 'updated_at':
datetime.datetime(2017, 9, 28, 21, 7, 25, 47381), 'my_number': 89, 'created_at':
datetime.datetime(2017, 9, 28, 21, 7, 25, 47372)}
-- Create a new object --
[BaseModel] (080cce84-c574-4230-b82a-9acb74ad5e8c) {'name': 'My_First_Model',
'id': '080cce84-c574-4230-b82a-9acb74ad5e8c', 'updated_at':
datetime.datetime(2017, 9, 28, 21, 7, 51, 973308), 'my_number': 89, 'created_at':
datetime.datetime(2017, 9, 28, 21, 7, 51, 973301)}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ ./test_save_reload_base_model.py
-- Reloaded objects --
[BaseModel] (080cce84-c574-4230-b82a-9acb74ad5e8c) {'id': '080cce84-c574-4230-
b82a-9acb74ad5e8c', 'updated_at': datetime.datetime(2017, 9, 28, 21, 7, 51,
973308), 'created_at': datetime.datetime(2017, 9, 28, 21, 7, 51, 973301), 'name':
'My_First_Model', 'my_number': 89}
[BaseModel] (ee49c413-023a-4b49-bd28-f2936c95460d) {'id': 'ee49c413-023a-4b49-
bd28-f2936c95460d', 'updated_at': datetime.datetime(2017, 9, 28, 21, 7, 25,
47381), 'created_at': datetime.datetime(2017, 9, 28, 21, 7, 25, 47372), 'name':
```

```
'My_First_Model', 'my_number': 89}
-- Create a new object --
[BaseModel] (e79e744a-55d4-45a3-b74a-ca5fae74e0e2) {'id': 'e79e744a-55d4-45a3-
b74a-ca5fae74e0e2', 'updated_at': datetime.datetime(2017, 9, 28, 21, 8, 6,
151750), 'created_at': datetime.datetime(2017, 9, 28, 21, 8, 6, 151711), 'name':
'My_First_Model', 'my_number': 89}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ cat file.json ; echo ""
{"BaseModel.e79e744a-55d4-45a3-b74a-ca5fae74e0e2": {"__class__": "BaseModel",
"id": "e79e744a-55d4-45a3-b74a-ca5fae74e0e2", "updated_at": "2017-09-
28T21:08:06.151750", "created_at": "2017-09-28T21:08:06.151711", "name":
"My_First_Model", "my_number": 89}, "BaseModel.080cce84-c574-4230-b82a-
9acb74ad5e8c": {"__class__": "BaseModel", "id": "080cce84-c574-4230-b82a-
9acb74ad5e8c", "updated_at": "2017-09-28T21:07:51.973308", "created_at": "2017-
09-28T21:07:51.973301", "name": "My_First_Model", "my_number": 89},
"BaseModel.ee49c413-023a-4b49-bd28-f2936c95460d": {"__class__": "BaseModel",
"id": "ee49c413-023a-4b49-bd28-f2936c95460d", "updated_at": "2017-09-
28T21:07:25.047381", "created_at": "2017-09-28T21:07:25.047372", "name":
"My_First_Model", "my_number": 89}}
guillaume@ubuntu:~/AirBnB$
```

Repo:

- GitHub repository: `AirBnB_clone`
- File: `models/engine/file_storage.py, models/engine/__init__.py, models/__init__.py, models/base_model.py, tests/`

6. Console 0.0.1

mandatory

Write a program called `console.py` that contains the entry point of the command interpreter:

- You must use the module `cmd`
- Your class definition must be: `class HBNBCommand(cmd.Cmd):`
- Your command interpreter should implement:
  - `quit` and `EOF` to exit the program
  - `help` (this action is provided by default by `cmd` but you should keep it updated and documented as you work through tasks)
  - a custom prompt: `(hbnb)`
  - an empty line + `ENTER` shouldn't execute anything
- Your code 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

```
guillaume@ubuntu:~/AirBnB$ ./console.py
(hbnb) help

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

(hbnb)
(hbnb) help quit
Quit command to exit the program

(hbnb)
(hbnb)
(hbnb) quit
guillaume@ubuntu:~/AirBnB$
```

No unittests needed

Repo:

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

7. Console 0.1

mandatory

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 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 doesn't exist **` (ex: `$ show MyModel`)
  - If the `id` is missing, print `** instance id missing **` (ex: `$ show BaseModel`)
  - If the instance of the class name doesn't exist for the `id`, print `** no instance found **` (ex: `$ show BaseModel 121212`)
- `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 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 121212`)
- `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 (like the example below)
  - If the class name doesn't exist, print `** class 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). Ex: `$ update BaseModel 1234-1234-1234 email "aibnb@mail.com"`.
    - 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 attribute value must be casted to the attribute type
    - If the class name is missing, print `** class name missing **` (ex: `$ update`)
    - If the class name doesn't exist, print `** class 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 121212`)
    - If the attribute name is missing, print `** attribute name missing **` (ex: `$ update BaseModel existing-id`)
    - If the value for the attribute name doesn't exist, print `** value missing **` (ex: `$ update BaseModel existing-id first_name`)
    - All other arguments should not be used (Ex: `$ update BaseModel 1234-1234-1234 email "aibnb@mail.com" first_name "Betty"` = `$ update BaseModel 1234-1234-1234 email "aibnb@mail.com"`)
    - `id`, `created_at` and `updated_at` cant' be updated. You can assume they won't be passed in the `update` command
    - Only "simple" arguments can be updated: string, integer and float. You can assume nobody will try to update list of ids or datetime

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 error management starts from the first argument to the last one

```
guillaume@ubuntu:~/AirBnB$ ./console.py
(hbnb) all MyModel
** class doesn't exist **
(hbnb) show BaseModel
** instance id missing **
(hbnb) show BaseModel My_First_Model
** no instance found **
(hbnb) create BaseModel
49faff9a-6318-451f-87b6-910505c55907
(hbnb) all BaseModel
["[BaseModel] (49faff9a-6318-451f-87b6-910505c55907) {'created_at':
datetime.datetime(2017, 10, 2, 3, 10, 25, 903293), 'id': '49faff9a-6318-451f-
87b6-910505c55907', 'updated_at': datetime.datetime(2017, 10, 2, 3, 10, 25,
903300)}"]
(hbnb) show BaseModel 49faff9a-6318-451f-87b6-910505c55907
[BaseModel] (49faff9a-6318-451f-87b6-910505c55907) {'created_at':
datetime.datetime(2017, 10, 2, 3, 10, 25, 903293), 'id': '49faff9a-6318-451f-
87b6-910505c55907', 'updated_at': datetime.datetime(2017, 10, 2, 3, 10, 25,
903300)}
(hbnb) destroy
** class name missing **
(hbnb) update BaseModel 49faff9a-6318-451f-87b6-910505c55907 first_name "Betty"
(hbnb) show BaseModel 49faff9a-6318-451f-87b6-910505c55907
[BaseModel] (49faff9a-6318-451f-87b6-910505c55907) {'first_name': 'Betty', 'id':
'49faff9a-6318-451f-87b6-910505c55907', 'created_at': datetime.datetime(2017, 10,
2, 3, 10, 25, 903293), 'updated_at': datetime.datetime(2017, 10, 2, 3, 11, 3,
49401)}
(hbnb) create BaseModel
2dd6ef5c-467c-4f82-9521-a772ea7d84e9
(hbnb) all BaseModel
["[BaseModel] (2dd6ef5c-467c-4f82-9521-a772ea7d84e9) {'id': '2dd6ef5c-467c-4f82-
9521-a772ea7d84e9', 'created_at': datetime.datetime(2017, 10, 2, 3, 11, 23,
639717), 'updated_at': datetime.datetime(2017, 10, 2, 3, 11, 23, 639724)}", "
[BaseModel] (49faff9a-6318-451f-87b6-910505c55907) {'first_name': 'Betty', 'id':
'49faff9a-6318-451f-87b6-910505c55907', 'created_at': datetime.datetime(2017, 10,
2, 3, 10, 25, 903293), 'updated_at': datetime.datetime(2017, 10, 2, 3, 11, 3,
49401)}"]
(hbnb) destroy BaseModel 49faff9a-6318-451f-87b6-910505c55907
(hbnb) show BaseModel 49faff9a-6318-451f-87b6-910505c55907
** no instance found **
(hbnb)
```

No unittests needed

Repo:

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

8. First User

mandatory

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 `FileStorage` to manage correctly serialization and deserialization of `User`.

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

```
guillaume@ubuntu:~/AirBnB$ cat test_save_reload_user.py
#!/usr/bin/python3
from models import storage
from models.base_model import BaseModel
from models.user import User

all_objs = storage.all()
print("-- Reloaded objects --")
for obj_id in all_objs.keys():
    obj = all_objs[obj_id]
    print(obj)

print("-- Create a new User --")
my_user = User()
my_user.first_name = "Betty"
my_user.last_name = "Bar"
my_user.email = "airbnb@mail.com"
my_user.password = "root"
my_user.save()
print(my_user)

print("-- Create a new User 2 --")
my_user2 = User()
my_user2.first_name = "John"
my_user2.email = "airbnb2@mail.com"
my_user2.password = "root"
my_user2.save()
print(my_user2)

guillaume@ubuntu:~/AirBnB$ cat file.json ; echo ""
{"BaseModel.2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4": {"__class__": "BaseModel",
"id": "2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4", "updated_at": "2017-09-
28T21:11:14.333862", "created_at": "2017-09-28T21:11:14.333852"},
"BaseModel.a42ee380-c959-450e-ad29-c840a898cfce": {"__class__": "BaseModel",
"id": "a42ee380-c959-450e-ad29-c840a898cfce", "updated_at": "2017-09-
28T21:11:15.504296", "created_at": "2017-09-28T21:11:15.504287"},
"BaseModel.af9b4cbd-2ce1-4e6e-8259-f578097dd15f": {"__class__": "BaseModel",
"id": "af9b4cbd-2ce1-4e6e-8259-f578097dd15f", "updated_at": "2017-09-
28T21:11:12.971544", "created_at": "2017-09-28T21:11:12.971521"},
"BaseModel.38a22b25-ae9c-4fa9-9f94-59b3eb51bfba": {"__class__": "BaseModel",
"id": "38a22b25-ae9c-4fa9-9f94-59b3eb51bfba", "updated_at": "2017-09-
28T21:11:13.753347", "created_at": "2017-09-28T21:11:13.753337"},
"BaseModel.9bf17966-b092-4996-bd33-26a5353cccb4": {"__class__": "BaseModel",
"id": "9bf17966-b092-4996-bd33-26a5353cccb4", "updated_at": "2017-09-
28T21:11:14.963058", "created_at": "2017-09-28T21:11:14.963049"}}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ ./test_save_reload_user.py
-- Reloaded objects --
[BaseModel] (38a22b25-ae9c-4fa9-9f94-59b3eb51bfba) {'id': '38a22b25-ae9c-4fa9-
9f94-59b3eb51bfba', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 13,
753337), 'updated_at': datetime.datetime(2017, 9, 28, 21, 11, 13, 753347)}
[BaseModel] (9bf17966-b092-4996-bd33-26a5353cccb4) {'id': '9bf17966-b092-4996-
bd33-26a5353cccb4', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 14,
963049), 'updated_at': datetime.datetime(2017, 9, 28, 21, 11, 14, 963058)}
[BaseModel] (2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4) {'id': '2bf3ebfd-a220-49ee-
9ae6-b01c75f6f6a4', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 14,
```

```
333852), 'updated_at': datetime.datetime(2017, 9, 28, 21, 11, 14, 333862)}
[BaseModel] (a42ee380-c959-450e-ad29-c840a898cfce) {'id': 'a42ee380-c959-450e-
ad29-c840a898cfce', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 15,
504287), 'updated_at': datetime.datetime(2017, 9, 28, 21, 11, 15, 504296)}
[BaseModel] (af9b4cbd-2ce1-4e6e-8259-f578097dd15f) {'id': 'af9b4cbd-2ce1-4e6e-
8259-f578097dd15f', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 12,
971521), 'updated_at': datetime.datetime(2017, 9, 28, 21, 11, 12, 971544)}
-- Create a new User --
[User] (38f22813-2753-4d42-b37c-57a17f1e4f88) {'id': '38f22813-2753-4d42-b37c-
57a17f1e4f88', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 42, 848279),
'updated_at': datetime.datetime(2017, 9, 28, 21, 11, 42, 848291), 'email':
'airbnb@mail.com', 'first_name': 'Betty', 'last_name': 'Bar', 'password': 'root'}
-- Create a new User 2 --
[User] (d0ef8146-4664-4de5-8e89-096d667b728e) {'id': 'd0ef8146-4664-4de5-8e89-
096d667b728e', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 42, 848280),
'updated_at': datetime.datetime(2017, 9, 28, 21, 11, 42, 848294), 'email':
'airbnb2@mail.com', 'first_name': 'John', 'password': 'root'}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ cat file.json ; echo ""
{"BaseModel.af9b4cbd-2ce1-4e6e-8259-f578097dd15f": {"id": "af9b4cbd-2ce1-4e6e-
8259-f578097dd15f", "updated_at": "2017-09-28T21:11:12.971544", "created_at":
"2017-09-28T21:11:12.971521", "__class__": "BaseModel"}, "BaseModel.38a22b25-
ae9c-4fa9-9f94-59b3eb51bfba": {"id": "38a22b25-ae9c-4fa9-9f94-59b3eb51bfba",
"updated_at": "2017-09-28T21:11:13.753347", "created_at": "2017-09-
28T21:11:13.753337", "__class__": "BaseModel"}, "BaseModel.9bf17966-b092-4996-
bd33-26a5353cccb4": {"id": "9bf17966-b092-4996-bd33-26a5353cccb4", "updated_at":
"2017-09-28T21:11:14.963058", "created_at": "2017-09-28T21:11:14.963049",
"__class__": "BaseModel"}, "BaseModel.2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4":
{"id": "2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4", "updated_at": "2017-09-
28T21:11:14.333862", "created_at": "2017-09-28T21:11:14.333852", "__class__":
"BaseModel"}, "BaseModel.a42ee380-c959-450e-ad29-c840a898cfce": {"id": "a42ee380-
c959-450e-ad29-c840a898cfce", "updated_at": "2017-09-28T21:11:15.504296",
"created_at": "2017-09-28T21:11:15.504287", "__class__": "BaseModel"},
"User.38f22813-2753-4d42-b37c-57a17f1e4f88": {"id": "38f22813-2753-4d42-b37c-
57a17f1e4f88", "created_at": "2017-09-28T21:11:42.848279", "updated_at": "2017-
09-28T21:11:42.848291", "email": "airbnb@mail.com", "first_name": "Betty",
"__class__": "User", "last_name": "Bar", "password": "root"}, "User.d0ef8146-
4664-4de5-8e89-096d667b728e": {"id": "d0ef8146-4664-4de5-8e89-096d667b728e",
"created_at": "2017-09-28T21:11:42.848280", "updated_at": "2017-09-
28T21:11:42.848294", "email": "airbnb_2@mail.com", "first_name": "John",
"__class__": "User", "password": "root"}}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ ./test_save_reload_user.py
-- Reloaded objects --
[BaseModel] (af9b4cbd-2ce1-4e6e-8259-f578097dd15f) {'updated_at':
datetime.datetime(2017, 9, 28, 21, 11, 12, 971544), 'id': 'af9b4cbd-2ce1-4e6e-
8259-f578097dd15f', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 12,
971521)}
[BaseModel] (2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4) {'updated_at':
datetime.datetime(2017, 9, 28, 21, 11, 14, 333862), 'id': '2bf3ebfd-a220-49ee-
9ae6-b01c75f6f6a4', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 14,
333852)}
[BaseModel] (9bf17966-b092-4996-bd33-26a5353cccb4) {'updated_at':
datetime.datetime(2017, 9, 28, 21, 11, 14, 963058), 'id': '9bf17966-b092-4996-
bd33-26a5353cccb4', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 14,
963049)}
```

```
[BaseModel] (a42ee380-c959-450e-ad29-c840a898cfce) {'updated_at':
datetime.datetime(2017, 9, 28, 21, 11, 15, 504296), 'id': 'a42ee380-c959-450e-
ad29-c840a898cfce', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 15,
504287)}
[BaseModel] (38a22b25-ae9c-4fa9-9f94-59b3eb51bfba) {'updated_at':
datetime.datetime(2017, 9, 28, 21, 11, 13, 753347), 'id': '38a22b25-ae9c-4fa9-
9f94-59b3eb51bfba', 'created_at': datetime.datetime(2017, 9, 28, 21, 11, 13,
753337)}
[User] (38f22813-2753-4d42-b37c-57a17f1e4f88) {'password':
'63a9f0ea7bb98050796b649e85481845', 'created_at': datetime.datetime(2017, 9, 28,
21, 11, 42, 848279), 'email': 'airbnb@mail.com', 'updated_at':
datetime.datetime(2017, 9, 28, 21, 11, 42, 848291), 'last_name': 'Bar', 'id':
'38f22813-2753-4d42-b37c-57a17f1e4f88', 'first_name': 'Betty'}
[User] (d0ef8146-4664-4de5-8e89-096d667b728e) {'password':
'63a9f0ea7bb98050796b649e85481845', 'created_at': datetime.datetime(2017, 9, 28,
21, 11, 42, 848280), 'email': 'airbnb_2@mail.com', 'updated_at':
datetime.datetime(2017, 9, 28, 21, 11, 42, 848294), 'id': 'd0ef8146-4664-4de5-
8e89-096d667b728e', 'first_name': 'John'}
-- Create a new User --
[User] (246c227a-d5c1-403d-9bc7-6a47bb9f0f68) {'password': 'root', 'created_at':
datetime.datetime(2017, 9, 28, 21, 12, 19, 611352), 'email': 'airbnb@mail.com',
'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 19, 611363), 'last_name':
'Bar', 'id': '246c227a-d5c1-403d-9bc7-6a47bb9f0f68', 'first_name': 'Betty'}
-- Create a new User 2 --
[User] (fce12f8a-fdb6-439a-afe8-2881754de71c) {'password': 'root', 'created_at':
datetime.datetime(2017, 9, 28, 21, 12, 19, 611354), 'email': 'airbnb_2@mail.com',
'updated_at': datetime.datetime(2017, 9, 28, 21, 12, 19, 611368), 'id':
'fce12f8a-fdb6-439a-afe8-2881754de71c', 'first_name': 'John'}
guillaume@ubuntu:~/AirBnB$
guillaume@ubuntu:~/AirBnB$ cat file.json ; echo ""
{"BaseModel.af9b4cbd-2ce1-4e6e-8259-f578097dd15f": {"updated_at": "2017-09-
28T21:11:12.971544", "__class__": "BaseModel", "id": "af9b4cbd-2ce1-4e6e-8259-
f578097dd15f", "created_at": "2017-09-28T21:11:12.971521"}, "User.38f22813-2753-
4d42-b37c-57a17f1e4f88": {"password": "63a9f0ea7bb98050796b649e85481845",
"created_at": "2017-09-28T21:11:42.848279", "email": "airbnb@mail.com", "id":
"38f22813-2753-4d42-b37c-57a17f1e4f88", "last_name": "Bar", "updated_at": "2017-
09-28T21:11:42.848291", "first_name": "Betty", "__class__": "User"},
"User.d0ef8146-4664-4de5-8e89-096d667b728e": {"password":
"63a9f0ea7bb98050796b649e85481845", "created_at": "2017-09-28T21:11:42.848280",
"email": "airbnb_2@mail.com", "id": "d0ef8146-4664-4de5-8e89-096d667b728e",
"updated_at": "2017-09-28T21:11:42.848294", "first_name": "John", "__class__":
"User"}, "BaseModel.9bf17966-b092-4996-bd33-26a5353cccb4": {"updated_at": "2017-
09-28T21:11:14.963058", "__class__": "BaseModel", "id": "9bf17966-b092-4996-bd33-
26a5353cccb4", "created_at": "2017-09-28T21:11:14.963049"}, "BaseModel.a42ee380-
c959-450e-ad29-c840a898cfce": {"updated_at": "2017-09-28T21:11:15.504296",
"__class__": "BaseModel", "id": "a42ee380-c959-450e-ad29-c840a898cfce",
"created_at": "2017-09-28T21:11:15.504287"}, "BaseModel.38a22b25-ae9c-4fa9-9f94-
59b3eb51bfba": {"updated_at": "2017-09-28T21:11:13.753347", "__class__":
"BaseModel", "id": "38a22b25-ae9c-4fa9-9f94-59b3eb51bfba", "created_at": "2017-
09-28T21:11:13.753337"}, "BaseModel.2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4":
{"updated_at": "2017-09-28T21:11:14.333862", "__class__": "BaseModel", "id":
"2bf3ebfd-a220-49ee-9ae6-b01c75f6f6a4", "created_at": "2017-09-
28T21:11:14.333852"}, "User.246c227a-d5c1-403d-9bc7-6a47bb9f0f68": {"password":
"root", "created_at": "2017-09-28T21:12:19.611352", "email": "airbnb@mail.com",
"id": "246c227a-d5c1-403d-9bc7-6a47bb9f0f68", "last_name": "Bar", "updated_at":
"2017-09-28T21:12:19.611363", "first_name": "Betty", "__class__": "User"},
```

```
"User.fce12f8a-fdb6-439a-afe8-2881754de71c": {"password": "root", "created_at":
"2017-09-28T21:12:19.611354", "email": "airbnb_2@mail.com", "id": "fce12f8a-fdb6-
439a-afe8-2881754de71c", "updated_at": "2017-09-28T21:12:19.611368",
"first_name": "John", "__class__": "User"}}
guillaume@ubuntu:~/AirBnB$
```

No unittests needed for the console

Repo:

- GitHub repository: `AirBnB_clone`
- File: `models/user.py, models/engine/file_storage.py, console.py, tests/`

9. More classes!

mandatory

Write all those classes that inherit from `BaseModel`:

- `State` (`models/state.py`):
  Public class attributes:
    - `name`: string - empty string
- `City` (`models/city.py`):
  Public class attributes:
    - `state_id`: string - empty string: it will be the `State.id`
    - `name`: string - empty string
- `Amenity` (`models/amenity.py`):
  Public class attributes:
    - `name`: string - empty string
- `Place` (`models/place.py`):
  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` (`models/review.py`):
    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: `AirBnB_clone`
- File: `models/state.py, models/city.py, models/amenity.py, models/place.py, models/review.py, tests/`

10. Console 1.0

mandatory

Update `FileStorage` to manage correctly serialization and deserialization of all our new classes: `Place`, `State`, `City`, `Amenity` 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: `AirBnB_clone`
- File: `console.py, models/engine/file_storage.py, tests/`