$f: A \to B$ | Set = Unordered collection of elements without repetition.

$f: A \to B$: It is rule set which pairs all elements in A with exactly one element B.

A = Domain of function.

B = Co-domain of function.

$R_f = \{ y$: there is x in A with which y is paired $\}$

$f: A \to B$

$f(n) = n^2 + 1.$

$\underline{n \in A.}$ $\quad f(n) = n^2+1 \in B.$

$$f(n) = \begin{cases} n^2 - 1 & \text{if } n \text{ is even.} \\ n^2 + 1 & \text{if } n \text{ is odd} \end{cases}$$

$f(1) = 1^2+1 \qquad f(3) = 3^2+1$

$f(2) = 2^2-1 \qquad f(4) = 4^2-1$

$$f(n) = \begin{cases} n \cdot f(n-1) & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

$f(5) = 5 \cdot f(4)$ $\quad$ $f(5) = 5 \cdot 24 = 120$

$f(4) = 4 \cdot f(3)$ $\quad$ $f(4) = 4 \cdot 6 = 24$

$f(3) = 3 \cdot f(2)$ $\quad$ $f(3) = 3 \cdot 2 = 6$

$f(2) = 2 \cdot f(1)$ $\quad$ $f(2) = 2 \cdot 1 = 2$

$f(1) = 1 \cdot f(0) = 1$ $\quad$ $f(1) = 1$

```
def f(n: int) -> int:
    if n == 0:
        return (1)
    return n * f(n-1)
```

$\equiv$

$$f(n) = \begin{cases} n \cdot f(n-1) & \dots \text{if } n > 0 \\ 1 & \dots \text{if } n = 0 \end{cases}$$

$f(5)$

Stack trace

$f(5) = 5 \cdot f(4)$

$f(4) = 4 \cdot f(3)$

$f(3) = 3 \cdot f(2)$

$f(2) = 2 \cdot f(1)$

$f(1) = 1 \cdot f(0) = 1$

$f(5)$

$f(5) = 5 \cdot 24 = 120$

$f(4) = 4 \cdot 6 = 24$

$f(3) = 3 \cdot 2 = 6$

$f(2) = 2 \cdot 1 = 2$

$f(1) = 1$

Printing list: let L be a list of ints.

```
def prn_iter(L: [int]) -> None:
    for i in range(len(L)):
        print(i, L[i])
```

$$prn(L, N, i) = \begin{cases} \text{print}(i, L[i]) \\ prn(L, N, i+1) & \text{if } i < N. \\ \\ \text{STOP/RETURN} & \text{if } i == N. \end{cases}$$

Trigger Call: prn(L, len(L), 0)

L = [10, 20, 30, 40, 50] (indices 0, 1, 2, 3, 4)

prn(L, 5, 0): 0<5: print(0, L[0]); prn(L, 5, 0+)
prn(L, 5, 1): 1<5: print(1, L[1]); prn(L, 5, 2)
prn(L, 5, 2): 2<5: print(2, L[2]); prn(L, 5, 3)
prn(L, 5, 3): 3<5: print(3, L[3]); prn(L, 5, 4)
prn(L, 5, 4): 4<5: print(4, L[4]); prn(L, 5, 5)
prn(L, 5, 5) 5<5: False
Return

0, 10
1, 20
2, 30
3, 40
4, 50

```
def f5():
    print(" I am at the last pos ")

def f4():
    print(" In func1 ")
    f5()

def f3():
    print (" In f3() ")
    f4()

def f2():
    print (" In f2() ")
    f3()

def f1():
    print (" In f1() ")
    f2()

f1()
```

```
def f():
    f()
```

Verb = Action = function.

$$prn(L, i) \equiv \begin{cases} print(i, L[i]) \\ prn(L, i+1) & \text{if } i < len(L) \\ \text{Return} & \text{if } i = len(L) \end{cases}$$

$$prn(L, N, i) = \begin{cases} print(i, L[i]) \\ prn(L, N, i+1) & \text{if } i < len(L) \\ \text{Return} & \text{if } i == N \end{cases}$$

L = [—, —, —, —, —, —]

0 1 2 3 4 5

```
def sum_list(L: [int]) -> int:
    S = 0
    for i in range(len(L)):
        S = S + L[i]
    return(S)
```

L = [10, 20, 30, 40, 50, 60]

Sum([10,20,30,40,50,60])
= 10 + Sum([20,30,40,50,60])
= 20 + Sum([30,40,50,60])
= 30 + Sum([40,50,60])
= 40 + Sum([50,60])
= 50 + Sum([60])
60 + Sum([])
0

Sum(L, N, i) =

return L[i] + Sum(L, N, i+1) if i < N

return (0) if i == N

```
def Sum(L:[int], N:int, i: int) -> int:
    if i==N:
        return (0)
    return L[i]+Sum(L, N, i+1)
S = Sum(L, len(L), 0)
```

Study:

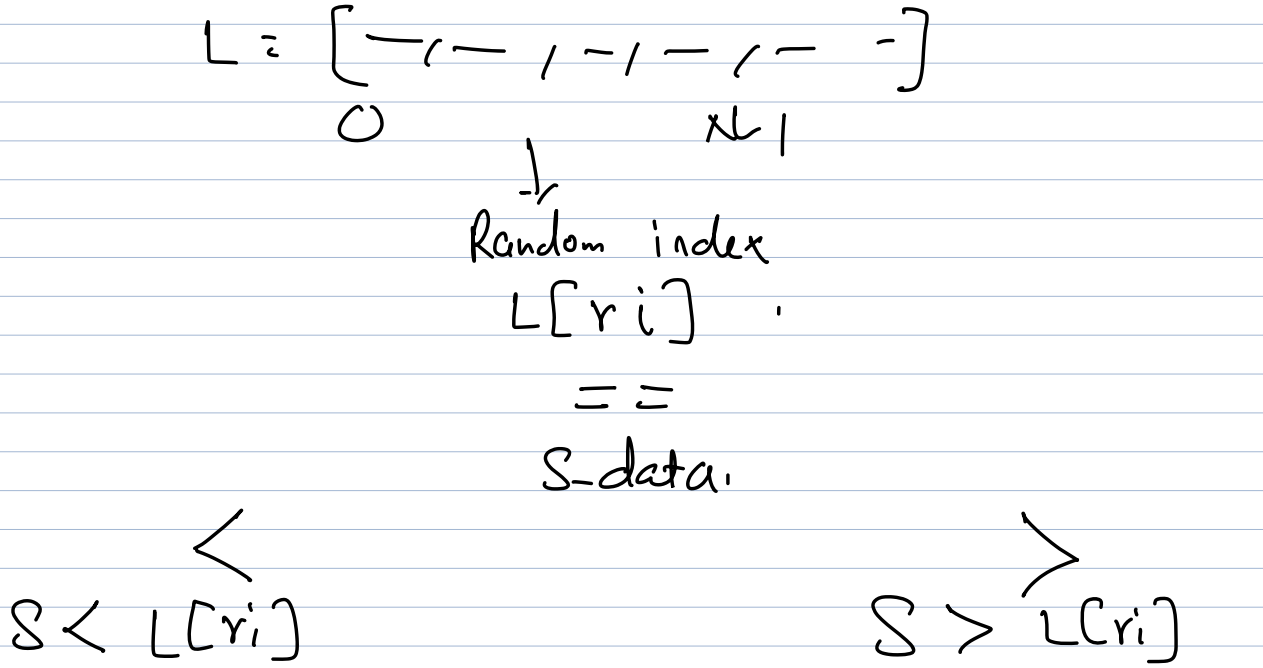
1) factorial:
   Math notation ⟶ Code

2) Printing list in forward order.
   Math notation ⟶ Code.

3) Printing list in backward order
   Math notation ⟶ Code.

4) Linear Search recursive
   Math notation ⟶ code.

5) Sum the list
   Math notation ⟶ code.



$L = [x_0, x_1, x_2, \ldots, x_{N-1}]$

mid = (0 + N-1) / 2

L[mid] == S_data

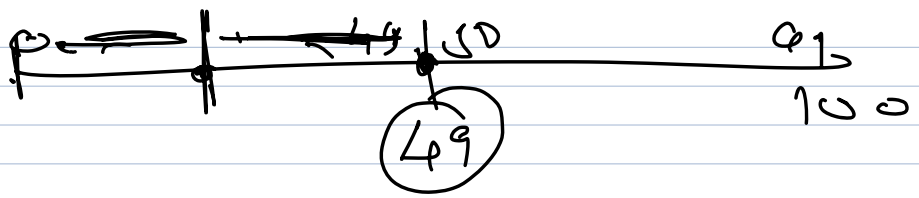
S < L[mid] | mid — N-1 eliminate

Search in

L[0 ... mid-1]

S > L[mid] | 0 ... mid eliminate

L[mid+1 .... N-1]



100 50 25 12 6 3 1 0

bin_search(L_{sorted}, s_data, s_ind, e_ind)

= return False <----------> if s_ind $\geq$ e_ind

mid $\leftarrow$ $\frac{(s\_ind + e\_ind)}{2}$

return True <----------> if L_{sorted}[mid] = s_data

return bin_search(L_{sorted}, s_data, s_ind, mid-1)

----------> if s_data < L_{sorted}[mid]

return bin_search(L_{sorted}, s_data, mid+1, e_ind)

----------> if s_data > L_{sorted}[mid]