Elisha Vernee Hart 3/6/2023

Decorator, Generator, And Modules Revision

A Generator is a function responsible for generating a sequence of values. A Generator is just like an ordinary function except it uses yield instead of return.
An example is:

```
def func_gen( ):
     i = 0
    while i < 5:
          yield i
          i += 1

values = func_gen( )

for i in values:
          print(i)
```

Generators are easy to use. It improves memory utilization and performance. It is best suitable for reaching bulk Data.

Generators work great for web scraping and crawling.

A Decorator is a function which takes the function as a function argument and extends its functionality of existing functions without changing the function definition.

```
def outer(func):
     def inner(age):
          if age > 18:
               print('Valid Voter')
          else:
              func(age)
    return inner
@outer
  Def voters(age):
      print('Not a valid voter')

voters(10)
```

Another example is 🙂

```
def smart_division(func):
   Def inner(x, y):
         If y ==0:
                  print(‘Division by 0 is not possible.’)

                  return
         else:
                  return func(x, y)
   return inner

@smart_division

def division(a,b):
         print(a/b)

division(9, o)
```

The main advantage of the Decorator is to extend the functionality of the existing functions without changing the function definition.

Defining a Decorator involves Modules. A Module in Python is the normal python program, with .py extension. Module is a collection of functions, variables, classes, and objects.

An example is:

```
from process_mod import a, square

print(a)

print(square(3))
```