Q1. Which keyword is used to create a function? Create a function to return a list of odd numbers in the range of 1 to 25.

In [1]:
```
#  The def keyword is used to create a function
```

In [18]:
```
b=range(1,25)
```

In [17]:
```
def test1 (a):
    n=[]
    for i in range(1,25):
        if i%2 !=0:
            n.append(i)
    return n
```

In [19]:
```
test1(b)
```

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

Q2. Why *args and **kwargs is used in some functions? Create a function each for args and **kwargs to demonstrate their use.

In [21]:
```
# args is used to pass the n number of argument in the function
# *kwargs is used for passe the n number of key,value pairs
def test2(*args,**kwargs):
    return args,kwargs
```

In [25]:
```
test2("vivek","kkum",1,2,3,a=5,b=6,c=10)
```

Out[25]:
```
(('vivek', 'kkum', 1, 2, 3), {'a': 5, 'b': 6, 'c': 10})
```

Q3. What is an iterator in python? Name the method used to initialise the iterator object and the method

used for iteration. Use these methods to print the first five elements of the given list [2, 4, 6, 8, 10, 12, 14, 16, 18, 20].

In [60]:
```
#  ans :- An iterator is an object that contains a countable number of values.
b=[2, 4, 6, 8, 10, 12, 14, 16,18, 20]
```

In [68]:
```
def test4(a):
    n=[]
    for i in b:
        if i <=10:
            n.append(i)
    return n
```

In [71]:
```
test4(b)
```

Out[71]:
```
[2, 4, 6, 8, 10]
```

Q4. What is a generator function in python? Why yield keyword is used? Give an example of a generatorfunction.

In [72]:
```
#  ans:-  a generator is a function that returns an iterator that produces a sequence of values when iterated over.
#  yield keyword is used for the value of yield function send out of loop
```

In [73]:
```
#  example of generator function
```

In [84]:
```
def fibn(n):
    a,b=0,1
    for i in range(n):
        yield a
        a,b=b,a+b
```

In [85]:
```
fibn(10)
```

Out[85]:
```
<generator object fibn at 0x00000203B60EC7B0>
```

In [86]:
```
for i in fibn(10):
    print(i)
```

```
0
1
1
2
3
5
8
13
21
34
```

Q5. Create a generator function for prime numbers less than 1000. Use the next() method to print the first 20 prime numbers.

In [17]:
```
def test7(x):
    if x>1:
        for i in range(2,x):
            if x%i==0:
                print(x,"not a prime num")
                break
            else:
                print(x,"prime num")
        else:
                print(X,"not a prime")
        return (x)
```

In [22]:
```
test7(4)
```

Out[22]:
```
4 not a prime num
4
```

In [ ]:
```

```