DATA ANALYTICS REFERENCE DOCUMENT	
Document Title:	Data Class in python
Document No.:	1597156878
Author(s):	Gerhard van der Linde
Contributor(s):	

REVISION HISTORY

Revision	Details of Modification(s)	Reason for modification	Date	By
0	Draft release	A brief summary on data classes	2020/08/11 14:41	Gerhard van der Linde

Data Class in Python

Create three data classes called a, b and c with a and b contained in c.

Also note the `iter` function is the classes and the `yield` instead of `return` that allows iteration in the `iter` function without the need for a `next` function inside the class.

```
class a():
    def __init__(self):
        self.val1=123
        self.str1='abc'
    def __iter__(self):
        for attr,val in self.__dict__.items():
            yield attr,val
class b():
    def __init__(self):
        self.val2=456
        self.str2='def'
    def __iter__(self):
        for attr,val in self.__dict__.items():
            yield attr,val
class c(a,b):
    def __init__(self):
        self.c=dict((zip(('val3','str3'),(789,'ghi'))))
        self.a=dict(a())
        self.b=dict(b())
    def __iter__(self):
        for attr,val in self.__dict__.items():
            yield attr,val

if __name__ == '__main__':
  data=c()
  for key,value in dict(data).items():
    print(f'{key}: {value}')
```

The 'main' function then instantiates the data class that now contains a and b in c.

Data is then type casted to a dictionary that yields the key/value pairs in the for loop and prints them to the console.

```
c: {'val3': 789, 'str3': 'ghi'}
a: {'val1': 123, 'str1': 'abc'}
b: {'val2': 456, 'str2': 'def'}
```

The really cool feature of this approach is the ability to keep on dynamically adding variables and getting the iterator to expose them dynamically without knowing what is in the class when passed between functions or even applications.

This approach also sets up the class for passing on json strings between applications with very little code required to prepare the string.

```
import json

json.dumps(dict(data),indent=2)
```

The resulting output is:

```
{
  "c": {
    "val3": 789,
    "str3": "ghi"
  },
  "a": {
    "val1": 123,
    "str1": "abc"
  },
  "b": {
    "val2": 456,
    "str2": "def"
  }
}
```