Name:_____ **Partner:** _____

Python Activity 37: Classes - Special Methods

Some common actions are simplified through implementing special methods.

Learning Objectives

Students will be able to:

Content:

- Define **special methods** in Python
- Identify which special method is being called implicitly
- Explain how to call a special method implicitly

Process:

- Write code that calls special methods implicitly
- Write code to implement special methods for user-defined types

Prior Knowledge

- Python concepts: user-defined classes, attributes, methods, __init__

Critical Thinking Questions:

1. Examine the following code below, which we've seen in previous activities.

book.py

```
0 class Book:
1     """ This class represents a book """

2     def __init__ (self, book_title, book_author, book_year):
3         self._title = book_title
4         self._author = book_author
5         self._year = book_year

8 if __name__ == "__main__":
9     lotr = Book("Fellowship of the Ring", "Tolkein", 1954)
10    print(lotr)
```

a. What do you think will be displayed by line 10?

_____

b. This code actually prints "<__main__Book object at 0x105eecca0". How does this differ from what you predicted in part (a)?

_____

c. Write a method for the `Book` class that will return a more meaningful *string* representation of `Book` objects:

_____

_____

_____

2. Examine the following code below, that extends our previous code:

book.py

```
0 class Book:
1     """ This class represents a book """

2     def __init__ (self, book_title, book_author, book_year):
3         self._title = book_title
4         self._author = book_author
5         self._year = book_year

6     def __str__(self):
7         return "'" + self._title + "', by " + self._author +
                  ", in " + str(self._year)

8 if __name__ == "__main__":
9     lotr = Book("Fellowship of the Ring", "Tolkein", 1954)
10    print(lotr)
```

a. Place a star next to the code that is new in this example.

b. How does the code on line 7 differ from the code you wrote in 1c?

_____

c. After running this code, line 10 will print the following, why might that be?
`'Fellowship of the Ring', by Tolkein, in 1954.`

_____

c. What method might the `print(..)` built-in function call *implicitly*?

_____

d. What other Python built-in function that we've used before might also call this method implicitly? _____ (*Hint: What function has a similar name?*)

e. What other Python *method* have we seen that starts & ends with double-underscore (`__xxxx__`)? _____

FYI: In Python, *methods* that begin and end with a double underscore (such as `__str__`) have special behaviors built-in to Python. They are called ***special methods***.

3. Match up special methods on the left-hand column with the code that implicitly calls them in the right-hand column (make educated guesses using special method names and parameters!):

Special Method	Called By
a. `__len__(self)`	`b = Book()`
b. `__init__(self)`	`len(m)`
c. `__str__(self)`	`b**2`
d. `__contains__(self, item)`	`b * 2`
e. `__eq__(self, other)`	`b < 5`
f. `__lt__(self, other)`	`b > 5`
g. `__gt__(self, other)`	`b + 2`
h. `__add__(self, other)`	`b == 5`
i. `__sub__(self, other)`	`b and True`
j. `__mul__(self, other)`	`22 in b`
k. `__truediv__(self, other)`	`str(b)`
l. `__pow__(self, other)`	`b / 5`
m. `__and__(self, other)`	`b - 2`

(*There's many more special methods, we'll see some of these again later!*)

Confirm your responses by checking the python3 documentation:

https://docs.python.org/3/reference/datamodel.html#special-method-names

Application Questions: Use Python to check your work

1. Implement additional special methods for our class, `Book`:
   a. Our `Book` class will include an attribute, `_words`, which is a list of strings representing all the words in the `Book` instance. As an example: `_words` for the `lotr` object looks something like: `["When", "Mr.", "Bilbo", "Baggins", "of", "Bag", "End", "announced", ...,"down", "into", "the", "Land", "of", "Shadow."]`

   Create a new *special* method which returns the number of words in the entirety of the book. As an example, `len(lotr)` should return `187790`.

   _____

   _____

   _____

b. Add a *special* method for `Book` that, when given a string, `word`, returns True if that word is in `_words`, False otherwise:

c. Add a *special* method for `Book`, that takes another Book object as a parameter and returns True if the two books are the same (defined as having the same title and author), False otherwise:

d. Create two different instances of `Book` objects:

e. Write some lines of code that use the *special* methods you wrote on the `Book` instance objects:

2. Create a new class, `Name`, which represents someone's name. When designing this class, consider: what should be the attributes? How should we initialize these values? What should the string representation look like? If we wanted to make an `initials()` method which returns just the initials of the name, how might we do that? What about a method `official()`, that returns the first and middle initial, and the full name?