# 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]:

```
def get_odd_numbers():
    return [num for num in range(1, 26) if num % 2 != 0]

# Test the function
odd_numbers_list = get_odd_numbers()
print(odd_numbers_list)
```

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

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

In [2]:

```
def example_args(*args):
    for arg in args:
        print(arg)

def example_kwargs(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

# Test the functions
example_args(1, "apple", True)
example_kwargs(name="John", age=25, city="New York")
```

```
1
apple
True
name: John
age: 25
city: New York
```

# What is an iterator in Python? Name the method used to initialize 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 [3]:

```
my_list = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
my_iterator = iter(my_list)

for _ in range(5):
    print(next(my_iterator))
```

```
2
4
6
8
10
```

# What is a generator function in Python? Why is the yield keyword used? Give an example of a generator function.

In [4]:

```
def generate_squares(n):
    for i in range(n):
        yield i ** 2

# Example usage
squares_generator = generate_squares(5)
for square in squares_generator:
    print(square)
```

```
0
1
4
9
16
```

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

In [5]:

```
def generate_primes():
    count = 0
    number = 2

    while count < 20:
        for i in range(2, int(number**0.5) + 1):
            if number % i == 0:
                break
        else:
            yield number
            count += 1
        number += 1

# Example usage
prime_generator = generate_primes()
for _ in range(20):
    print(next(prime_generator))
```

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

# Write a Python program to print the first 10 Fibonacci numbers using a while loop.

In [6]:

```
a, b = 0, 1
count = 0

while count < 10:
    print(a, end=" ")
    a, b = b, a + b
    count += 1
```

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

# Write a List Comprehension to iterate through the given string: ‘pwskills’.

In [7]:

```
string = 'pwskills'
result = [char for char in string]
print(result)
```

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

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

In [8]:

```
def is_palindrome(number):
    original_number = number
    reverse_number = 0

    while number > 0:
        digit = number % 10
        reverse_number = reverse_number * 10 + digit
        number = number // 10

    return original_number == reverse_number

# Test the function
print(is_palindrome(121))
```

```
True
```

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

In [9]:

```
odd_numbers = [num for num in range(1, 101) if num % 2 != 0]
print(odd_numbers)
```

```
[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, 7
5, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99]
```

In [ ]: