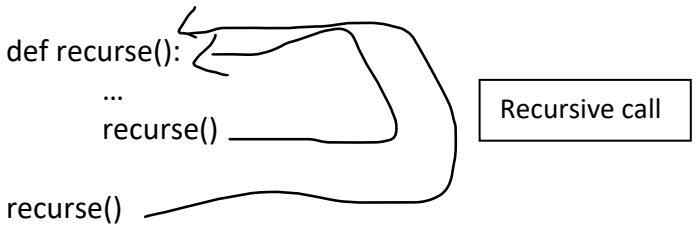
Function Recursion:

1. Recursion is the process of defining something in terms of itself.
2. When a function calls itself, it is known as recursion.
3. A physical world example would be to place two parallel mirrors facing each other. Any object in between them would be reflected recursively.



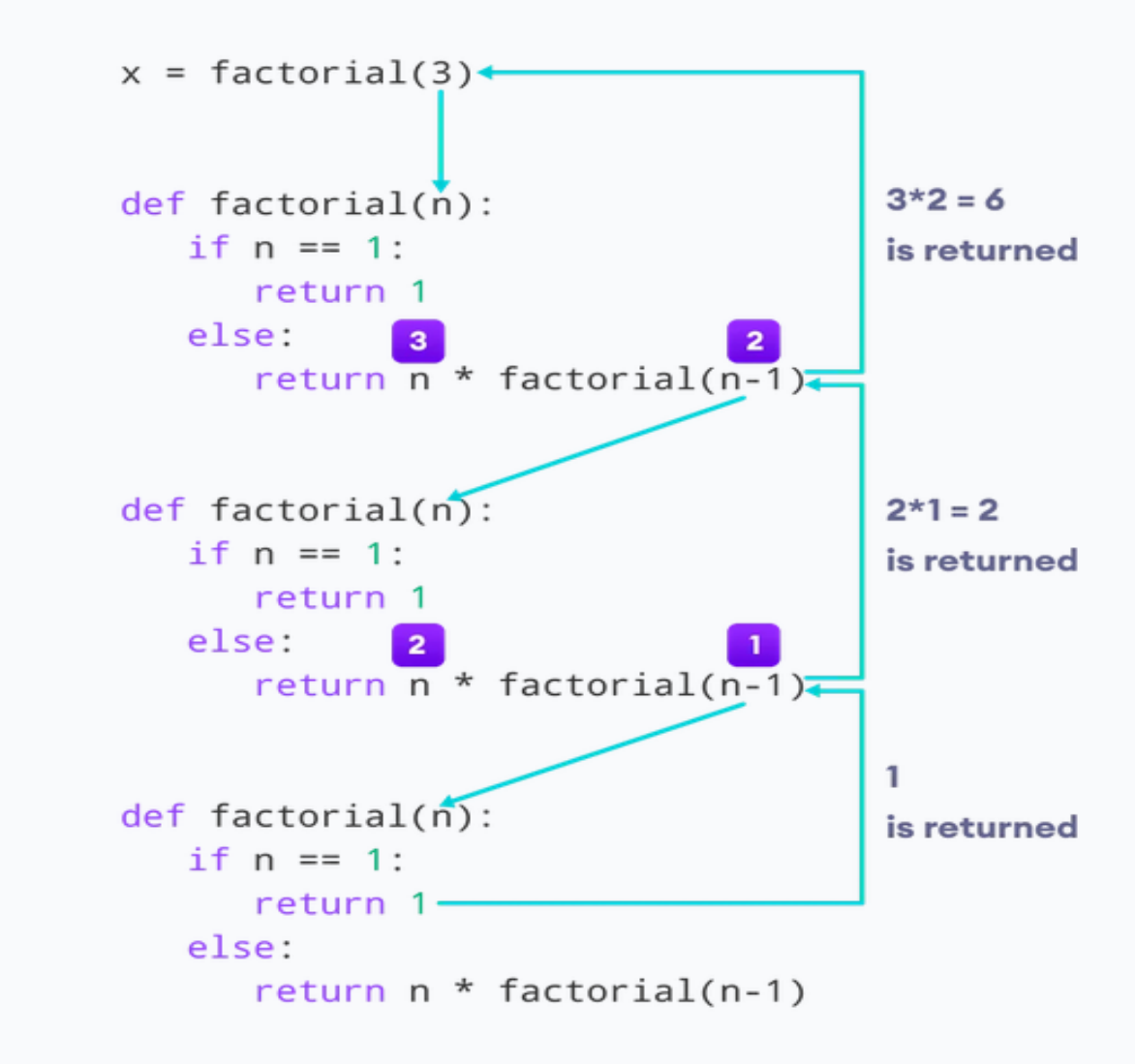
Example of recursive function (Program of Factorial):

```
def factorial(x):
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))


num = 3
print("The factorial of", num, "is", factorial(num))
```

Recursive call:

```
factorial(3)          # 1st call with 3
3 * factorial(2)      # 2nd call with 2
3 * 2 * factorial(1)  # 3rd call with 1
3 * 2 * 1             # return from 3rd call as number=1
3 * 2                 # return from 2nd call
6                     # return from 1st call
```

Working:



Advantages:

1. Recursive functions make the code look clean and elegant.
2. A complex task can be broken down into simpler sub-problems using recursion.
3. Sequence generation is easier with recursion than using some nested iteration.

Disadvantages:

1. Sometimes the logic behind recursion is hard to follow through.
2. Recursive calls are expensive (inefficient) as they take up a lot of memory and time.
3. Recursive functions are hard to debug.

Tail Recursion:

1. A unique type of recursion where the last procedure of a function is a recursive call.
2. The recursion may be automated away by performing the request in the current stack frame and returning the output instead of generating a new stack frame.
3. The tail-recursion may be optimized by the compiler which makes it better than non-tail recursive functions.

Problems:

1. Consider the following recursive function. What is f(0)?

```
def f(x):
    if x > 1000:
        return x - 4
    else:
        return f(f(x+5))
```

Ans: 997

2. Consider the following recursive functions

```python
def f1(n):
    if n == 0:
        return 0
    return f1(n-1) + 2*n - 1

def f2(n):
    if n == 0:
        return 0
    return f2(n-1) + 3*(f1(n)) - 3*n + 1
```

What is the value of f1(5)? f2(5)? F2(123)?

Ans: 25
125
1860867

3. Consider the following function

```python
def mystery(n):
    if n == 0 or n == 1:
        return
    mystery(n-2)
    print(n)
    mystery(n-1)
```

What does mystery(6) print out?

Ans: 2
4
3
2
6
3
2
5
2
4
3
2

4. What will be the output of the following codes?

```
def f(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    if n == 2:
        return 1

    return 2*f(n-2) + f(n-3)
```

Ans: Fibonacci Sequence

5. Consider the following function. What does *mystery(0, 8)* do?

```
def mystery(a, b):
    if a != b:
        m = (a + b)/2
        mystery(a, m)
        print(m)
        mystery(m, b)
```

Ans: Infinite Loop

6. A 5-digit positive integer is entered through the keyboard, write a function to calculate sum of digits of the 5-digit number using recursion.

```
def recGetSum(num):
    sum = 0
    if num == 0:
        return sum
    sum = num%10+recGetSum(num//10)
    return sum

print("Enter a five digit number : ")
num=int(input())
print("Sum: ",recGetSum(num))
```

7. A positive integer is entered through the keyboard. Write a recursive function to obtain the prime factors of this number.

```
def printPrimeFactors(num):
    x=2
```

```
    while x<=num:
        if num%x==0:
            print(x)
            printPrimeFactors(num/x)
            break
        x+=1

number=int(input("Enter a number: "))
print("Prime factors are: ")
printPrimeFactors(number)
```

8. Write a recursive function to obtain the first 25 numbers of a Fibonacci sequence.

```
def getFibonacci(num):
    if num==0:
        return 0
    elif num==1:
        return 1
    else:
        return getFibonacci(num-1)+getFibonacci(num-2)

print("Fibonacci Series: ")
c=0
for i in range(1,26):
    print(getFibonacci(c))
    c+=1
```

9. A positive integer is entered through the keyboard, write a function to find the binary equivalent of this number using recursion.

```
def decToBin(num):
    if num == 0:
        return
    decToBin(num//2)
    print(num%2,end="")

number=int(input("Enter the decimal number: "))
decToBin(number)
```

10. Write a recursive function to obtain the running sum of first 25 natural numbers.

```
def getSum(num):
    if num == 0:
        return num
    num = num + getSum(num - 1)
    return num

print(getSum(25))
```

11. Write a recursive function to find the sum of all even numbers in a given range starting from 0.

```
def getSum(num):
    if num == 0:
        return num
    if num%2==0:
        num = num + getSum(num - 2)
    else:
        getSum(num-1)
    return num

print(getSum(10))
```

12. Write a program to print all elements of a list using recursion.

```
import random
def printListRec(randomList, start, len):
    if start >= len:
        return

    print(randomList[start])
    printListRec(randomList, start + 1, len)

randomList=random.sample(range(10,30),5)
print(randomList)
printListRec(randomList,0,len(randomList))
```

13. Write a program having two different functions to find maximum and minimum elements in a list using recursion.

```
import random
def getMaximum(randomList, index, length):
    # max
    if index >= length-2:
        if randomList[index] > randomList[index + 1]:
            return randomList[index]
        else:
            return randomList[index + 1]

    max = getMaximum(randomList, index + 1, length)

    if randomList[index] > max:
        return randomList[index]
    else:
        return max

def getMinimum(randomList, index, length):
    if index >= length-2:
        if randomList[index] < randomList[index + 1]:
            return randomList[index]
        else:
            return randomList[index + 1]

    min = getMinimum(randomList, index + 1, length)

    if randomList[index] < min:
        return randomList[index]
    else:
        return min

randomList=random.sample(range(10,30),5)
print(randomList)
print("Maximum: ",getMaximum(randomList,0,len(randomList)))
print("Minimum: ", getMinimum(randomList,0,len(randomList)))
```

14. Write a function that takes in two numbers and recursively multiplies them together.

```
def getProductsRec(firstNumber, secondNumber, i):
    if i<secondNumber:
        return firstNumber+getProductsRec(firstNumber,secondNumber,i+1)
```

```
    else:
        return 0


firstNumber=int(input("Enter first number: "))
secondNumber=int(input("Enter second number: "))
print("Multiplication result:
",getProductsRec(firstNumber,secondNumber,0))
```

15. Write a function using recursion that takes in a string and returns a reversed copy of the string.

```
def reverseString(str):
    if len(str.strip())==0:
        print("String is empty.")
        return str
    else:
        return reverseString(str[1:])+str[0]

str=input("Enter a string: ")
print("Reverse: ",reverseString(str))
```