In [1]:
```
#Q1
''' "def" is used to create a functionx'''
def sum25():
    s=0
    for i in range(26):
        s+=i
    return s

sum25()
```

Out[1]:
```
325
```

In [2]:
```
#Q2
''' *args is used if we want to pass any number of arguements to our function.
**kwargs is used if we want to feed a dictionary into our function'''
def sumn(*args):
    i=0
    for j in args:
        i+=j
    return i

print(sumn(1,2,3,3,5,6,5))
####
def book_list(**kwargs):
    print(kwargs)
book_list(english=6,bio=9,maths=8)
```

```
25
{'english': 6, 'bio': 9, 'maths': 8}
```

In [3]:
```
#Q3
'''An object that can be iterated upon, or traversed through all the values, is kno
__iter__(): The iter() method is called for the initialization of an iterator. This
__next__():This method used for iteration.
'''
lst=[2, 4, 6, 8, 10, 12, 14, 16,18, 20]
iterator = iter(lst)

print(next(iterator))
print(next(iterator))
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

```
2
4
6
8
10
```

In [4]:
```
#Q4
'''Programmers can create iterators quickly, easily, and cleanly by using Python's
An object that has the ability to iterate or loop is known as an iterator.
A container of data is abstracted using this technique to make it behave like an it
```

```
Yield keyword is used to create a generator function.'''

def mygenerator():
    print('First item')
    yield 10

    print('Second item')
    yield 20

    print('Last item')
    yield 30
for gen in mygenerator():
    print(gen)
```

```
First item
10
Second item
20
Last item
30
```

In [5]:

```
#Q5
def generatorprime(n):
    i=2
    while(i<n):
        prime=True
        for a in range(2,i):
            if(i%a==0):
                prime=False
                break
        if(prime):
            yield i
        i += 1

k=0
for prime in generatorprime(1000):
    if(k<20):
        print(prime)
        k+=1
```

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

In [ ]:

In [ ]: