Iterators, Iterables and Generators in python explained | Advanced python tutorials

Introduction: in this blog we will talk about:

- Iterable
- Iterator
- Iteration
- Generators

Note: Everything is in objects here so If you want to recall objects then go to this link!

Iterable: It is an object in which __iter__ or __getitem__ method is defined. It means this is an object which can give you an iterator.

Iterator: It is an object in which __next__ method is defined. It means you can get next element through this method.

Iteration: Fetching it's next element is called iteration.

Generators: In a function if the data is printed or returned then it's a simple function but if it's yielded then it's a generator.

When to use generator?

If some process takes a big chunk of RAM then it's better to use generator.

Why to use generator?

Well generator could become a hard topic to understand when it comes to implementations. But generator could be useful when we need on the fly data. For example:

```
def gen(n):
    for i in range(n):
        yield i

for i in gen(100000):
    print(i)
```

Here we are getting numbers on the fly. To understand this we first need to understand **How generators work?**
Usually you would print the data or return the data or store the data somewhere but in generators you yield the data.
How is yield different from simply returning the value?
In returning the value it is processing all that data and storing it in RAM but in yield it processes data and after returning it, it moves to next value and doesn't store the old one, that way it saves alot of RAM.

#tut4.py file as described in the video

Copy

```
'''
Iterable -
Iterator -
Iteration -

'''

def gen(n):
    for i in range(n):
        yield i

ob1 = gen(4)
# print(next(ob1))
# print(next(ob1))
# print(next(ob1))
# print(next(ob1))
# print(next(ob1))
num = "harry"
```

```
iter1 = iter(num)
print(next(iter1))
print(next(iter1))
print(next(iter1))
print(next(iter1))
print(next(iter1))
print(next(iter1))
print(next(iter1))
print(next(iter1))
```

Previous

Next

CodeWithHarry