```
def f_1(n: int):
    for i in range(10):          0..9
        for j in range(n):       0..n-1
            print("Hello!")      1 step ⇒ Θ(1) (constant complexity)
```

$T(n) = 10 * n * 1 = 10n \in O(n)$
$\in \Theta(n)$

```
def f_2(n: int):
    for i in range(n, n+10):     n, n+1, ..., n+9 ⇒ 10 (regardless of n)
        for j in range(n):       0..n-1
            print("Hello!")      1
```

$T(n) = 10 * n * 1 = 10n \in \Theta(n)$

```
def f_3(n: int):
    for i in range(1,n):
        for j in range(i,n):
            print("Hello!")
```

I $i = 1, \ldots, n-2, n-1$
$j = i, i+1, \ldots n-1$

	j	STEPS
$i=1$	1 2 ... $n-1$	$n-1$
$i=2$	2 ... $n-1$	$n-2$
$i=3$	3 ... $n-1$	$n-3$
⋮		
$i=n-2$	$n-2$ $n-1$	2
$i=n-1$	$n-1$	1
		+
		$=T(n)$

$T(n) = 1+2+\ldots+n-1 = \frac{(n-1)\cdot n}{2}$

$= \frac{n^2-n}{2} = \underbrace{\frac{1}{2}n^2} - \frac{1}{2}n$

dominant term, discard constant

$T(n) \in O(n^2), \Theta(n^2)$

II $T(n) = \sum_{i=1}^{n-1} \sum_{j=i}^{n-1} 1$

$\sum_{j=i}^{n-1} 1 = \underbrace{1+1+1+1+\ldots+1+1}_{n-1-i+1 \text{ times}}$ $\Rightarrow$ $\underbrace{i \; i+1 \ldots n-1}_{n-i \text{ times}}$

how many terms?

e.g $i=5$, $n=10$: $\underbrace{5\,6\,7\,8\,9}_{9-5+1}$ (i … $n-1$)

$$\Rightarrow T(n) = \sum_{i=1}^{n-1} n-i = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i$$

$$= n \cdot \sum_{i=1}^{n-1} 1 - \sum_{i=1}^{n-1} i$$

$$= n \cdot (n-1) - (1+2+\ldots+n-1)$$

$$= n(n-1) - \frac{(n-1)\cdot n}{2}$$

$$= \frac{n(n-1)}{2} \in \Theta(n^2)$$

```
def f_4(n: int):
    for i in range(n):
        for j in range(2*i+1):
            print("Hello!")
```

i: 0..n-1; j: 0..2i

	j	STEPS
I $i=0$	0	1
$i=1$	0,1,2	3
$i=2$	0,1,2,3,4	5
$i=3$	0,1,2,3,4,5,6	7
$i=n-1$	0,1,…,2n-2	2n-1

\+

T(n)

$$T(n) = 1+3+5+\ldots+2n-1 = 1+2+3+\ldots+2n-2+2n-1 - 2(1+2+\ldots+n-1)$$

$$= \frac{(2n-1)\cdot 2n}{2} - 2\cdot\frac{(n-1)\cdot n}{2} = n(2n-1) - n(n-1)$$

$$= n(2n-1-n+1)$$

II $T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{2i} 1$

$\sum_{j=0}^{2i} = 1 = \underbrace{1+1+1+\ldots+1}_{2i+1} = 2i+1$

$T(n) = \sum_{i=0}^{n-1} 2i+1 = 2\sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1$

$= 2 \cdot (0+1+\ldots+n-1) + \underbrace{(1+1+1+\ldots+1)}_{n}$

$= \cancel{2} \cdot \frac{(n-1)\cdot n}{\cancel{2}} + n$

$= n(n-\cancel{1}+\cancel{1})$

$= n^2 \in \Theta(n^2)$

```
def f_5(n: int):
    for i in range(n**2):        0..n^2-1
        for j in range(i):       0...i-1
            print("Hello!")      1
```

I $\sum_{i=0}^{n^2-1} \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n^2-1} i = 0+1+\ldots+n^2-1$

$= \frac{(n^2-1)\cdot n^2}{2}$

$= \frac{n^4-n^2}{4} \in O(n^4)$

$\Theta(n^4)$

$1^2+2^2+\ldots+n^2 = \frac{n(n+1)(2n+1)}{6}$

II

	j	STEPS
$i=0$	–	0
$i=1$	0	1
$i=2$	0,1	2
$i=3$	0,1,2	3
$i=n^2-1$	$0,\ldots,n^2-2$	n^2-1

$T(n) = 0+1+\ldots+n^2-1$

$= \frac{(n^2-1)\cdot n^2}{2}$

$= \ldots$

```
def f_9(n: int):
    for i in range(n):
        for j in range(n):
            print("Hello World!")
        for k in range(n):
            print("Hello world!")
```

$$T(n) = \sum_{i=0}^{n-1}\left(\underbrace{\sum_{j=0}^{n-1} 1}_{n} + \underbrace{\sum_{k=0}^{n-1} 1}_{n}\right) = \sum_{i=0}^{n-1} 2n = 2n\underbrace{\sum_{i=0}^{n-1} 1}_{\text{does not depend on } i} = 2n \cdot n = 2n^2 \in \Theta(n^2)$$

```
def f_13(data: list):
    if len(data) == 0:
        return 0
    if len(data) == 1:
        return data[0]
    m = len(data)//2
    return f_13(data[:m]) + f_13(data[m:])
```

$$T(n) = \begin{cases} 1 \text{ if } n = len(data) <= 1 \\ 2 \cdot T(n/2) + 1 \end{cases}$$

TIME COMPLEXITY

$T(n) = 2 \cdot T(n/2) + 1$

$T(n/2) = 2 \cdot T(n/4) + 1$

$T(n/4) = 2 \cdot T(n/8) + 1$

$\vdots$

$T(2) = 2T(1) + 1$

$T(1) = 1$

$T(n) = 2 \cdot [2 \cdot T(n/4) + 1] + 1$

$= 2^2 T(n/4) + 2 + 1$

$= 2^2 [2T(n/8) + 1] + 2 + 1$

$= 2^3 T(n/8) + 4 + 2 + 1$

$= \dots$

Let k = number of times list is split in halves $\Rightarrow n = 2^k$

$k = \log_2 n$

$\Rightarrow T(n) = 2^k \cdot T(1) + 2^{k-1} + 2^{k-2} + \ldots + 2^2 + 2 + 1$

$= 2^k + 2^{k-1} + \ldots + 2^2 + 2 + 1$

$= 2^{k+1} - 1$

$= 2^k \cdot 2 - 1$

$= 2n - 1 \in \Theta(n)$

EXTRA SPACE COMPLEXITY

$$T(n) = \begin{cases} 1, & n \leq 1 \\ 2T(n/2) + n & \text{otherwise} \end{cases}$$

! SLICING

→ repeat the same process as for time complexity

$T(n) = 2 \cdot T(n/2) + n$

$T(n/2) = 2T(n/4) + n/2$

$T(n/4) = 2T(n/8) + n/4$

$T(2) = 2T(1) + 2$

$T(1) = 1$

$T(n) = 2 \cdot T(n/2) + n$

$= 2[2T(n/4) + n/2] + n$

$= 2^2 T(n/4) + n + n$

$= 2^2 [2T(n/8) + n/4] + n + n$

$= 2^3 T(n/8) + n + n + n$

$= \ldots$

$= 2^k T(1) + k \cdot n$

$= n + n \log n$

$= (n+1) \log n \in \Theta(n \log n)$

```
def recursive_f_1(n: int):
    if n<=0:
        return 1
    else:
        return 1+ recursive_f_1(n-1)
```

$$T(n) = \begin{cases} 1 \text{ if } n \leq 0 \\ T(n-1)+1 \text{ otherwise} \end{cases}$$

$T(n) = T(n-1)+1$

$T(n-1) = T(n-2)+1$

$T(n-2) = T(n-3)+1$

$\vdots$

$T(2) = T(1)+1$

$T(1) = T(0)+1$

$T(0) = 1$

$T(n) = T(n-1)+1$

$= T(n-2)+1+1$

$= T(n-3)+1+1+1$

$= \dots$

$= \underbrace{1+1+1+\dots+1}_{n+1}$

$= n+1 \in \Theta(n)$

```
def recursive_f_2(n: int)
    if n<=1:
        return 1
    else:
        return 1+recursive_f_2(n-5)
```

$$T(n) = \begin{cases} 1 \text{ if } n <= 1 \\ T(n-5)+1 \text{ otherwise} \end{cases}$$

$T(n) = T(n-5)+1$

$T(n-5) = T(n-10)+1$

$\vdots$

$T(k) = 1, \; k \leq 1$

$n/5$ (how many times can I subtract 5 from my number?)

$T(n) = 1 + T(n-5)$

$= 1 + 1 + T(n-10)$

$= n/5 \in \Theta(n)$

```
def recursive_f3(n: int)
    if n<=1:
      return 0
    else
       return 1+recursive_f3(n/2)
```

$$T(n) = \begin{cases} 1 \text{ if } n \leq 1 \\ T(n/2)+1 \text{ otherwise} \end{cases}$$

$\rightarrow T(n) = T(n/2) + 1$

$T(n/2) = T(n/4) + 1$

....

$T(k) = 1, \; k \leq 1$

$T(n) = T(n/2) + 1$

$= T(n/4) + 1 + 1$

$= T(n/8) + 1 + 1 + 1$

$= \ldots$

RESULT: $T(n) \in \Theta(\log n)$

```
def recursive_f4(n: int, m: int, o: int):
    if n<=0:
       print(n, m, o)
    else:
       recursive_f4(n-1, m+1, o)
       recursive_f4(n-1, m, o+1)
```

$$T(n) = \begin{cases} 1 \text{ if } n <= 0 \\ 2T(n-1)+1 \text{ otherwise} \end{cases}$$

$T(n) = 2T(n-1) + 1$

$T(n-1) = 2T(n-2) + 1$

$T(n-2) = 2T(n-3) + 1$

$\vdots$

$T(n) = 2T(1) + 1$

$T(1) = 2 \cdot T(0) + 1$

$T(0) = 1$

then: $T(n) = 2T(n-1) + 1$

$= 2 \cdot [2T(n-2) + 1] + 1$

$= 2^2 \cdot T(n-2) + 2 + 1$

$= 2^2 [2T(n-3) + 1] + 2 + 1$

$= 2^3 T(n-3) + 2^2 + 2 + 1$

$= \dots$

$= 2^{n-1} T(1) + 2^{n-2} + 2^{n-3} + \dots + 1$

$= 2^n T(0) + 2^{n-1} + \dots + 2 + 1$

$= 1 + 2 + \dots + 2^n$

$= 2^{n+1} - 1 \in \Theta(2^n)$

```
def recursive_f5(n):
    print(n)
    for i in range(n):
        print("Hello!")
    if n<1:
        return 1
    else:
        return 1+recursive_f_5(n-1)
```

$$T(n) = \begin{cases} 1 \text{ if } n < 1 \\ T(n-1) + n \text{ otherwise} \end{cases}$$

$T(n) = T(n-1) + n$

$T(n-1) = T(n-2) + n - 1$

$\vdots$

$T(2) = T(1) + 2$

$T(1) = T(0) + 1$

$T(0) = 1$

then $T(n) = T(n-1) + n$

$= T(n-2) + n-1 + n$

$= T(n-3) + n-2 + n-1 + n$

$= \ldots$

$= T(2) + 3 + 4 + \ldots + n$

$= T(1) + 2 + 3 + 4 + \ldots + n$

$= T(0) + 1 + 2 + 3 + \ldots + n$

$= 1 + 1 + 2 + 3 + \ldots + n$

$= 1 + \frac{n(n+1)}{2}$

$= \frac{2 + n^2 + n}{2} \quad \in \Theta(n^2)$