Balchunas
AP Computer Science Principles
Problem Set 5 (Object Oriented Programming)

# Specification

Submit a .py file named `your_name_pset5.py`. Your file should **NOT** execute any functions. You are just filling them in. When I run your program there should be no errors and no output. You may choose to test your functions as you go, but you may delete the function calls after you test or comment them out. Comment your problems with the problem number above the function header. Remember to import the random module to use the library's functions.

# Cookie Jar

Here we will create a `Cookie` object, and a `Jar` which will hold a specific number of cookies. The specifications for the classes are shown below.

## Cookie Class

| **Attribute Name** | **Representation** |
|---|---|
| cookie_type | Choc Chip, Snickerdoodle, etc.<br>Set by creator |
| price | Cost of cookie in cents<br>Set by creator |
| bites_left | starts at 3 always, does not go below 0 |

| **Method Header** | **Functionality** |
|---|---|
| bite() | If there are bites left, print "chomp"<br>and subtract 1 from the number of bites, otherwise do nothing. |
| is_eaten() | return `True` if `bites_left` is 0<br>`False` otherwise. |
| __repr__() | return a string formatted as `Cookie(cookie_type, price)` |

## Jar Class

| **Attribute Nate** | **Representation** |
|---|---|
| capacity | The maximum number of cookies the jar can hold |
| cookies | A list of cookies currently in the jar<br>Starts as an empty list |

| Method Header | Functionality |
| --- | --- |
| deposit(cookie) | Adds a `cookie` to the jar if space remains.<br>Raise a `ValueError` if no space remains.<br>Raise a `ValueError` if the cookie has no bites left. |
| withdraw(n) | Removes `n` random cookies from the jar.<br>Returns a list of removed cookies.<br>Raise a `ValueError` if there are not enough cookies to withdraw. |
| withdraw(cookie_type) | Removes a cookie of type `cookie_type` if one exists.<br>Raise a `ValueError` if there is not a cookie of `cookie_type` in the jar. |
| size() | return the number of cookies currently in the jar. |
| cookie_counter() | return a dictionary where the keys are the different<br>types of cookiesand the values are the number<br>of those cookies currently in the jar. |
| value() | return the total value (in cents) of the cookies currently in the jar. |
| bites_left() | return the total number of cookie bites remaining in the jar. |
| __str__() | return a string of `n` 🍪<br>characters where `n` is the number of cookies in the jar. |

## Test Code

It's a good idea to test your code by doing the following:

1. Create 3 chocolate chip cookies at 125 cents each.
2. Create 2 sugar cookies at 100 cents each.
3. Create 4 m and m cookies at 225 cents each.
4. Create a cookie jar that holds 10 cookies.
5. Add all your cookies to the jar.
6. Remove 2 cookies, take two bites of each, and put them back.

Test the rest of your methods and check that they work as expected!