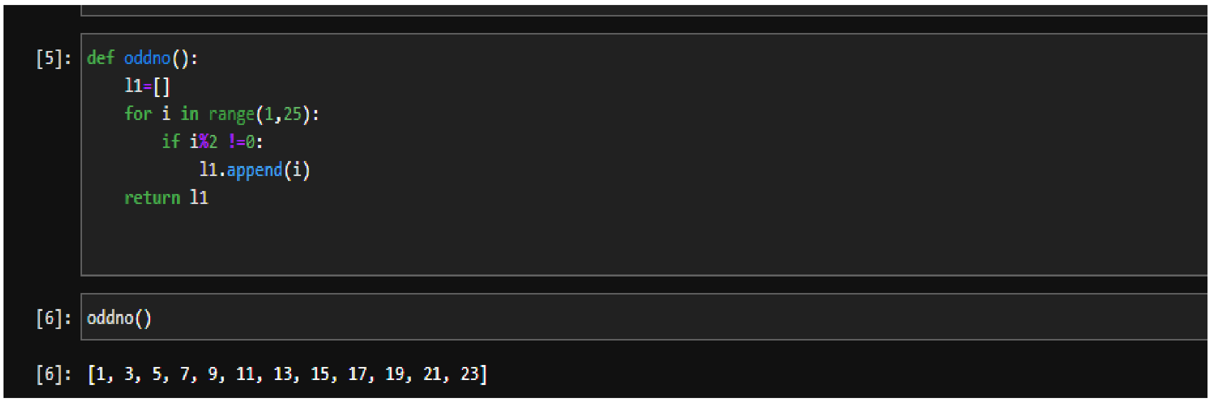
ASSIGNMENT-3

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.

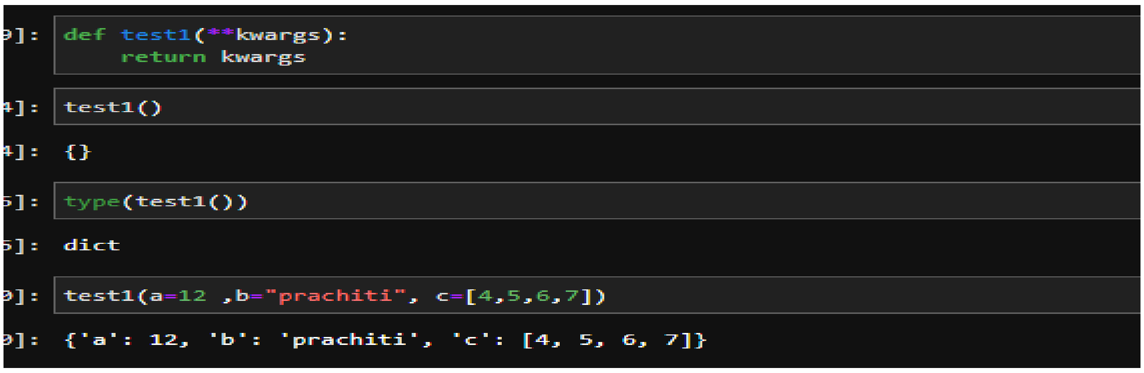


```
[5]: def oddno():
         l1=[]
         for i in range(1,25):
             if i%2 !=0:
                 l1.append(i)
         return l1

[6]: oddno()

[6]: [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.

Ans: *args (Non-Keyword Arguments):

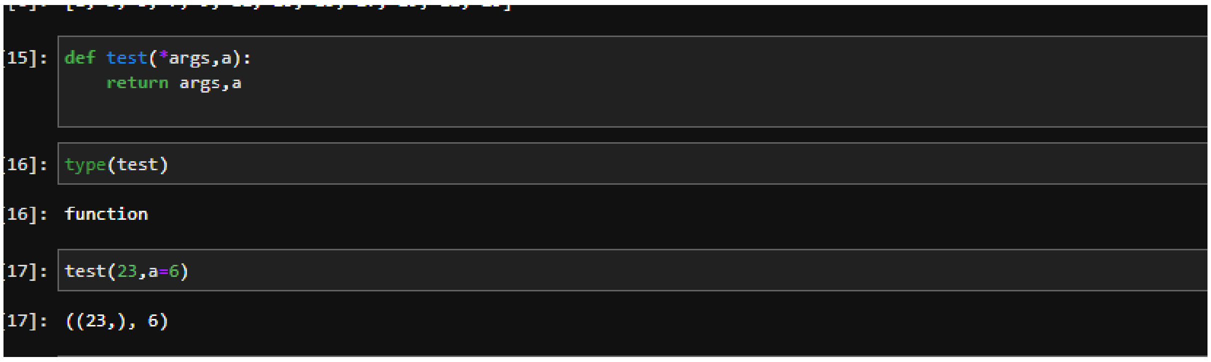
**args* in function definitions in Python is used to pass a variable number of arguments to a function. It is used to pass a non-keyworded, variable-length argument list.



```
9]:  def test1(**kwargs):
         return kwargs

4]:  test1()

4]:  {}

5]:  type(test1())

5]:  dict

9]:  test1(a=12 ,b="prachiti", c=[4,5,6,7])

9]:  {'a': 12, 'b': 'prachiti', 'c': [4, 5, 6, 7]}
```

- ****kwargs (Keyword Arguments)**

***kwargs* in function definitions in Python is used to pass a keyworded, variable-length argument list. We use the name *kwargs* with the double star. The reason is that the double star allows us to pass through keyword arguments



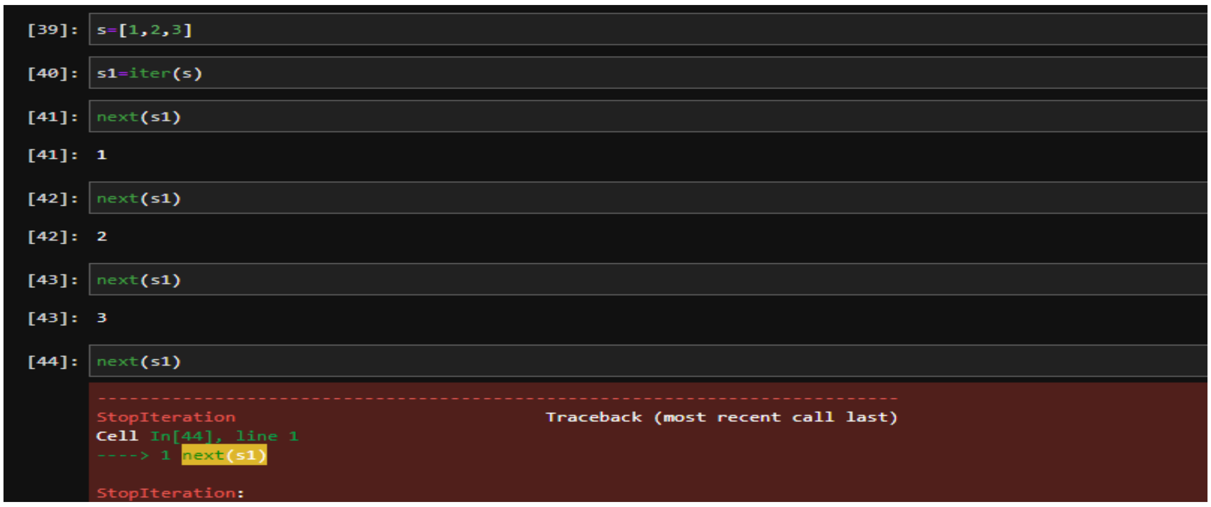
```
[15]: def test(*args,a):
          return args,a

[16]: type(test)

[16]: function

[17]: test(23,a=6)

[17]: ((23,), 6)
```

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].

Ans:

An iterator in Python is an object that is used to iterate over iterable objects like lists, tuples, dicts, and sets. The Python iterators object is initialized using the **iter()** method. It uses the **next()** method for iteration.

- **__iter__():** The iter() method is called for the initialization of an iterator. This returns an iterator object
- **__next__():** The next method returns the next value for the iterable. When we use a for loop to traverse any iterable object, internally it uses the iter() method to get an iterator object, which further uses the next() method to iterate over. This method raises a StopIteration to signal the end of the iteration.



```
[39]: s=[1,2,3]

[40]: s1=iter(s)

[41]: next(s1)

[41]: 1

[42]: next(s1)

[42]: 2

[43]: next(s1)

[43]: 3

[44]: next(s1)

      ---------------------------------------------------------------------------
      StopIteration                             Traceback (most recent call last)
      Cell In[44], line 1
      ----> 1 next(s1)

      StopIteration:
```

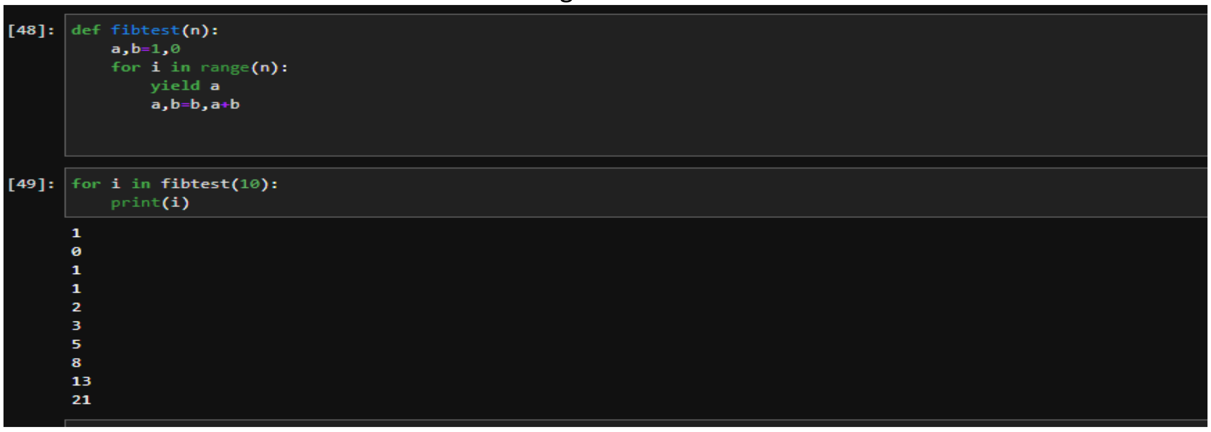
Q4. What is a generator function in python? Why yield keyword is used? Give an example of a generator function.

- **Generator:**

A generator function is defined just like a regular function in Python, but it contains one or more yield statements. Instead of using return to send a value back and exit the function, a yield statement is used to temporarily suspend the function's execution and produce a value to the caller. The state of the function is saved, allowing it to be resumed from where it left off.

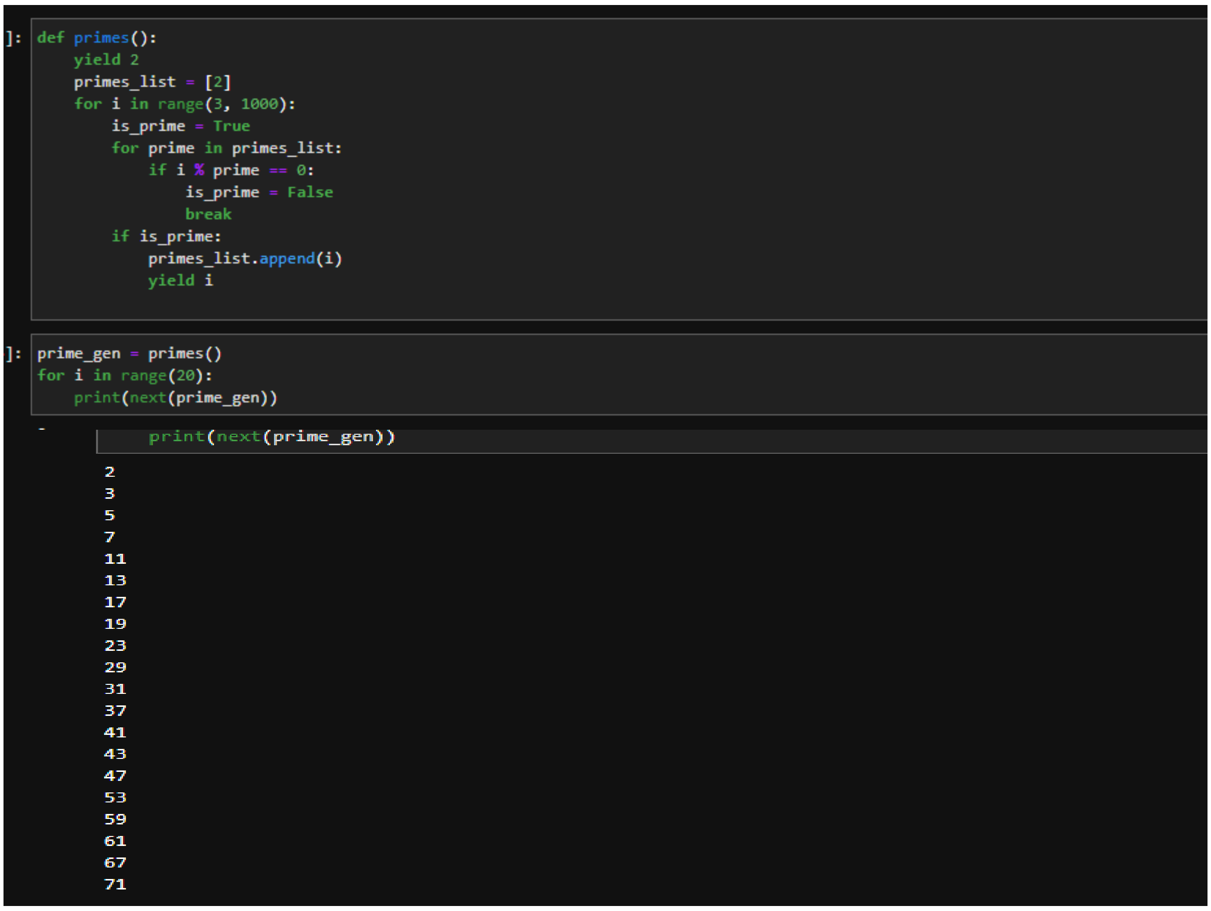
- **Yield:**

When the yield statement is encountered, the current value of i is produced, and the function's state is saved. The function can be resumed later to generate the next value.



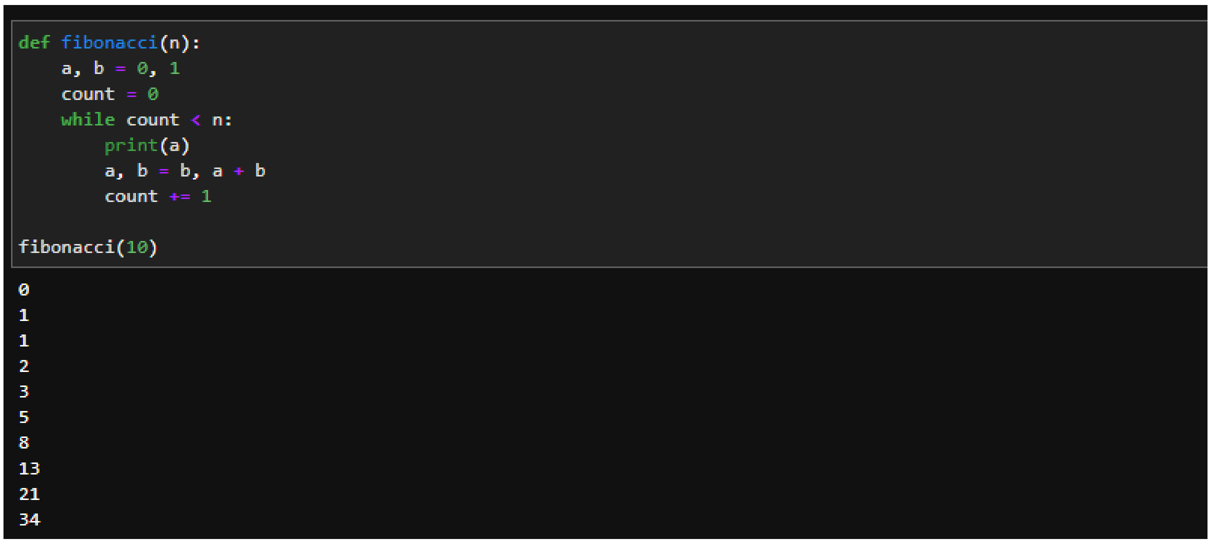
```
[48]: def fibtest(n):
          a,b=1,0
          for i in range(n):
              yield a
              a,b=b,a+b

[49]: for i in fibtest(10):
          print(i)

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

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



```
]: def primes():
       yield 2
       primes_list = [2]
       for i in range(3, 1000):
           is_prime = True
           for prime in primes_list:
               if i % prime == 0:
                   is_prime = False
                   break
           if is_prime:
               primes_list.append(i)
               yield i

]: prime_gen = primes()
   for i in range(20):
       print(next(prime_gen))

           print(next(prime_gen))

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

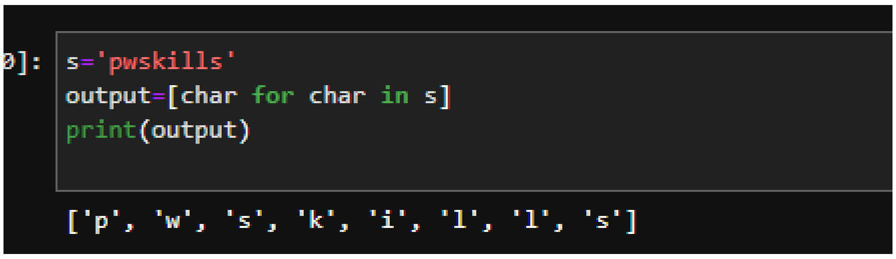
Q6. Write a python program to print the first 10 Fibonacci numbers using a while loop.
Ans:



```
def fibonacci(n):
    a, b = 0, 1
    count = 0
    while count < n:
        print(a)
        a, b = b, a + b
        count += 1

fibonacci(10)
```

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

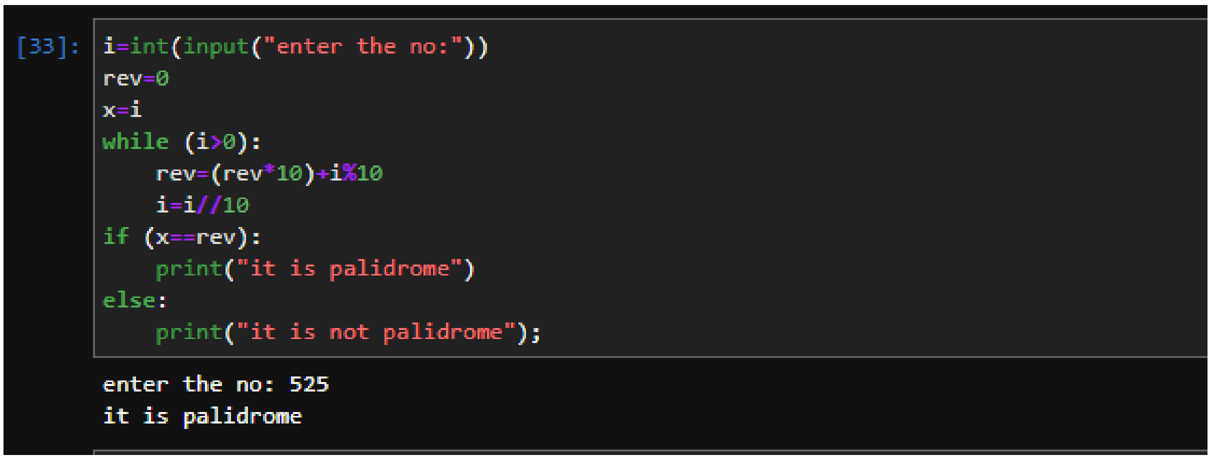
Q7. Write a List Comprehension to iterate through the given string: 'pwskills'.
Expected output: ['p', 'w', 's', 'k', 'i', 'l', 'l', 's']
Ans:



```
0]: s='pwskills'
    output=[char for char in s]
    print(output)
```

```
['p', 'w', 's', 'k', 'i', 'l', 'l', 's']
```

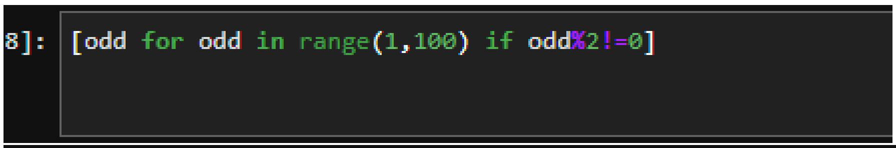
Q8. Write a python program to check whether a given number is Palindrome or not using a while loop.
Ans:



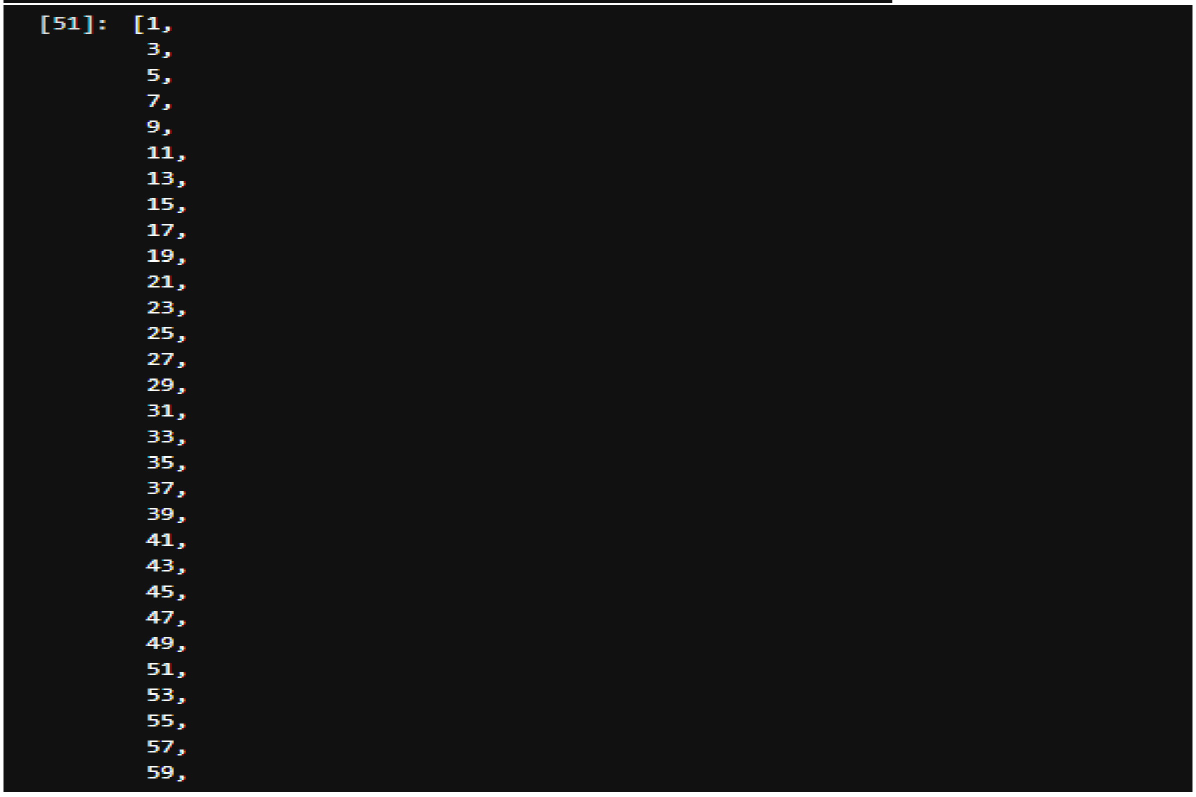
```
[33]: i=int(input("enter the no:"))
      rev=0
      x=i
      while (i>0):
          rev=(rev*10)+i%10
          i=i//10
      if (x==rev):
          print("it is palidrome")
      else:
          print("it is not palidrome");
```

```
enter the no: 525
it is palidrome
```

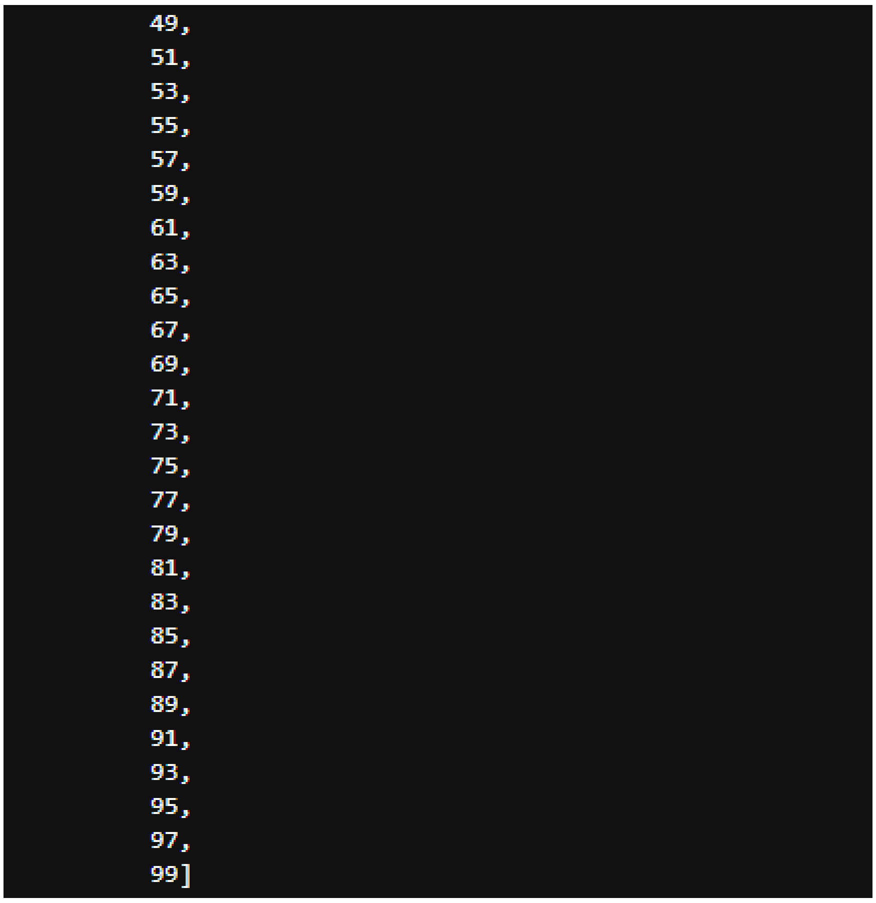
Q9. Write a code to print odd numbers from 1 to 100 using list comprehension.
Ans:



```
8]:  [odd for odd in range(1,100) if odd%2!=0]
```



```
[51]:  [1,
        3,
        5,
        7,
        9,
        11,
        13,
        15,
        17,
        19,
        21,
        23,
        25,
        27,
        29,
        31,
        33,
        35,
        37,
        39,
        41,
        43,
        45,
        47,
        49,
        51,
        53,
        55,
        57,
        59,
```



```
        49,
        51,
        53,
        55,
        57,
        59,
        61,
        63,
        65,
        67,
        69,
        71,
        73,
        75,
        77,
        79,
        81,
        83,
        85,
        87,
        89,
        91,
        93,
        95,
        97,
        99]
```