

AI/HUB

Lesson 3

Object Oriented Programming & Custom Iterators

SUMMARY

For this lesson, we will be reviewing the principles of object-oriented programming in Python. As well as, the creation of custom iterators.

BASIC CLASSES

The following is a basic example of a Python class.

- Look at how the object is instantiated and some of the properties/methods you can access from the class.
- Observe the output of each print function listed below to understand what they do.

```
class MyClass:
    "This is my second class"
    a = 10
    def func(self):
        print('Hello')

ob = MyClass()

# Output: 10
print(MyClass.a)
print(ob.a)

# Output: <function MyClass.func at 0x000000003079BF8>
print(MyClass.func)

# Output: 'This is my second class'
print(MyClass.__doc__)
```

INITIALIZATION & CONSTRUCTORS

The following is a basic example of a class with a constructor.

- You can see what gets called within the constructor upon instantiation of the class
- Also note the name of the function that is considered the constructor (`__init__`)
- *This is a built-in Python function interpreted at runtime – there are many of these!*

```
class ComplexNumber:
    def __init__(self,r = 0,i = 0):
        self.real = r
        self.imag = i

    def getData(self):
        print("{0}+{1}j".format(self.real,self.imag))

# Create a new ComplexNumber object
c1 = ComplexNumber(2,3)

# Call getData() function
# Output: 2+3j
c1.getData()
```

NOTE

Variables that are defined within the namespace of the class are *only* accessible within the class. See the example on the first page for a method of accessing a class' public variables.

CUSTOM ITERATORS

Python allows programmers to create their own custom iterators.

- Study the snippet of code below to understand the built-in functions included.

```
class CustomRange:
    def __init__(self, max):
        self.max = max

    def __iter__(self):
        self.curr = 0
        return self

    def next(self):
        numb = self.curr
        if self.curr >= self.max:
            raise StopIteration
        self.curr += 1
        return numb

for i in CustomRange(10):
    print i

# Output:
# 0 1 2 3 4 5 6 7 8 9
```

EXERCISE YOUR PYTHON

1. Create a new class called 'toolkit', using all the custom functions you created in the previous lesson.
 - a. Add a new function which prints the odd numbers from the given list of numbers in the even numbers function (*1st activity of the last lesson*)

NOTE

This means that you should be creating a new function within your class to print all the odd numbers of the given list. Call this new function within your outer 'even-number' function.

CHALLENGE

Create a custom iterator that prints the Fibonacci sequence up to a given integer 'n'.

For example, if given 'n = 6', output should print '1, 1, 2, 3, 5, 8' (6 steps).

