Class Average

- Given marks secured in CSE1001 by the students in a class, design an algorithm and write a Python code to determine the class average. Print only two decimal digits in average

Class Average

Input	**Processing**	**Output**
Number of students in class, mark scored by each student	Determine total of marks secured by students Find average of marks	Class average of marks

Average marks scored by ‘N’ number of Students

```
Step 1:  Start
Step 2 : Read Number Of Students
Step 3 : Initialize  counter as 0
Step 4 : Input  mark
Step 5 : Add the mark with total
Step 6 : Increment  the counter by 1
Step 7: repeat Step 4 to Step 6 until counter  less than number of  students
 Step 7: Divide the total by number of students and store it in average
Step 8: Display the average
Step 9: Stop
```

Test Cases

Input

5

90 85 70 50 60

Output

71.00

Processing Involved

Already Know

- To read values from user
- To check if a condition is satisfied
- Print characters

Yet to learn

- Repeatedly execute a set of statements

Need of iterative control

Repeated execution of set of statements

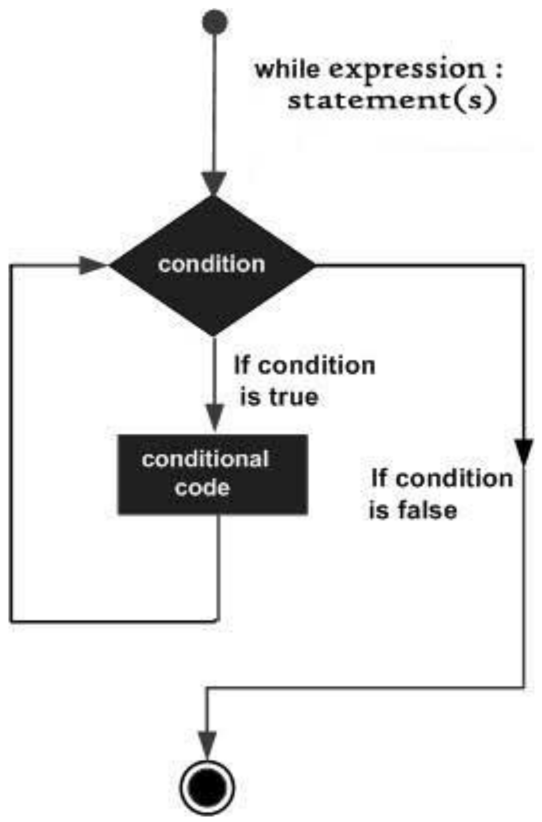
- An **iterative control statement** is a control statement providing repeated execution of a set of instructions
- Because of their repeated execution, iterative control structures are commonly referred to as “loops.”

While statement

- Repeatedly executes a set of statements based on a provided Boolean expression (condition).
- All iterative control needed in a program can be achieved by use of the while statement.

Syntax of While in Python

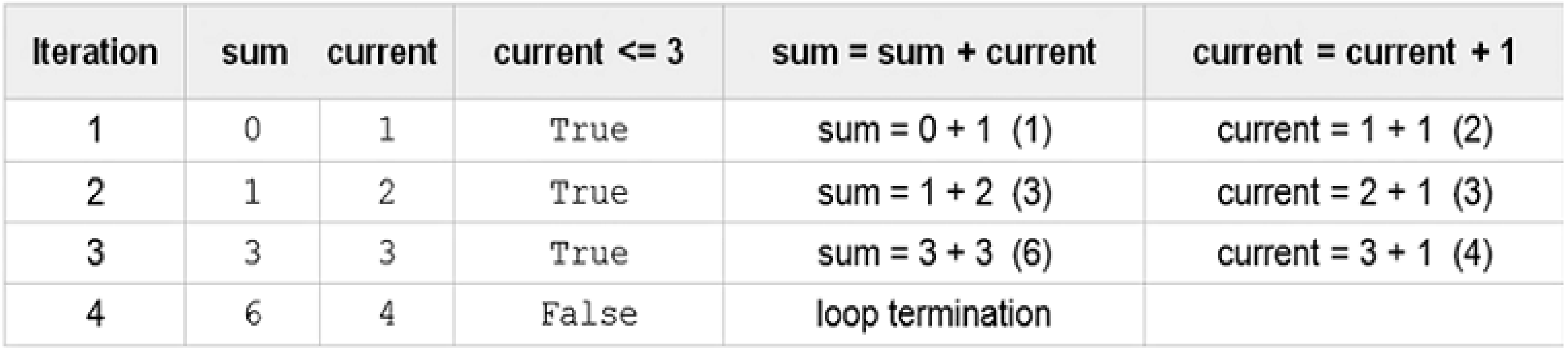
```
while test:                 # Loop test
    statements              # Loop body
else:                       # Optional else
    statements
# Run if didn't exit loop with break
```



Example use

Sum of first ‘n’ numbers

```
sum =0
current =1
n=3
while  current  <= n:
    sum=sum + current
    current = current + 1
```



Iteration	sum	current	current <= 3	sum = sum + current	current = current + 1
1	0	1	True	sum = 0 + 1 (1)	current = 1 + 1 (2)
2	1	2	True	sum = 1 + 2 (3)	current = 2 + 1 (3)
3	3	3	True	sum = 3 + 3 (6)	current = 3 + 1 (4)
4	6	4	False	loop termination	

```
N=int(input("Enter number of students :"))
counter=1
total=0
while counter <= N:
    mark=int(input("Enter Mark:"))
    total=total+mark
    counter+=1
avr=total/N
print(avr)
```

Print values from 0 to 9 in a line

```
a=0; b=10
while a < b:            # One way to code counter loops
        print(a, end=' ')
        a += 1                  # Or, a = a + 1
```

Output:

```
0 1 2 3 4 5 6 7 8 9
```

Include end=‘ ‘ in print statement to suppress default move to new line

Break, continue, pass, and the Loop else

- break Jumps out of the closest enclosing loop
- continue Jumps to the top of the closest enclosing loop
- pass Does nothing at all: it’s an empty statement placeholder
- Loop else block Runs
- if and only if the loop is exited normally (i.e., without hitting a break)

Program with Continue

```
counter=1
total=0
while True:
    mark=int(input("Enter Mark:"))
    if mark<0:
        break
    total=total+mark
    counter+=1
avr=total/counter
print(avr)
```

Break statement

```
• while True:
        name = input('Enter name:')
        if name == 'stop': break
       age  = input('Enter age: ')
        print('Hello', name, '=>', int(age) ** 2)
```

Output:

```
 Enter name:bob
Enter age: 40
Hello bob => 1600
```

Pass statement

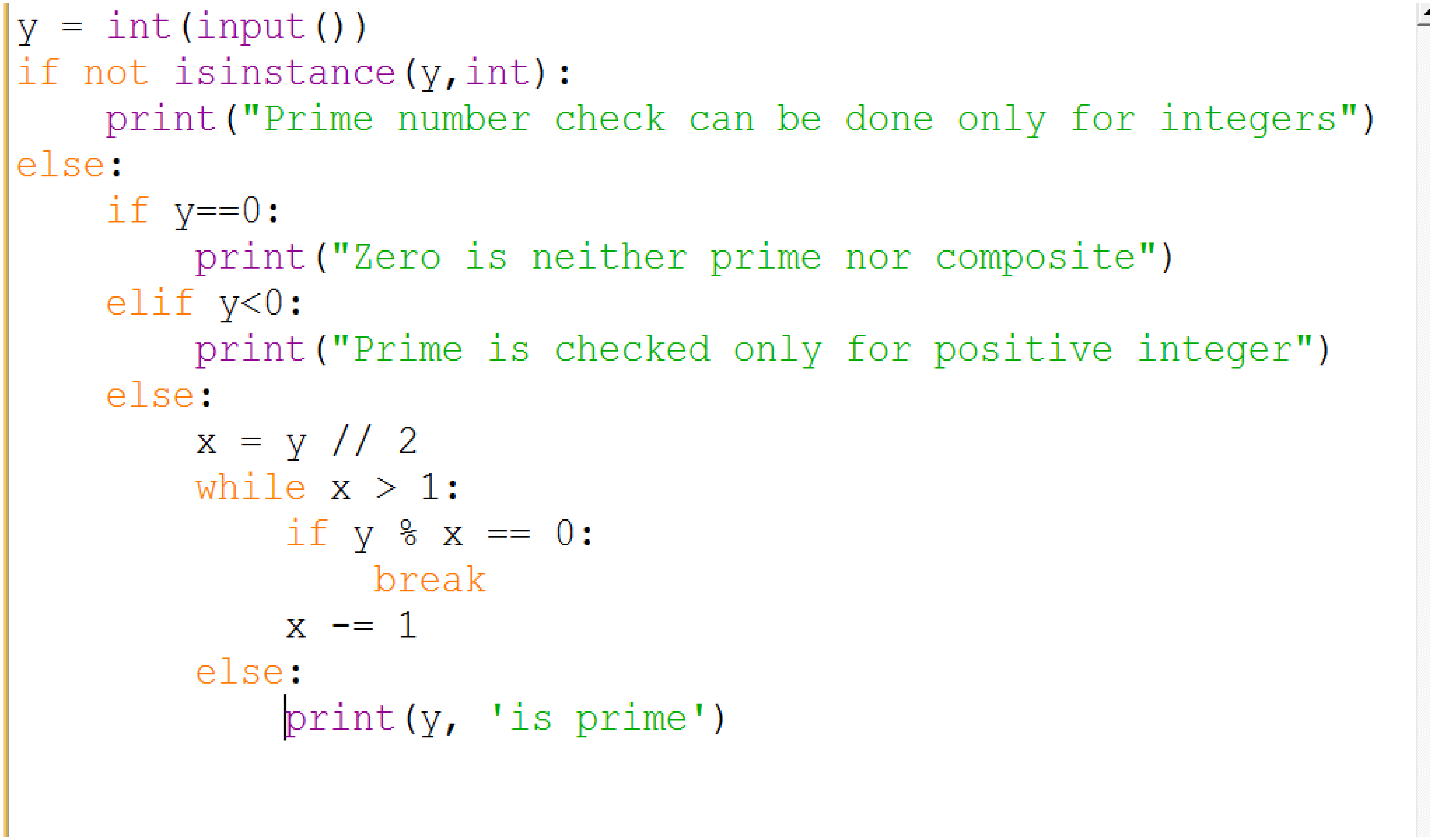
- Infinite loop
- while True: pass

  # Type Ctrl-C to stop me!

Print all even numbers less than 10 and greater than or equal to 0

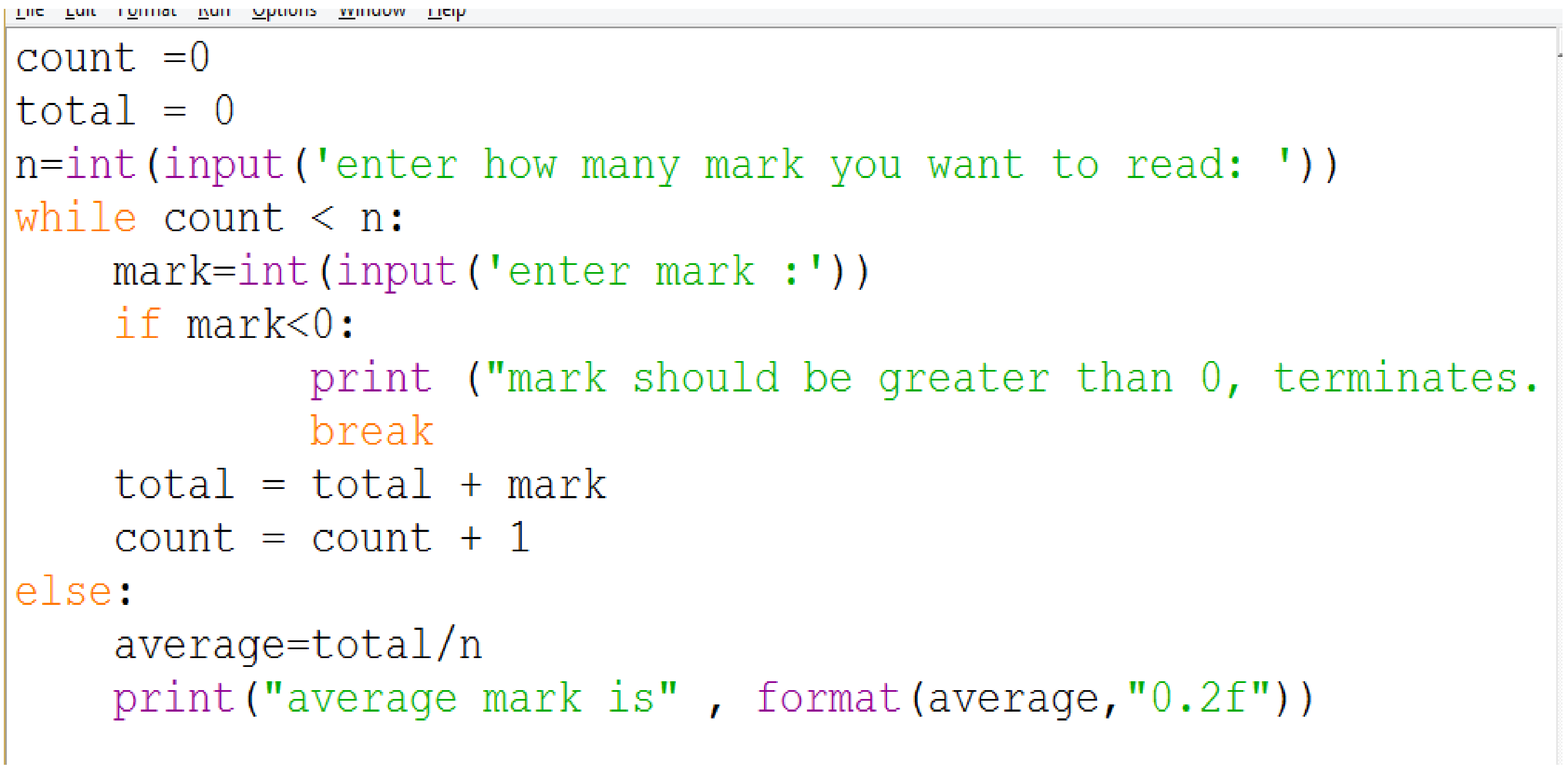
```
x = 10
while x:
        x = x-1                          # Or, x -= 1
        if x % 2 != 0: continue          # Odd? -- skip print
        print(x, end=' ')
```

Check if a given number is Prime



```python
y = int(input())
if not isinstance(y,int):
    print("Prime number check can be done only for integers")
else:
    if y==0:
        print("Zero is neither prime nor composite")
    elif y<0:
        print("Prime is checked only for positive integer")
    else:
        x = y // 2
        while x > 1:
            if y % x == 0:
                break
            x -= 1
        else:
            print(y, 'is prime')
```

Class Average



```
count =0
total = 0
n=int(input('enter how many mark you want to read: '))
while count < n:
    mark=int(input('enter mark :'))
    if mark<0:
            print ("mark should be greater than 0, terminates.
            break
    total = total + mark
    count = count + 1
else:
    average=total/n
    print("average mark is" , format(average,"0.2f"))
```

Pattern Generation

- Your teacher has given you the task to draw the structure of a staircase. Being an expert programmer, you decided to make a program for the same. You are given the height of the staircase. Given the height of the staircase, write a program to print a staircase as shown in the example. For example, Staircase of height 6:

```
#
##
###
####
#####
######
```

Boundary Conditions: height >0

Pattern Generation

Input	Processing	Output
Staircase height	Create steps one by one To create a step print character equal to length of step	Pattern

Pseudocode

```
READ staircase_height
if staircase_height > 0
x = 1
Repeat
y = 1
Repeat
    print #
    y = y + 1
Until y <= x
x = x + 1
Until x <= staircase_height
End if
Else
Print “Invalid input”
```

Test Cases

Input

3

Output

```
#
# #
# # #
```

Processing Involved

Print step by step

Test Cases

Input

-1

Output

Invalid input

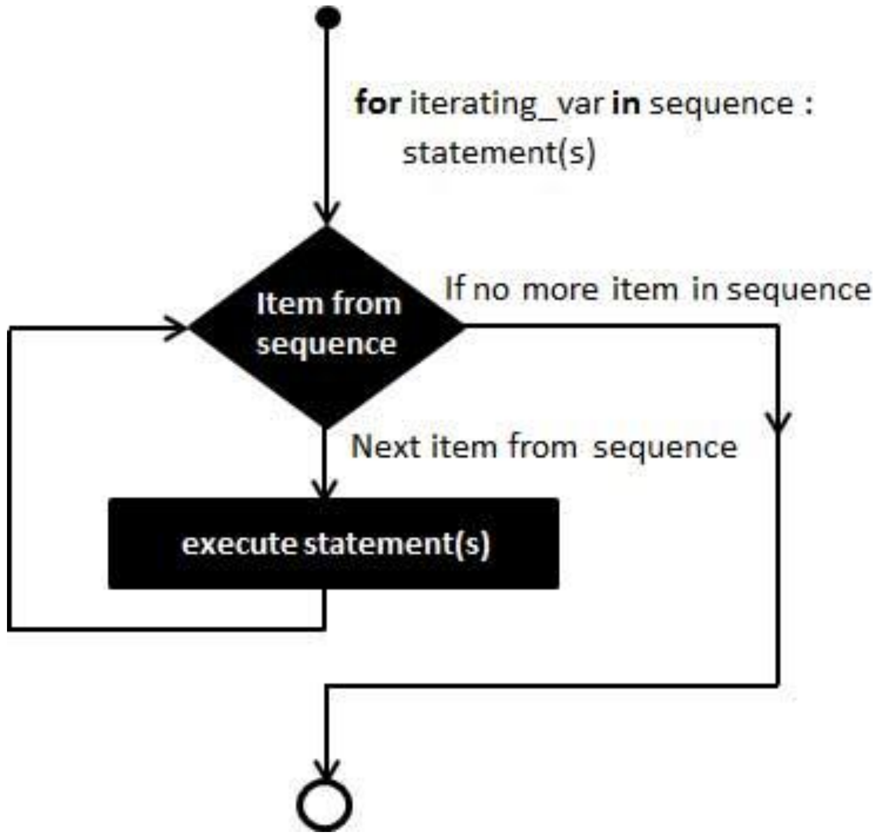
Processing Involved

Boundary condition check fails

For iteration

- In while loop, we cannot predict how many times the loop will repeat
- The number of iterations depends on the input or until the conditional expression remains true
- While loop is ideal when stop criteria is not explicit

Control flow of for statement



Syntax of for Statement

```
for target in object:
    # Assign object items to target
    statements
    if test: break              # Exit loop now, skip else
    if test: continue           # Go to top of loop now
else:    statements             # If we didn't hit a
    'break’
```

For and Strings

```
for iterating_var in sequence or range:
        statement(s)
```

Example:

```
for letter in 'Python':
   print 'Current Letter :', letter
```

For and Strings

When the above code is executed:

```
Current Letter : P
 Current Letter : y
 Current Letter : t
 Current Letter : h
 Current Letter : o
 Current Letter : n
```

For and Range

```
for n in range(1, 6):
    print(n)
```

When the above code is executed:

```
1
2
3
4
5
```

range function call

Syntax - range(begin,end,step)

where

Begin - first value in the range; if omitted, then default value is 0

end - one past the last value in the range; end value may not be omitted

Step - amount to increment or decrement; if this parameter is omitted, it defaults to 1 and counts up by ones

begin, end, and step must all be integer values;

floating-point values and other types are not allowed

Example for Range

range(10) → 0,1,2,3,4,5,6,7,8,9

range(1, 10) → 1,2,3,4,5,6,7,8,9

range(1, 10, 2) → 1,3,5,7,9

range(10, 0, -1) → 10,9,8,7,6,5,4,3,2,1

range(10, 0, -2) → 10,8,6,4,2

range(2, 11, 2) → 2,4,6,8,10

range(-5, 5) → −5,−4,−3,−2,−1,0,1,2,3,4

range(1, 2) → 1

range(1, 1) → (empty)

range(1, -1) → (empty)

range(1, -1, -1) → 1,0

range(0) → (empty)

Print Even Numbers Using Range

```
>>> for i in range(2,10,2):
        print(i)

Output:
2
4
6
8
```

```python
print("Enter number of steps")
n = int(input())
for i in range(0,n):
    for j in range(0,i+1):
        print('#',end = ' ')
    print()
```