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 - Keyword is def

In [1]:
```
#Here is a function that return a list of odd numbers in range of 1 to 25
#first create "def"function
def get_odd_numbers():
    odd_numbers = [] #odd numbers ko store karne k liye ek khali list banaya h
    for number in range(1,26): #According to question range hamne 1 to 26 liya h
                               #for loop siliye use kar rahe h q ki ek ek numbe

     if number % 2!= 0 :#jadi har ek number 2 se divide hone k baad not equal to
            odd_numbers.append(number) #opper wala satisfy hone k baad numbers a
    return odd_numbers #uske baad return ayga odd_numbers
#ye loop continously chalte rahega jab tak 1 to 25 na ho jaye

Odd_numbers_list = get_odd_numbers() #Uske baad odd numbers list me , jo functio

print(Odd_numbers_list) # abhi hamlog log print() function ka use karke odd numb
```

```
[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: - The *args and **kwargs are used in function to allow for variable numbers of arguments - *args is used to pass a variable number of non-keyword arguments to a functon -**kwargs is used to pass a variable of keyword arguments to a function . It allows you to pass any number of named arguments to a function

In [2]:
```
# Using *args
def sum_numbers(*args): #create def function for any number sum
    return sum(args) #uske baad return de dega

print(sum_numbers(1,2,3)) # yan pe hamne arguments pass kara diya , function ka
#jaise hi hum sum_numbers() ko recall karenge , body k ander kuch v function lek
```

```
6
```

In [3]:
```
# using **kwargs - iako use hum key : value pair k liye use karte h
def print_details(**kwargs):#create function
    #abhi hum log body k ander likenge
    return kwargs
#recall print_details function
print_details(name="sujen" , age=20 , city="Noamundi") #acoording to function ,
```

Out[3]:
```
{'name': 'sujen', 'age': 20, 'city': 'Noamundi'}
```

Q3. What is an iterator in python? Name the method used to initialise the iterator object and the methodused 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 interator in python is an object that implements the interator protocol, which consists of the method _iter_() and _next_() - The _iter_() method initialize the interator object and return it -The _next_() method returns the next value from the iterator . If there are no more items it raise a stop iteration exception

In [4]:
```
#Using this method
my_list = [2, 4, 6, 8, 10, 12, 14,16, 18, 20]
#get iterator object from the list
iterator = iter(my_list)

#use for loop to get the first five element
for __ in range(5):
    #get the next item from the iterator
    element = next(iterator)
    print(element)
```

```
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.

In [5]:
```
#using generator function
def fibonacci(n):
    a,b=0,1
    count = 0
    while count < n:
        yield a
        a , b = b,a+b
        count +=1
#using the generator function
fibo=fibonacci(10)

#printing the fibonnacci numbers using the generator
for num in fibo:
    print(num)
```

```
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 [6]:
```python
def prime_generator(limit=1000):
    """Generator function to yield prime numbers less than a given limit."""
    def is_prime(n):
        """Check if a number is prime."""
        if n <= 1:
            return False
        for i in range(2, int(n**0.5) + 1):
            if n % i == 0:
                return False
        return True

    for num in range(2, limit):
        if is_prime(num):
            yield num

# Create a prime number generator
primes = prime_generator()

# Print the first 20 prime numbers using the next() method
for _ in range(20):
    print(next(primes))
```

```
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 [7]:
```python
#According to question , Here function to print the 10 first fibonacci numbers
def print_fibonacci_numbers(count):
    a,b=0,1
    index = 0

    while index < count:
        print(a)
        a,b = b,a+b

        index +=1

#print the first 10 fibonacci numbers
print_fibonacci_numbers(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']

In [8]:
```
#Here given string
input_string = "pwskills"

#Now list comprehension to iterate through the string and create the expected ou
output_list = [char for char in input_string]

print(output_list)
```

```
['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.

In [9]:
```
#create the function
def is_palindrome(numbers):

   #Store the original number
    original_numbers = numbers

   #initialize a variable to hold the reversed number
    reversed_number = 0
    #Reverse the number using the while loop
    while numbers > 0:
        digit = numbers % 10
        reversed_number = reversed_number * 10 + digit
        numbers = numbers//10

   #check if the original number is equal to the reversed number
        return original_numbers == reversed_number
#Test the function with a number
        test_number = int(input("Enter a number to check if it is a palindrome"))
        if is_palindrome(test_number):
                            print(f"{test_number} is a palindrome")
        else:
                            print(f"{test_number} is not a palindrome")
```

Q9. Write a code to print odd numbers from 1 to 100 using list comprehension.Note: Use a list comprehension to create a list from 1 to 100 and use another List comprehension to filterout odd numbers.

In [10]:
```
# creat a list of numbers from 1 to 100
numbers = [i for i in range(1,101)]

#Filter out the odd numbers from the list
odd_numbers = [num for num in numbers if num % 2!=0]

#print the list of odd numbers
print(odd_numbers)
```

```
[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, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81,
83, 85, 87, 89, 91, 93, 95, 97, 99]
```

In [ ]: