def keyword is used to create a function

In [3]:
```
l = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]

def test(a):
    n = []
    for i in a:
        if i%2 != 0:
            n.append(i)
    return n
```

In [4]:
```
test(l)
```

Out[4]:
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

""" *args = to input n number of inputs* *kwargs = to input n number of key and value pairs """

In [6]:
```
def test1(*args):
    return args
```

In [8]:
```
test1(12,2,3,3,4,5)
```

Out[8]:
```
(12, 2, 3, 3, 4, 5)
```

In [9]:
```
def test2(**kwargs):
    return kwargs
```

In [10]:
```
test2(a = 34, b = "True",c = 24.5)
```

Out[10]:
```
{'a': 34, 'b': 'True', 'c': 24.5}
```

In [17]:
```
# iterator = that iterates collection of objects
# iter(), for loop

l = [2,4,6,8,10,12,14,16,18,20]

for i in l:
    if i == 12:
        break
    print(i)
```

```
2
4
6
8
10
```

In [29]:
```
# Generator function - to create your own iterator function
# yield function - returns a value and pauses the execution

def test5(x):
    for i in x:
        yield i
```

In [30]:
```
test5(2)
```

Out[30]:
```
<generator object test5 at 0x00000290B38A8120>
```

In [32]:
```
def prime_gen():
    prime = []
    num = 2

    while num < 1000:
        is_Prime = True

        for primes in prime:
            if num%primes == 0:
                is_Prime = False
                break

        if is_Prime:
            prime.append(num)
            yield(num)

        num += 1

p = prime_gen()

for i in range(20):
    print(next(p))
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
```