14.11.4. Exercise

Here's a definition for a Date class that represents a date – that is, a year, month, and day of the month.

```
class Date:
    """Represents a year, month, and day"""
```

1. Write a function called make_date that takes year, month, and day as parameters, makes a Date object, assigns the parameters to attributes, and returns the result the new object. Create an object that represents June 22, 1933.
2. Write a function called print_date that takes a Date object, uses an f-string to format the attributes, and prints the result. If you test it with the Date you created, the result should be 1933-06-22.
3. Write a function called is_after that takes two Date objects as parameters and returns True if the first comes after the second. Create a second object that represents September 17, 1933, and check whether it comes after the first object.

Hint: You might find it useful to write a function called date_to_tuple that takes a Date object and returns a tuple that contains its attributes in year, month, day order.

Modification from Module 06: Lab 01:

14.11.4 - but use an

`__init__`

function. Create your class definition in a .py file using spyder, and then import it into a Jupyter Notebook, create an instance of the class and test it from the Jupyter Notebook. Make sure the .py file and the notebook are in the same directory. It will help to have spyder and jupyter lab both open at once.

In [1]:
```
#imports
from date_14_11_4 import Date
```

Part 1

In [2]:
```
june_22_1933 = Date(1933, 6, 22)
```

Part 2

```
In [3]: june_22_1933.print_date()
```

```
1933-June-22
```

Part 3

```
In [4]: from date_14_11_4 import is_after
```

```
In [5]: sept_17_1933 = Date(1933, 9, 17)
        print(is_after(sept_17_1933,june_22_1933))
```

```
True
```

Additional Testing

Checking that the leap year handling and error handling work.

```
In [6]: day_error_checking = Date(-44, 2, 30)
```

```
ERROR: Invalid Day.
```

```
In [7]: month_error_checking = Date(-44, 13, 29)
```

```
ERROR: Invalid Month.
13
```

```
In [8]: bc_leap_year = Date(-44, 2, 29)
```

```
In [9]: bc_leap_year.print_date()
```

```
44-February-29 BC
```

```
In [10]: bc_pre_leap_error_check = Date(-2400, 2, 29)
```

```
ERROR: Invalid Day.
```

```
In [11]: bc_pre_leap = Date(-2400, 2, 28)
```

```
In [12]: bc_pre_leap.print_date()
```

```
2400-February-28 BC
```

```
In [13]: print(is_after(bc_leap_year, bc_pre_leap))
```

```
True
```