

Demystifying Custom Ansible Modules

by Bianca Henderson

Who am I?

- Software engineer at Anaconda
- Previously worked at Red Hat, specifically on AWX (the open source version of Ansible Tower)
- Used to work on the Ansible Tower technical support team
- Taught myself Python via books + creating random things like text-based adventure games
- My favorite things to work on are CLIs and games
- Other interests: video games, board games, reading about science and math, playing music, 3D printing, crocheting, painting, writing, and tons of other things

What is this talk about?

- Writing your own Ansible modules doesn't have to be scary or complicated!
- There are simple parallels between Python scripts and Ansible modules
- Wide range of use-cases and possibilities

Who is it for?

- Anyone who uses Ansible or has been thinking about using it
- Anyone who wants to expand the functionality of their current Ansible playbooks
- People who are curious about how else their Python scripts/modules can be utilized

What is Ansible?

- IT automation tool
- Some of the things it can do:
 - Configure systems
 - Deploy software
 - Orchestrate continuous deployments
 - Schedule and execute zero downtime rolling updates, etc.
- Uses YAML "playbooks" to carry out tasks using modules and roles
- Ansible host machine communicates with nodes via SSH
- Free and open source!
- Very "batteries included" but also extremely customizable

What is an Ansible module?

"...a reusable, standalone script that Ansible runs on your behalf, either locally or remotely. Modules interact with your local machine, an API, or a remote system to perform specific tasks...A module provides a defined interface, accepts arguments, and returns information to Ansible by printing a JSON string to stdout before exiting."

Some preliminaries...

Adhere to the standard Ansible module format:

- Python shebang & UTF-8 encoding
- Copyright and license
- ANSIBLE METADATA block
- DOCUMENTATION block
- EXAMPLES block
- RETURN block (optional)
- Python imports
- Code

For more information, check out the <u>related documentation</u>

Directory / file structure for running an Ansible playbook using custom modules

```
— ansible.cfg
├─ library
   -- sum_two_module.py
practice_playbook.yml
```

ansible.cfg contents

```
[defaults]
library = ./library
retry_files_enabled = False
```

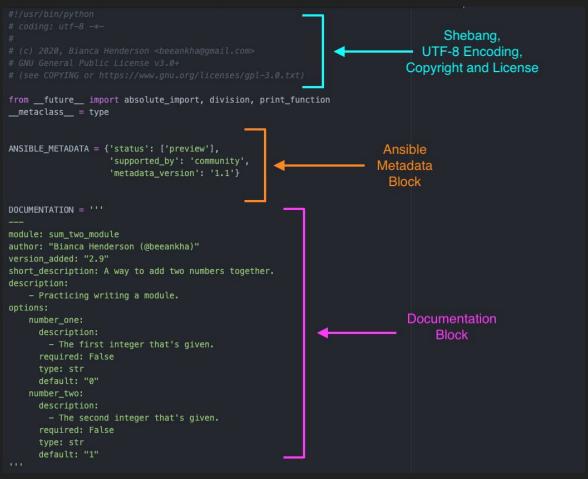
Now, let's look at some Python code and convert them into modules!

Summing two numbers with Python &

```
def sum(number_one, number_two):
    sum = number_one + number_two
    return sum
```

```
> python library/python_module_sum.py
The sum of 3 and 5 is 8
```

Summing two numbers via a custom Ansible module (1/2)



Summing two numbers via a custom Ansible module (1/2)

```
EXAMPLES = '''
- name: "Add two numbers"
  sum_two_module:
   number_one: 3
                                                          Examples
   number_two: 5
  register: sum_results
                                                            Block
- debug:
   var: sum_results
from ansible.module_utils.basic import AnsibleModule
def main():
   argument_spec = dict(
        number_one=dict(required=False, default='0', type='str'),
        number_two=dict(required=False, default='1', type='str'),
   module = AnsibleModule(argument_spec=argument_spec, supports_check_mode=True)
   number_one = module.params.get('number_one')
   number_two = module.params.get('number_two')
                                                                                                     Code
        json_output = {'sum': (int(number_one) + int(number_two))}
   except ValueError:
        module.fail_json(msg="You didn't pass in sum-able integers!")
   module.exit_json(**json_output)
if name == ' main ':
    main()
```

Summing two numbers via a playbook task

```
- name: "Run addition module"
 sum_two_module:
   number_one: 3
   number_two: 5
  register: sum_results
debug:
    var: sum_results
```

Successful playbook run output (for summing two numbers)

```
/CustomAnsibleModules > ansible-playbook practice playbook.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'
PLAY [My playbook]
TASK [Run addition module]
TASK [debug] **
ok: [localhost] => {
       "sum": 8
PLAY RECAP
localhost
                                    changed=0 unreachable=0 failed=0
                                                                              skippe
d=0
      rescued=0 ignored=0
```

Rolling a d20 via a Python script 📦

```
from random import randint
import sys
roll = (randint(1,20))
print(roll)
sys.exit()
```

...Converted to an Ansible Module (and made more flexible!) 🎲



```
DOCUMENTATION = '''
module: dice module
author: "Bianca Henderson (@beeankha)"
version added: "2.9"
short_description: The silliest way to roll some dice!
description:
    - A way to roll digital dice via Ansible playbook.
options:
                                              function
    dice_side_number:
                                                 arg
      description:

    Customize the number of sides for the die you roll.

      required: False
      type: int
      default: 20
                                             another
    number of rolls:
                                               arg
      description:
        - Choose how many times you want to roll the custom die.
      required: False
      type: int
      default: 1
111
```

Rolling dice via an Ansible module 🎲



```
import random
from ansible.module_utils.basic import AnsibleModule
                                                                           playbook
def custom dice roll(dice side number, number of rolls):
                                                                          task options
    dice_roll_results = []
    for _ in range(number_of_rolls):
        dice_roll_results.append(random.randint(1,int(dice_side_number)))
    return dice roll results
def main():
    argument_spec = dict(
        dice_side_number=dict(required=False, default=20, type='int'),
        number_of_rolls=dict(required=False, default=1, type='int'),
    module = AnsibleModule(argument_spec=argument_spec, supports_check_mode=True)
    dice_side_number = module.params.get('dice_side_number')
    number of rolls = module.params.get('number of rolls')
    try:
        json_output = {'custom_roll_result': custom_dice_roll(dice_side_number, number_of_rolls)}
    except ValueError:
        module.fail_json(msg='This module parameter takes an integer!')
    module.exit_json(**json_output)
if __name__ == '__main__':
    main()
```

Rolling dice via an Ansible playbook task 🎲

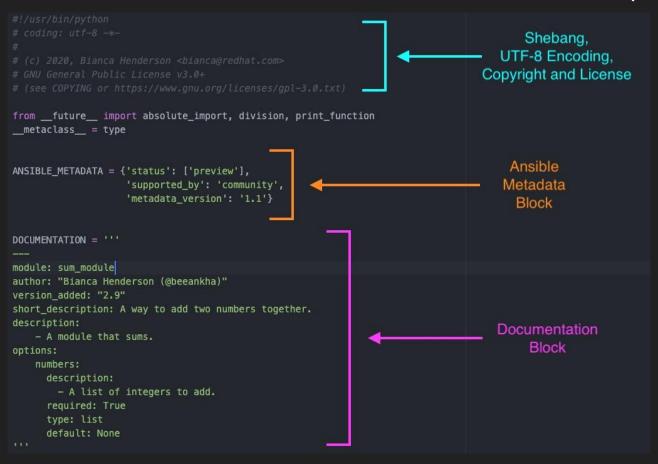
```
- name: "Roll some dice"
  dice_module:
    dice_side_number: 12
    number_of_rolls: 5
  register: roll_result
- debug:
    var: roll_result
```

Before we roll some dice...

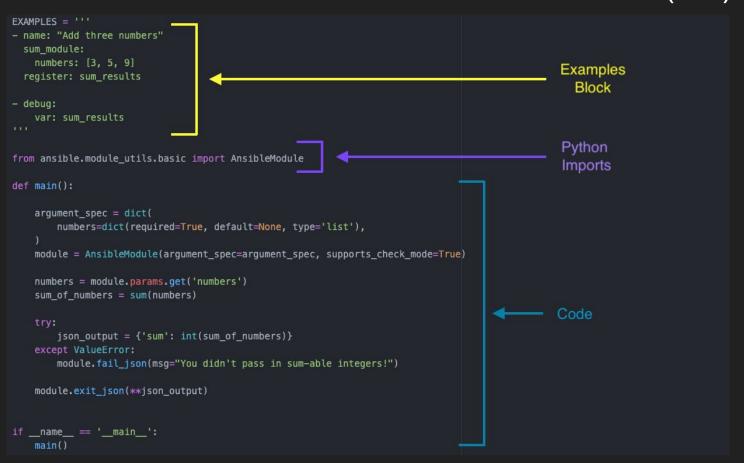
...let's refactor the

sum two module

A refactored Ansible module that sums a list (1/2)



A refactored Ansible module that sums a list (2/2)



Playbook tasks for rolling damage and getting the total 🐉



```
- name: "Roll damage with 4d8!"
  dice module:
    dice side number: 8
    number_of_rolls: 4
  register: damage_results
debug:
    var: damage results
- name: "Get the damage point total"
  sum module:
    numbers: "{{ damage results.custom roll result }}"
  register: damage sum
- debug:
    var: damage sum
```

Successful playbook run output (for rolling DnD damage)

Successful playbook run output (for rolling DnD damage)

```
TASK [Roll damage with 4d8!]
TASK [debug]
TASK [Get the damage point total]
TASK [debug]
        "sum": 8
```

Gathering data via Python (top stories from Hacker News)

```
from operator import itemgetter
import requests
url = 'https://hacker-news.firebaseio.com/v0/topstories.json'
r = requests.get(url)
print(f"Status code: {r.status code}")
submission_ids = r.json()
submission dicts = []
for submission_id in submission_ids[:30]:
   url = f"https://hacker-news.firebaseio.com/v0/item/{submission id}.json"
   r = requests.get(url)
   print(f"id: {submission_id}\tstatus: {r.status_code}")
    response_dict = r.json()
   submission dict = {
        'title': response_dict['title'],
        'hn_link': f"http://news.ycombinator.com/item?id={submission_id}",
        'story id': response dict['id'],
        'score': response_dict['score'],
        'author': response_dict['by'],
        'comments': response dict.get('descendants', 0),
   submission dicts.append(submission_dict)
submission_dicts = sorted(submission_dicts, key=itemgetter('comments'), reverse=True)
```

...converted into an Ansible module (1/3)

```
from collections import OrderedDict
from operator import itemgetter
import requests
def api_scrape_result(number_of_entries, story_id, author, comments):
    url = 'https://hacker-news.firebaseio.com/v0/topstories.json'
   r = requests.get(url)
   print(f"Status code: {r.status_code}")
   submission ids = r.json()[:number of entries]
    submission dicts = []
    for submission_id in submission_ids:
        url = f"https://hacker-news.firebaseio.com/v0/item/{submission_id}.json"
        r = requests.get(url)
        print(f"id: {submission_id}\tstatus: {r.status_code}")
        response_dict = r.json()
        submission dict = {
            'title': response_dict['title'],
            'score': response dict['score'],
            'hn_link': f"http://news.ycombinator.com/item?id={submission_id}",
        if story id:
            id_entry = {'story_id': response_dict['id']}
           submission_dict.update(id_entry)
            show author = {'author': response dict['bv'],}
           submission_dict.update(show_author)
            comment_number = {'comments': response_dict.get('descendants', 0)}
            submission_dict.update(comment_number)
        submission_dicts.append(submission_dict)
        submission_dicts = sorted(submission_dicts, key=itemgetter('score'), reverse=True)
    return submission dicts
```

...converted into an Ansible module (2/3)

```
def main():
   argument spec = dict(
        number of entries=dict(required=False, default=1, type='int'),
        story_id=dict(required=False, default=False, type='bool'),
        author=dict(required=False, default=False, type='bool'),
        comments=dict(required=False, default=False, type='bool'),
   module = AnsibleModule(argument spec=argument spec, supports check mode=True)
   number of entries = module.params.get('number of entries')
   story id = module.params.get('story id')
   author = module.params.get('author')
   comments = module.params.get('comments')
   try:
        json_output = {'top_hn_submissions': api_scrape_result(number_of_entries, story_id, author, comments)}
   except ValueError:
        module.fail_json(msg="Something went wrong!")
   module.exit json(**json output)
if __name__ == '__main__':
   main()
```

...converted into an Ansible module (3/3)

```
DOCUMENTATION = '''
module: hn_top_stories
author: "Bianca Henderson (@beeankha)"
version_added: "2.9"
short description: Hacker News API data gathering
description:
    - A way to display data from the top stories on Hacker News API.
    number_of_entries:
     description:
        - Number of entries to look at.
      required: False
      type: int
     default: 1
    story id:
      description:
        - Select if story ID should be displayed.
        required: False
        type: bool
        default: False
    author:
      description:

    Shows name of author.

        required: False
        type: bool
        default: False
    comments:
      description:
        - Select if number of comments should be displayed.
        required: False
        type: bool
        default: False
```

Gathering data via a custom module + playbook task

```
- name: "Get data from the Hacker News API"
  hn_top_stories:
    number_of_entries: 20
    story_id: True
    author: True
    comments: True
  register: top_stories
debug:
    var: top_stories
```

Let's run this playbook task!

(we'll get into environment setup details shortly)

What will the next demo playbook show?

- 1. Scrape data off of Hacker News and format it as HTML via a custom get news module
- 2. Create a www/directory via ansible.builtin.file
- 3. Create an HTML index file with the data gathered from the first task via ansible.builtin.copy module
- 4. Check for Docker or Podman installation + permissions via ansible.builtin.set-factmodules
- 5. Run an Nginx container web server using either Docker (ansible.builtin.shell) or Podman (container) to serve the static index.html file

Environment setup requirements / recommendations

- Create a conda environment that has the following installed:
 - Ansible
 - Docker or Podman
 - make
 - Python 3
 - requests

How to create and use a compatible conda environment

(after installing miniconda)

```
$ conda create -n ansible "python>=3" -y
$ conda activate ansible
```

(miniconda installer links)

Demo time!

Conclusions and key takeaways

- For simple use-cases, think of playbook options as the arguments in a Python function
- Make sure you adhere to the <u>standard Ansible module format / syntax</u>
- Use custom Ansible modules in combination with <u>"standard" Ansible modules</u> in order to expand any playbook's functionality
- You never know who else might find your custom module helpful!
 Custom Collections can be hosted on <u>Ansible Galaxy</u> for others to find and use

Reference materials

Developing Ansible Modules

https://docs.ansible.com/ansible/latest/dev_guide/developing_modules_general.html

Getting Started with Ansible

https://docs.ansible.com/ansible-core/devel/getting_started/index.html

Developing Custom Collections

https://docs.ansible.com/ansible/devel/dev_guide/developing_collections.html#dev_eloping-collections

GitHub Repository with Demo-Related Content

https://github.com/thenets/study-ansible/tree/main/how-to-create-a-module

Where to find today's presentation + demo materials

https://github.com/beeankha/CustomAnsibleModules

Thank you!

