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.

ans : def key used for create function.

In [1]:

```
x = range(1,26)
def odd_numbers(x):
    odd_nums = []
    for i in x:
        if i % 2 != 0:
            odd_nums.append(i)
    return odd_nums

odd_numbers(x)
```

Out[1]:

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

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

ans : *args is used to pass a variable number of non keyworded arguments to a function. It allows you to pass an arbitrary number of arguments to a function in a single call.

```
 *kwargs on the other hand, allows you to pass a variable number of keyword arguments  to a function.
```

In [2]:

```
#*args
def name(*args):
    for args in args:
        print(args)
name('abhi','naitik','varun','rahul')

#**kwargs
def my_name(**kwargs):
    return kwargs
my_name(first='shah', mid='vishal', last='bharatbhai')
```

```
abhi
naitik
varun
rahul
```

Out[2]:

```
{'first': 'shah', 'mid': 'vishal', 'last': 'bharatbhai'}
```

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 : iterator is an object that allows you to iterate over collections of data, such as lists, tuples, dictionaries, and sets.

```
method : iter()
         next()
```

In [3]:

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

my_iter = iter(l)
for i in range(5):
    print(next(my_iter))
```

```
2
4
6
8
10
```

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

ans : generator is a function that returns an iterator that produces a sequence of values.

yield : the yield keyword will convert an expression that is specified along with it to a generator object and return it to the caller.

In [4]:

```
def fibonacci():
    a, b = 0, 1
    for i in range(1,8):
        print(b)
        a, b = b, a + b
fibonacci()
```

```
1
1
2
3
5
8
13
```

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

In [5]:

```
def primes():

    primes = [2]
    yield 2
    for i in range(3, 1000, 2):
        if all(i % p != 0 for p in primes):
            primes.append(i)
            yield i
prime_gen = primes()
for _ in range(20):
    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.

In [6]:

```
def fibonacci():
    a, b = 0, 1
    for i in range(1,11):
        print(b)
        a, b = b, a + b
fibonacci()
```

```
1
1
2
3
5
8
13
21
34
55
```

Q7. Write a List Comprehension to iterate through the given string: ‘pwskills’. Expected output: ['p',

'w', 's', 'k', 'i', 'l', 'l', 's']

In [7]:

```
string = "pwskil"
s = []
for i in string:
    s.append(i)
print(s)
```

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

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

In [8]:

```
number = int(input("Enter a number: "))
temp = number
reverse = 0

while temp > 0:
    digit = temp % 10
    reverse = reverse * 10 + digit
    temp //= 10

if number == reverse:
    print(number, "is a palindrome number")
else:
    print(number, "is not a palindrome number")
```

```
Enter a number: 23432
23432 is a palindrome number
```

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

In [9]:

```
num = []
for i in range(1,101):
    if i%2 !=0:
        num.append(i)
print(num)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 3
9, 41, 43, 45, 47, 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]
```

In [ ]: