

	S	M	T	W	T	F	S
APR 2024	1	2	3	4	5	6	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
28	29	30					

## Recursion

MARCH

09

SATURDAY

# def fun1():

    Print('fun1() called')

08     def fun2():     Print('Before fun2()')

    def fun2():     Print('Before fun2()')

09     Print('Before fun1()')

    fun1()

10     Print('After fun1()')

    After fun1()

    Print('Before fun2()')

11     fun2()

    Print('After fun2()')

12

# Many algorithms technique are based on Recursion.

~~Application~~ → Dynamic Programming

eg. Fibonacci Series, factorial

→ Backtracking

→ Divide & Conquer (Binary search, Quick sort and merge sort)

Many problem inherently recursive -

→ Tower of Hanoi

→ DFS based traversal (DFS of Graph and Inorder/Preorder/Postorder traversal of tree)

# factorial

def f(n):

    if n == 0:     Print("Calling f(0) if n in [0,1]")

    return 1     Print("at start of f(0) return 1")

    if n > 0:     Print("Calling f(n) if n > 0")

# fibonacci

def fib(n):

    if n in [0,1]:

        return n

    else:

        return f(n-1) + f(n-2)

→ Recursion = Terminates when the base case becomes true.

    ↳ Every recursive call needs extra spaces in stack memory.

Iteration = Terminates when the condition becomes false.

    ↳ Every iterations doesn't require any extra spaces.

Success has many fathers, while failure is an orphan

MARCH

10

↳ Function which calls itself again &amp; again ↫

'24 - until a specific condition met.

11th Week  
070-296

eg

Inception movie example of

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MAR 2024

SUNDAY

- Recursion works on the principle of mathematical Induction.
- Performing the same operation multiple times with different inputs.
- In every step we try smaller input to make the problem smaller.
- Base condition is needed to stop the recursion, otherwise infinite loop will occur.
- Recursive thinking is really important in programming and it helps you break down big problem into smaller ones and easier to use.

11

	Iteration	Recursion
12 Time efficient	✓	✗
Space efficient	✗	✗
01 easy to code	✗	✓

# There are two types of recursion:-

① Finite Recursion

② Infinite Recursion

03 Recursive method is going to execute finite times, it stops

Recursive method is going to execute infinite times, never ending process but python terminates this kind of recursion automatically if stack got overflow.

04 automatically once if condition is satisfied such type of recursion

05 is called finite recursions

#06 Base Condition

Inside a recursion its stop recursion process at a finite nos of steps, we

07 can use base conditions. We have to take this base conditions as first statement inside recursion call, if the condition is satisfied it terminates else continue the execution flow.

eg def fun(n):

if n==0: base condition

return

Print('Good morning')

fun(1)

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

→ Infinite recursion may lead to running out of stack memory. And result in stack overflow.

MARCH  
24  
11th Week  
07/295

# Based on function calls/ method calls we have two types of recursion. MONDAY

① Direct recursion

08 def fun():  
= 09 =  
fun()  
10

② Indirect recursion.

def fun(): → def f1():  
= {  
def f2(): } =  
def f2(): def f1():  
= {  
def f1(): } =

# When we go for recursion?

→ If the problem is very big and we are unable to solve the problem then we should go for recursion.

# Based on position of recursive statement again it is divided into two types.

① Tail recursive (perfect recursion)

A recursive function is called as tail recursion if the function doesn't contain any statement after recursive function calls.

② Non-tail recursion (backtracking)

A recursive function is called as non-tail recursion if the function contains statement after recursive function calls.

eg def fun(n):

if n==0:

return

print(n)

fun(n-1)

o/p - 4 3 2 1

def fun(n):

if (n==0):

return

fun(n-1)

print(n)

fun(4)

o/p - 1 2 3 4

eg quick sort

post order tree traversal

post order is faster than other two traversals.

# To print on same line use end operator.

Print(n, end="") in

MARCH

12

24

11th Week  
072-294

S	M	T	W	T	F	S
31	1	2	3	4	5	6
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MAR 2024

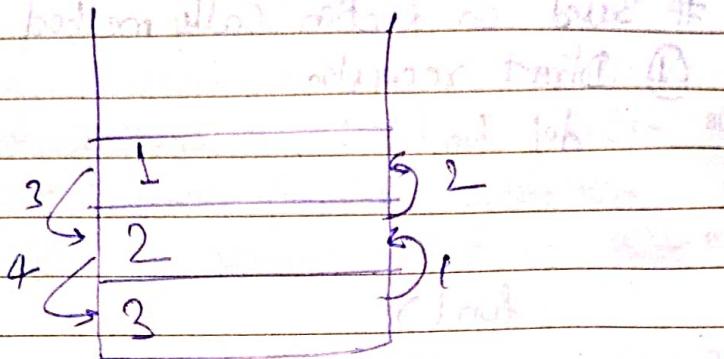
TUESDAY

eg def fun(n):

```

08 if n==0:
09     return
10    print(n)
11    fun(n-1)
12    print(n)

```



f(3)

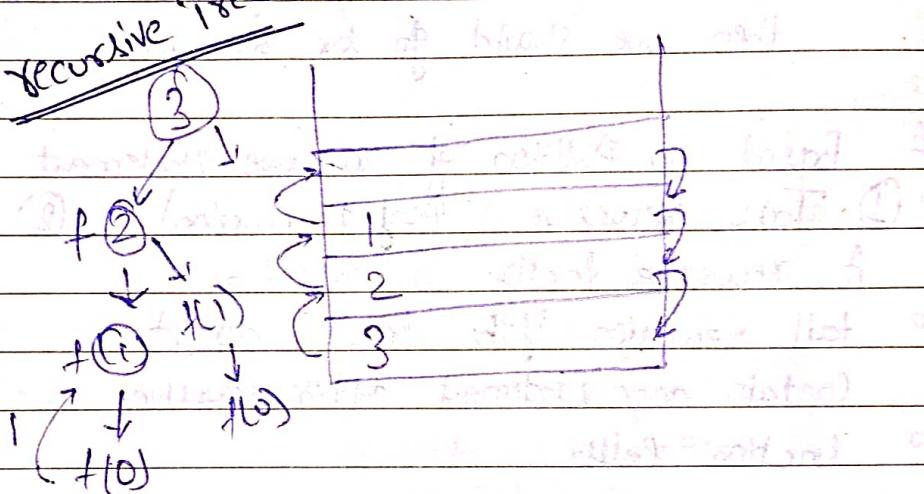
o/p - 0/0 - 3,2,1,1,2,3

eg2 def fun(n):

```

01 if n==0:
02     return
03    fun(n-1)
04    print(n)
05    fun(n-1)
06    print(n)
07    fun(n-1)
08    print(n)

```



o/p - 1,2,1,3,1,2,1

eg def fun(n):

```

05 if n<=1:
06     return 0

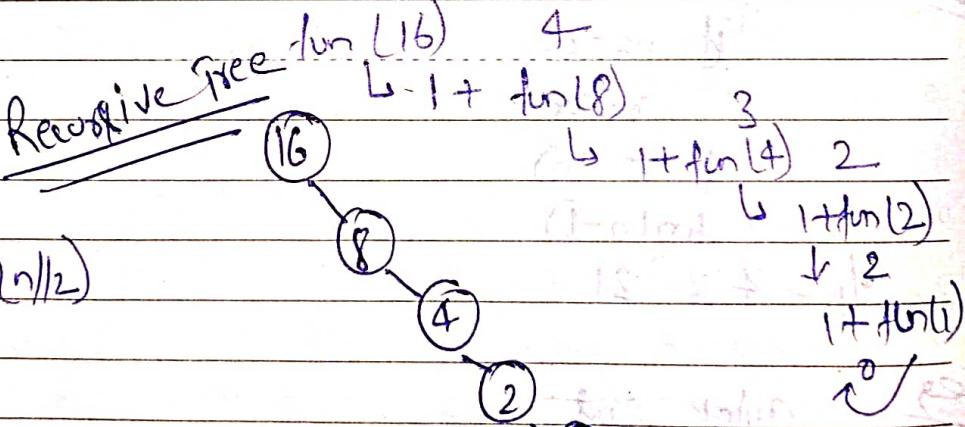
```

else:

return 1 + fun(n//2)

f(16) -

o/p - 4



eg print 1 to N using recursion.

def fun(n):

if n&gt;0:

return

fun(n-1)

print(n)

fun(4)

if n&gt;0:

fun(n-1)

print(n)

fun(4)

2

fun(3)

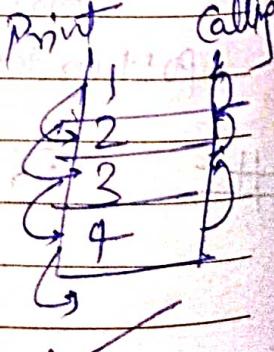
1

fun(2)

3

fun(1)

4



Sudden acquaintance brings repentance

<table border="1" style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th>S</th><th>M</th><th>T</th><th>W</th><th>T</th><th>F</th><th>S</th></tr> </thead> <tbody> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td></tr> <tr><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td></tr> <tr><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td></tr> <tr><td>29</td><td>30</td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>	S	M	T	W	T	F	S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30						दृष्टि बाल के लिए सोचना है   जाकि के रुद्र प्रियं <b>दृष्टि जोरी</b>	<b>MARCH</b> <b>24</b> 11th Week 073-293
S	M	T	W	T	F	S																																						
1	2	3	4	5	6	7																																						
8	9	10	11	12	13	14																																						
15	16	17	18	19	20	21																																						
22	23	24	25	26	27	28																																						
29	30																																											

eg def fun(n):

08 if n == 0:

09 return

(1) fun(8)

09 print(n%2)

(1) fun(13)

10 o/p - 1101

11 # Arbitrarily choose

eg print N to 1 using recursion.

01 def fun(n):

02 if n == 0:

03 return

04 print(n)

05 fun(n-1)

06 fun(4)

# sum of digit using recursion

01 def fun(n):

02 if n < 10:

03 return n

04 return fun(n//10) + n%10

05 fun(253)

06 fun(253)

07 # Palindrome

01 def isPalindrome(sts, start, end):

02 if start >= end:

03 return True

04 return sts[start] == sts[end] and isPalindrome(sts, start+1, end-1))

05 isPalindrome("123")

06 isPalindrome("12321")

07 isPalindrome("123456789")

08 isPalindrome("12345678987654321")

09 isPalindrome("12345678987654321")

10 isPalindrome("12345678987654321")

11 isPalindrome("12345678987654321")

12 isPalindrome("12345678987654321")

13 isPalindrome("12345678987654321")

14 isPalindrome("12345678987654321")

15 isPalindrome("12345678987654321")

16 isPalindrome("12345678987654321")

17 isPalindrome("12345678987654321")

18 isPalindrome("12345678987654321")

19 isPalindrome("12345678987654321")

20 isPalindrome("12345678987654321")

21 isPalindrome("12345678987654321")

22 isPalindrome("12345678987654321")

23 isPalindrome("12345678987654321")

24 isPalindrome("12345678987654321")

25 isPalindrome("12345678987654321")

26 isPalindrome("12345678987654321")

27 isPalindrome("12345678987654321")

28 isPalindrome("12345678987654321")

29 isPalindrome("12345678987654321")

30 isPalindrome("12345678987654321")

31 isPalindrome("12345678987654321")

32 isPalindrome("12345678987654321")

33 isPalindrome("12345678987654321")

34 isPalindrome("12345678987654321")

35 isPalindrome("12345678987654321")

36 isPalindrome("12345678987654321")

37 isPalindrome("12345678987654321")

38 isPalindrome("12345678987654321")

39 isPalindrome("12345678987654321")

40 isPalindrome("12345678987654321")

41 isPalindrome("12345678987654321")

42 isPalindrome("12345678987654321")

43 isPalindrome("12345678987654321")

44 isPalindrome("12345678987654321")

45 isPalindrome("12345678987654321")

46 isPalindrome("12345678987654321")

47 isPalindrome("12345678987654321")

48 isPalindrome("12345678987654321")

49 isPalindrome("12345678987654321")

50 isPalindrome("12345678987654321")

51 isPalindrome("12345678987654321")

52 isPalindrome("12345678987654321")

53 isPalindrome("12345678987654321")

54 isPalindrome("12345678987654321")

55 isPalindrome("12345678987654321")

56 isPalindrome("12345678987654321")

57 isPalindrome("12345678987654321")

58 isPalindrome("12345678987654321")

59 isPalindrome("12345678987654321")

60 isPalindrome("12345678987654321")

61 isPalindrome("12345678987654321")

62 isPalindrome("12345678987654321")

63 isPalindrome("12345678987654321")

64 isPalindrome("12345678987654321")

65 isPalindrome("12345678987654321")

66 isPalindrome("12345678987654321")

67 isPalindrome("12345678987654321")

68 isPalindrome("12345678987654321")

69 isPalindrome("12345678987654321")

70 isPalindrome("12345678987654321")

71 isPalindrome("12345678987654321")

72 isPalindrome("12345678987654321")

73 isPalindrome("12345678987654321")

74 isPalindrome("12345678987654321")

75 isPalindrome("12345678987654321")

76 isPalindrome("12345678987654321")

77 isPalindrome("12345678987654321")

78 isPalindrome("12345678987654321")

79 isPalindrome("12345678987654321")

80 isPalindrome("12345678987654321")

81 isPalindrome("12345678987654321")

82 isPalindrome("12345678987654321")

83 isPalindrome("12345678987654321")

84 isPalindrome("12345678987654321")

85 isPalindrome("12345678987654321")

86 isPalindrome("12345678987654321")

87 isPalindrome("12345678987654321")

88 isPalindrome("12345678987654321")

89 isPalindrome("12345678987654321")

90 isPalindrome("12345678987654321")

91 isPalindrome("12345678987654321")

92 isPalindrome("12345678987654321")

93 isPalindrome("12345678987654321")

94 isPalindrome("12345678987654321")

95 isPalindrome("12345678987654321")

96 isPalindrome("12345678987654321")

97 isPalindrome("12345678987654321")

98 isPalindrome("12345678987654321")

99 isPalindrome("12345678987654321")

100 isPalindrome("12345678987654321")

101 isPalindrome("12345678987654321")

102 isPalindrome("12345678987654321")

103 isPalindrome("12345678987654321")

104 isPalindrome("12345678987654321")

105 isPalindrome("12345678987654321")

106 isPalindrome("12345678987654321")

107 isPalindrome("12345678987654321")

108 isPalindrome("12345678987654321")

109 isPalindrome("12345678987654321")

110 isPalindrome("12345678987654321")

111 isPalindrome("12345678987654321")

112 isPalindrome("12345678987654321")

113 isPalindrome("12345678987654321")

114 isPalindrome("12345678987654321")

115 isPalindrome("12345678987654321")

116 isPalindrome("12345678987654321")

117 isPalindrome("12345678987654321")

118 isPalindrome("12345678987654321")

119 isPalindrome("12345678987654321")

120 isPalindrome("12345678987654321")

121 isPalindrome("12345678987654321")

122 isPalindrome("12345678987654321")

123 isPalindrome("12345678987654321")

124 isPalindrome("12345678987654321")

125 isPalindrome("12345678987654321")

126 isPalindrome("12345678987654321")

127 isPalindrome("12345678987654321")

128 isPalindrome("12345678987654321")

129 isPalindrome("12345678987654321")

130 isPalindrome("12345678987654321")

131 isPalindrome("12345678987654321")

132 isPalindrome("12345678987654321")

133 isPalindrome("12345678987654321")

134 isPalindrome("12345678987654321")

135 isPalindrome("12345678987654321")

136 isPalindrome("12345678987654321")

137 isPalindrome("12345678987654321")

138 isPalindrome("12345678987654321")

139 isPalindrome("12345678987654321")

140 isPalindrome("12345678987654321")

141 isPalindrome("12345678987654321")

142 isPalindrome("12345678987654321")

143 isPalindrome("12345678987654321")

144 isPalindrome("12345678987654321")

145 isPalindrome("12345678987654321")

146 isPalindrome("12345678987654321")

147 isPalindrome("12345678987654321")

148 isPalindrome("12345678987654321")

149 isPalindrome("12345678987654321")

150 isPalindrome("12345678987654321")

151 isPalindrome("12345678987654321")

152 isPalindrome("12345678987654321")

153 isPalindrome("12345678987654321")

154 isPalindrome("12345678987654321")

155 isPalindrome("12345678987654321")

156 isPalindrome("12345678987654321")

157 isPalindrome("12345678987654321")

158 isPalindrome("12345678987654321")

159 isPalindrome("12345678987654321")

→ Recursive Algorithms have two types of cases, recursive cases & base cases.

MARCH  
14

Every recursive function call must terminate at base case.  
Any problem that can be solved recursively can also be solved iteratively.

S	M	T	W	T	F	S
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MAR 2024

THURSDAY

# Sum of natural Nos. using recursion:-

```
08 def fun(n):  
09     if n == 1:  
10         return n  
11     else: return fun(n-1) + n
```

$$\text{sum}(1) = 1$$

$$\text{sum}(2) = 2 + \text{sum}(1)$$

$$\text{sum}(3) = 3 + \text{sum}(2)$$

$$\text{sum}(4) = 4 + \text{sum}(3)$$

# Palindrome using Recursion:-

```
11 def isPal(st, s, e):  
12     if (s == e):  
13         return True  
14     if (st[s] != st[e]):  
15         return False  
16     if (s < e + 1):  
17         return isPal(st, s+1, e-1)  
18     return True
```

def isPal(st):

n = len(st)

if (n == 0):

return True

return isPal(st, 0, n-1)

# Fibonacci Sequence:-

```
04 def fib(n):  
05     if n in [0, 1]:  
06         return n  
07     else:  
08         return fib(n-1) + fib(n-2)
```

# Point Even Numbers!

```
def Even(n):  
    if n <= 0:  
        return  
    if n % 2 == 0:  
        Even(n-2)  
        print(n)
```

# factorial

```
def fac(n):  
    if n in [0, 1]:  
        return 1  
    else:  
        return fac(n-1) * n
```

Even till 0 or for # even(n-1)  
decrease till 2

n = input()

a = 1

for i in range(2, int(n)):  
 a = a \* i

fac(a)

Sympathy is the key that fits the lock of any heart

	S	M	T	W	T	F	S
APR 2024	1	2	3	4	5	6	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
28	29	30					

MARCH

'24

11th Week  
075-291

15

FRIDAY

## # Tower of Hanoi

Tower of Hanoi is a mathematical puzzle where we have three rods (A, B, C) and N disks. Initially, all the disks are stacked in decreasing value of diameter, i.e. smallest disk is placed on the top and they are on rod A. The objective of the puzzle is to move the entire stack of another rod (here considered C), obeying the following simple rules.

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack. i.e a disk can only be moved if it is the uppermost disk on a stack.
- No disk may be placed on top of a smaller disk.

idea!

- Shift 'N-1' disks from 'A' to 'B', using C.
- Shift last disk from 'A' to 'C'. [Source to Destination]
- Shift 'N-1' disks from 'B' to 'C', using A. [Aux to Dest]

A = Source  
B = Auxiliary  
C = Destination

Source      destination      helper

⇒ def TOH (n, from-rod, to-rod, aux-rod):  
if n == 1:  
return # Print ("Move 1 from " + from-rod + " to " + to-rod)  
TOH (n-1, from-rod, aux-rod, to-rod)  
print ("Move disk " + n + " from " + from-rod + " to " + to-rod, to-rod)  
TOH (n-1, aux-rod, to-rod, from-rod).

↳ यह किसके द्वारा किया जाता है?

$$\begin{pmatrix} F & T & A \\ F & A & T \\ A & T & F \end{pmatrix}$$

$$\begin{pmatrix} A & B & C \\ A & C & B \\ B & A & C \end{pmatrix}$$

No. of Movement =  $2^n - 1$ 

$$TC = O(2^n)$$

$$SC = O(n)$$

MARCH

16

'24

11th Week  
076290

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MAR 2024

SATURDAY

Rope cutting Problem.

- 08 Given a Rod of length  $N$  meters, and the rod can be cut in only 3 sizes  $A, B$  and  $C$ . the task is to maximize the no. of cuts in rod.
- 09 If it is impossible to make cut then Point -1.

$$\frac{n}{b} = n=5, a=2, b=5, c=1$$

$$0 < a, b, c \leq n.$$

$$0/b = 5$$

{ we make 5 piece of length 1 each }

$$\frac{n}{b} = n=23, a=12, b=9, c=11$$

[ Length of every piece (after cuts) ]

$$\frac{n}{b} = 2 \quad \{ \text{make 2 piece of length 11 \& 12} \}$$

01 def maxpiece(n, a, b, c):

if  $n = 0$ :

return 0

if  $n \leq -1$ :

return -1

yes = max(maxpiece(n-a, a, b, c),

maxpiece(n-b, a, b, c),

maxpiece(n-c, a, b, c))

05 if yes == -1:

return -1

06 return yes + 1

07

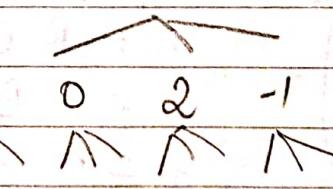
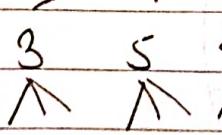
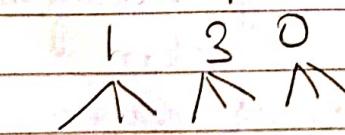
$n=23$

$n=12$

$n=9$

$n=11$

$\Theta(3^n)$



	S	M	T	W	T	F	S
APR 2024		1	2	3	4	5	6
	7	8	9	10	11	12	13
	14	15	16	17	18	19	20
	21	22	23	24	25	26	27
	28	29	30				

MARCH

124

12th Week  
OT7-289

19

SUNDAY

Subsequence

## # Subsequence of a given string (any order)

08 for string subset  $\approx$  subsequence.

Given a string, we have to find out all subsequence of it. A string is a

09 subsequence of a given string, that is generated by deleting some character of a given string w/o changing its order.

10

eg  $\text{?} / \text{p} = \text{"AB"}$

#  $\text{0} / \text{p} = \text{"", 'A', 'B', 'AB'}$

12  $\text{?} / \text{p} = \text{"ABC"}$

$\text{0} / \text{p} = \text{'A', ' ', 'B', 'C', 'AB', 'AC', 'BC', 'ABC'}$

01

curr = " "  $\rightarrow$  index = 0

02  $\nearrow$  " " include "A"  $\rightarrow$  index = 1

03 exclude  $\nearrow$  " " "B" "A" "AB"  $\rightarrow$  index = 2

04  $\nearrow$  " " "C" "B" "BC" "A" "AC" "AB" "ABC"  $\rightarrow$  index = 3

05

def sub (str, curr, ind):

06 if ind == len(str):

print (curr, end = ' ')

return

Sub (str, curr + index + 1)

Sub (str, curr + str[index], index + 1)

TC =  $O(2^n)$

SC =  $O(n)$

MARCH

18

'24

12th Week  
078-288

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MAR 2024

MONDAY

#

Printing all permutations.

08

- A permutation also called "assignment number" or "order" is a re-arrangement of the elements of an ordered list  $S$  into a one-to-one correspondence with  $S$  itself.
- A string of length  $n$  has  $n!$  permutations.

11 def Permute( $s, i$ ):

n = len(s)

12 if ( $i == n - 1$ ):

Print (".".join(s))

01 return

for  $j$  in range ( $i, n$ ):     $s[i], s[j] = s[j], s[i]$     Permute ( $s, i + 1$ )03      $s[i], s[j] = s[j], s[i]$ 

04

05

 $i=2$  $i=1$ 

ABC

ABC

ACB

 $i=0$ 

ABC

BAC

BAC

BAC

BCA

BCA

CBA

CBA

CAB

	S	M	T	W	T	F	S
APR 2024		1	2	3	4	5	6
	7	8	9	10	11	12	13
	14	15	16	17	18	19	20
	21	22	23	24	25	26	27
	28	29	30				

MARCH

24

12th Week  
079-287

19

TUESDAY

## # Subset Sum Problem

08

$$\text{if } p = \{10, 5, 2, 3, 6\} \text{ then } \text{if } p = \{10, 12, 15\}$$

09

$$\text{Sum} = 8 \text{ then } \text{if } \text{Sum} = 0$$

10

$$0/p = 1 + 2 \cdot \text{Sum} \text{ then } 0/p = 1$$

Subset of  $\{1, 2, 3\}$  are  $\{\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}$

11

exclude  $\{\}$  include  $\{\}$   $n=3$  then  $n=3$

01

$\{\}$   $\{15\}$   $n=2$  then  $n=2$

02

$\{\}, \{20\}, \{15\}, \{15, 20\} - n=1$

03

$\{\}, \{10\}, \{20\}, \{20, 10\}, \{15\}, \{15, 20\}, \{15, 20\}, \{15, 20, 10\} - n=0$

04

→ Answer

def subset(L, n, sum):  $T(n) = T(n-1) + O(1)$   $T(n) = O(2^n) + O(n) = O(n)$

05

If  $n=0$ :

return 1 if sum == 0 else 0

06

return subset(L, n-1, sum) + subset(L, n-1, sum - L[n-1])

07

if L:

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

else if L:

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

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

MARCH

20

'24

12th Week  
OSO-286

S	M	T	W	T	F	S
31				1	2	
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MAR 2024

WEDNESDAY

## # Josephus Problem in Python.

08

{10/2024} and ~~some notes~~

Given the total number of Person N and a number K which indicates that K-1 persons are skipped and the kth person is killed in a circle.

The task is to choose the person in the initial circle that survives.

10

e.g.  $\frac{N}{k} = N = 5$  and  $k = 2$   $\frac{N}{k} = 3$

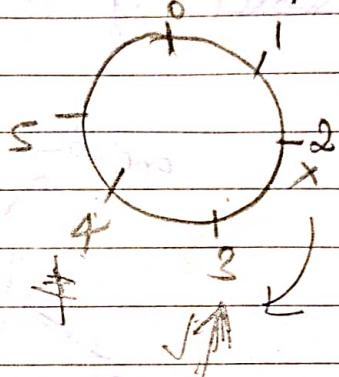
Firstly, the person at position 2 is killed.

then the person at position 4 is killed.

then the person at position 1 is killed.

Finally the person at position 5 is killed.

so the person at positions 3 survives.



02

```
def Jos(n, k):
```

TC = O(n)

03

```
    if n == 1:
```

SC = O(n)

04

```
    return 0
```

again starting  $\frac{1}{4} \rightarrow 3 \rightarrow 1$   
 $\frac{1}{2} \rightarrow 4 \rightarrow 1$

05

```
    else:
```

```
def gos(begin and (n, k)):
```

```
    return Jos(n, k) + 1
```

06

$$(Jos(4, 3) + 3) \% 5$$

$$((Jos(2, 3) + 3) \% 4 + 3) \% 5$$

07

$$(((Jos(1, 3) + 3) + 3) \% 3 + 3) \% 4$$

If counting begins with 1

$$\downarrow \quad \quad \quad + 3 \% 5$$

$$= (3) \checkmark$$