How to Make Your Class iterable in Python

Let's create our **own custom Range class** — because why use Python's when we can reinvent the wheel... but with *style*!

What is an iterator ?

An **iterator** in Python is an object that lets you go through items one by one — like turning pages of a book, one page at a time.

In simpler words, think of a **book** — the book (list) has many pages (items). The **bookmark** you move page by page is the **iterator** — it remembers where you are and gives you the next page each time you ask.

What are iterables ?

An **iterable** in Python is something you can loop over — like a book that has many pages.

In simpler words, the **book** itself (with all its pages) is iterable — it contains the items. When you want to read it page by page, you use a **bookmark (iterator)** to go through it.

Code: *(I know here the code is looking dirty, github liink is in the caption)*

```
class CustomRange:
    def __init__(self,start=1,end=10):
        self.start=start
        self.end=end
    def __iter__(self):
        return self.CustomRange_Iterator(self) #this
CustomRange_Iterator object is iterator object beacuse it has 'next'
method. It is a rule that an object must have the 'next' method to
become an iterator object

    #this is not child class, this is inner class.
    #This inner class will return iterator object
    class CustomRange_Iterator:
        def __init__(self,counter): #as if this is an inner class so we
cannot access the variable 'start' and 'end' from here. So we passed
the 'CustomRange' class from the 'return
self.CustomRange_Iterator(self) ' as an argument. That object is
catched here with 'counter'. Now we can access. Because without that we
cannot define when to start and when to stop.
            self.counter=counter
            self.beg=counter.start
        def __next__(self):
            if self.counter.start>self.counter.end:
                self.counter.start=self.beg
                raise StopIteration
            current_value=self.counter.start
            self.counter.start+=1
            return current_value


my_range=CustomRange(1,5)
#here the 'i' is calling the 'iter' method of the given object(here
'my_range')
print('1st time->')
for i in my_range:
    print(i)

print('2nd time->')
for i in my_range:
    print(i)
```