CONTENTS

(Learning Outcomes) ☺

XI CS 2020-21

FLOW OF CONTROL

- **Types of Statements**
- **INTRODUCTION TO FLOW OF CONTROL**
- **Conditional statements: if, if-else, if-elif-else;**
- **simple programs: Eg.**

  : absolute value : sort 3 numbers : divisibility of a number.
- **Notion of iterative computation and control flow:**

  for(range(),len()), while
- **Suggested programs:**

  : calculation of simple and compound interests,

  : finding the factorial of a positive number etc.
- **Some more extra program codes to strengthen your programming.**
- ***using flowcharts (Covered in Computational Thinking and Problem Solving topic)***

XI IP 2020-21

Control Statements: if-else, for loop

TYPES OF STATEMENTS

- Empty / Null statements
- Simple / Single statements
- Compound statements

TYPES OF STATEMENTS in Python

✓Statements are the instructions given to computer to perform any kind of action viz. data movements, making decision, repeating actions

✓Statements form the smallest executable unit within a program.

- **Empty statement/ null operation statement**
  - Statement which does nothing. It is as follows: →

```
>>> pass
>>>
```

- **Simple statement (Single statement)**
  - Any single executable statement is a simple statement.

```
>>> a=10+20
>>> print(a)
```

- **Compound statement**

A group of statements executed as a unit is a compound statement.

A compound statement has:

(i) **a header line** which begins with a keyword and ends with a colon.

(ii) **a body** consisting of one or more Python statements, each indented inside the header line. All the statements are at the same level of indentation.

```
>>> if a>0:
        print("Positive Number")
        print("Thank you")
```

Header Line

BODY

COMPOUND STATEMENT

FLOW OF CONTROL

OR

CONTROL STRUCTURES

OR

CONTROL FLOW OF STATEMENTS

1. SEQUENCE
2. SELECTION / CONDITIONAL/DECISIONAL
   → if, if-else, if-elif, nested ifs
3. ITERATIVE / LOOPING
   → for loop, while loop

FLOW OF CONTROL OR CONTROL STRUCTURES OR CONTROL FLOW OF STATEMENTS

Flow of Control is referred as the way/ order of the flow of execution of the program statements.

In a program, statement may be executed sequentially, selectively or iteratively.

Every program language provides or supports the following :-

Types of constructs :

1) Sequence

2) Selection/Conditional/Decisional

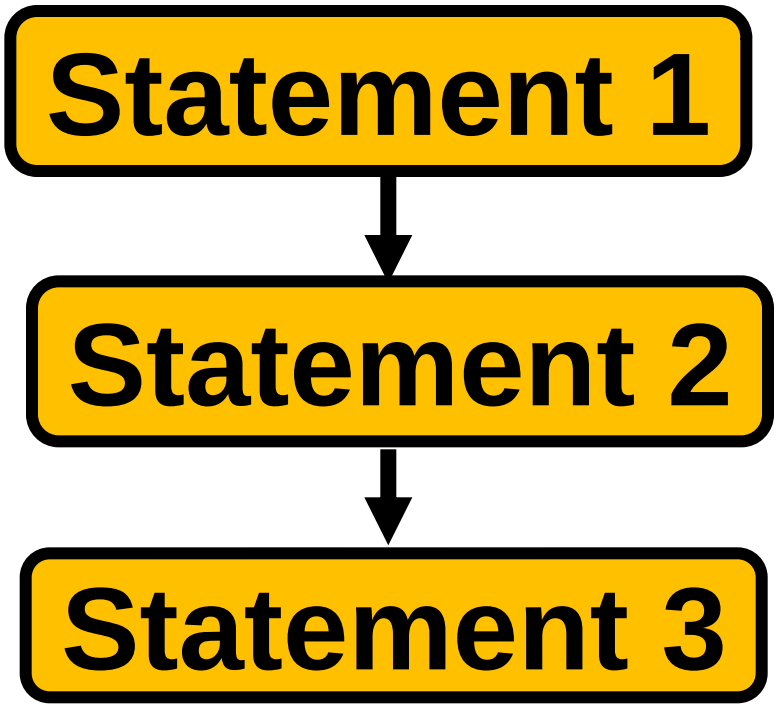
→if, if-else, if-elif, nested ifs

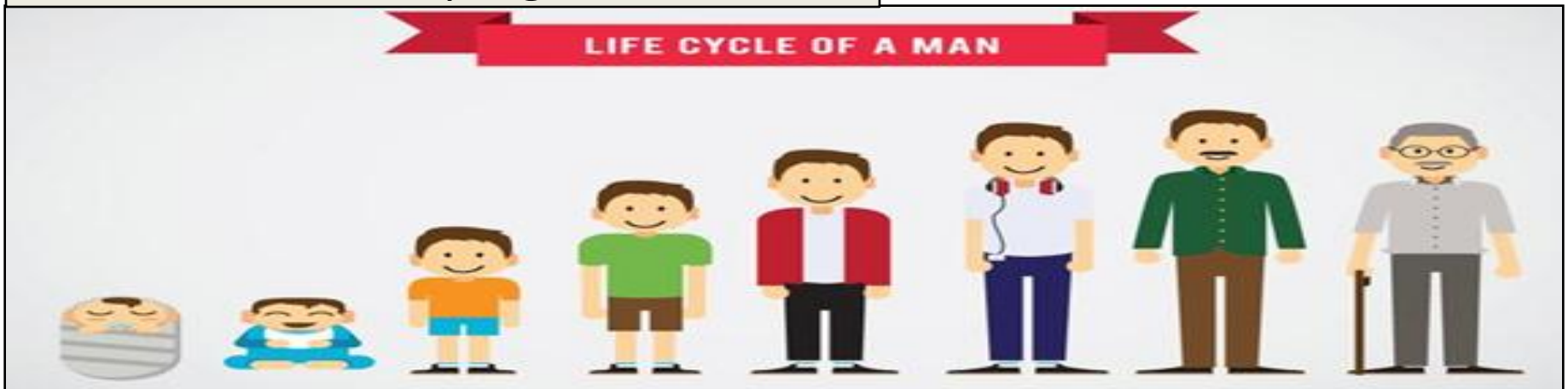
3) Iteration/Looping

→ For loop,while loop

1. SEQUENCE CONSTRUCT / SEQUENTIAL FLOW OF CONTROL

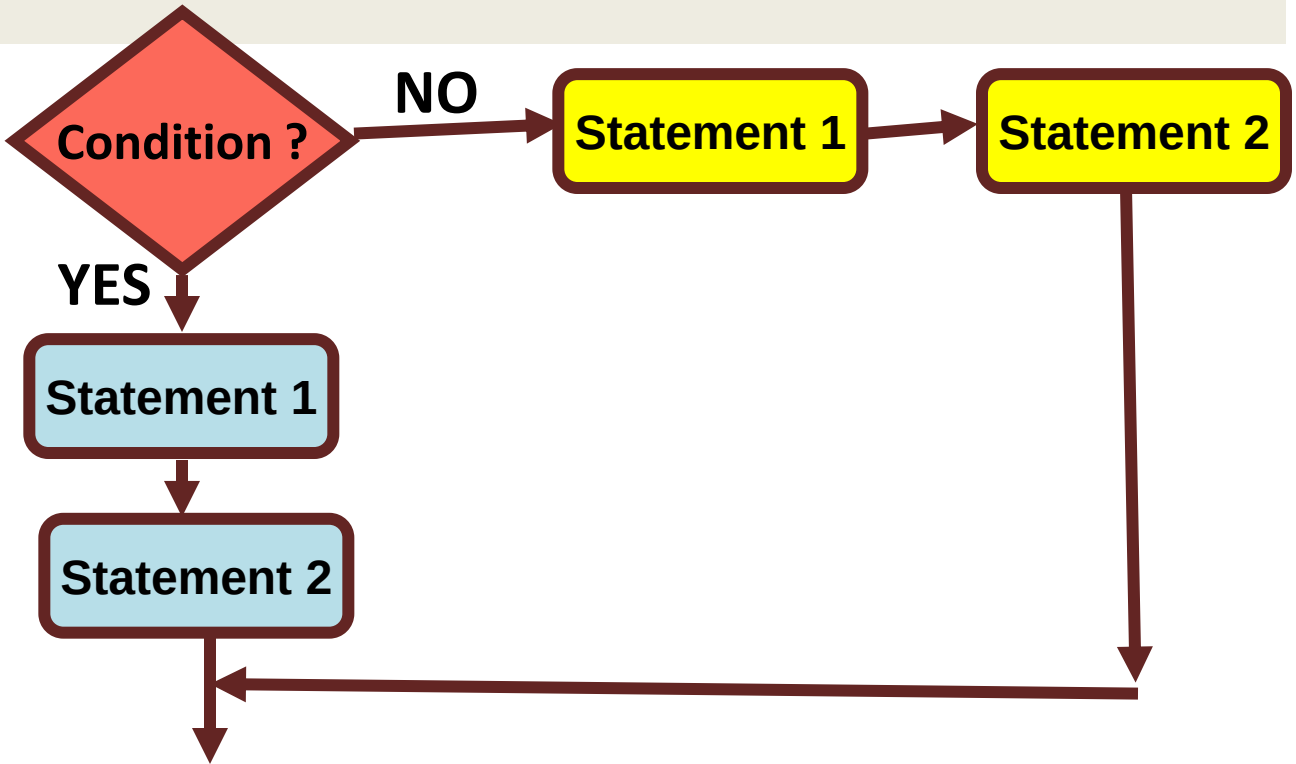
- Sequence construct means statement are executed sequentially.
- It represents the default flow of statements.
- Every program begins with the first statement of the program. When the final statement of the program is executed, the program is done.





2. SELECTION / DECISIONAL / CONDITIONAL CONSTRUCT or SELECTION FLOW OF CONTROL

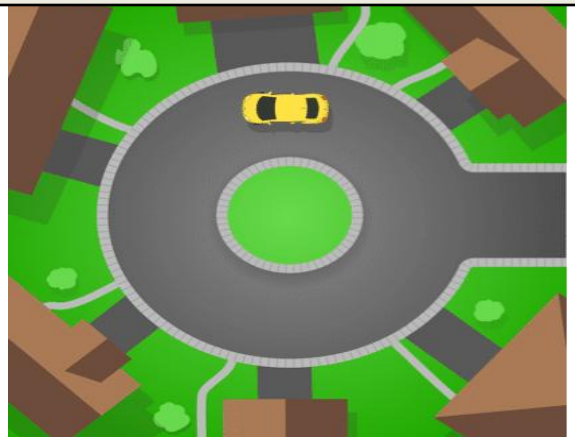
- The Selection construct means the execution of statement(s) depending upon a condition-test.
- If a condition evaluates to true, a course-of-action(a set of statements) is followed otherwise another course-of-action (a different set of statements).
- It helps in making decision about which set-of-statements is to be executed.

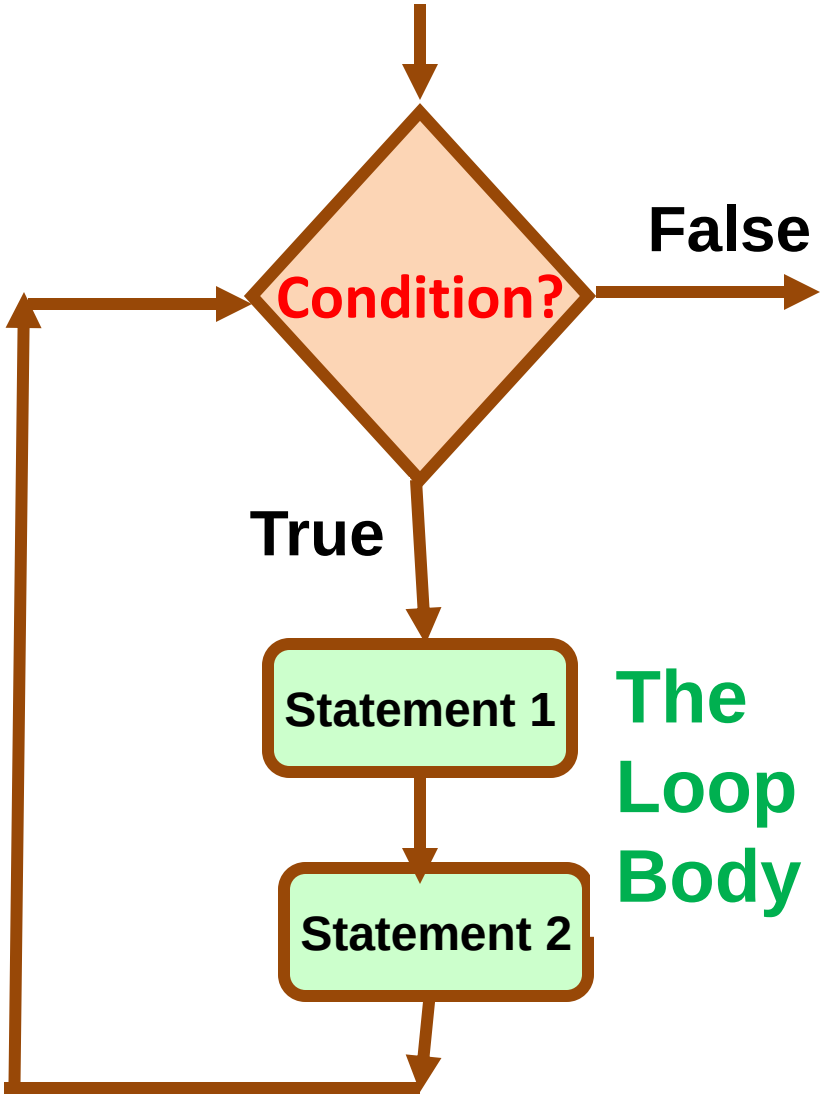


3. ITERATION CONSTRUCT / LOOPING CONSTRUCT / ITERATIVE FLOW OF CONTROL

- Iteration construct means repetition of set of statements depending upon a condition test till the time of condition is true (or false depending upon the loop)
- A set of statements are repeated again and again. As soon as the condition become false (or true), the repetition stops.
- The iteration construct is also called **"Looping Construct"**.

  Eg. *for loop* , *while loop*





SELECTION/ CONDITIONAL/ DECISION MAKING CONSTRUCT

There are three types of decision making statement.

1. **if statements**
2. **if-else statements**
3. **Nested if-else statement**

if statements

- **If** statement must be provided with a condition.
- If the condition is **True**, the indented body / block gets executed.
- If the condition is **False**, then the control doesn't execute the if body/block.

ifprog1.py - C:\Users\Agni\AppData\Local\Programs\Python\Python38-32\ifprog1.py (3.8.4)

File Edit Format Run Options Window Help

```
a=10
b=30
if a<b:                          # Header line
    print("a is smaller")         # Loop body
```

Condition

Ln: 5 Col: 0

```
= RESTART: C:\Users\Agni\AppData\Local\Prog
a is smaller
>>>
```

if statements examples contd.

File Edit Format Run Options Window Help

```
a=10
b=30
if a<b:                          # Header line
    print("a is smaller")        # Loop body
    print("Thank you")           # Loop body
```

Ln: 5 Col: 3

```
== RESTART: C:\Users\Agni\AppData\Local\Pro
a is smaller
Thank you
```

File Edit Format Run Options Window Help

```
a=10
b=30
if a<b:                          # Header line
    print("a is smaller")        # Loop body
print("Thank you")
```

Ln: 4 C

```
== RESTART: C:\Users\Agni\AppData\Local\P
a is smaller
Thank you
```

if statements examples contd.

```
File  Edit  Format  Run  Options  Window  Help
a=10
b=30
if a>b:                          # Header line
    print("a is greater")        # Loop body
print("Thank you")

Ln: 4  Col: 3
== RESTART: C:\Users\Agni\AppData\Local\Pro
Thank you
```

```
File  Edit  Format  Run  Options  Window  Help
a=10
b=30
if a>b:                          # Header line
    print("a is greater")        # Loop body
    print("Thank you")           # Loop body

Ln: 5  Col: 3
== RESTART: C:\Users\Agni\AppData\Local\Pro
>>>
```

Note:

To indicate a block of code in Python, you must indent each line of the block by the same amount. In above e.g. both print statements are part of if condition because of both are at same level indented.

HANDS ON….FOR YOU ☺

Programs on only if

Q1) Write a program to accept three integers and print the largest of the three. Make use of only if statement.

if-else statements

This form of **if statement tests a condition and if the condition evaluates to true**, it carries out statements indented below if and in case **condition evaluates to false, it carries out statements indented below else.**

File Edit Format Run Options Window Help

```
n=int(input('Enter a number'))
if n%2==0:
    print('EVEN number')
else:
    print('ODD number')
```

OUTPUT

```
Enter a number8
EVEN number
```

NOTE: Conditions are provided only with **if** and not with the **else** clause.

COMPARISON BETWEEN

only if statements and if-else statements

```
File  Edit  Format  Run  Options  Window  Help
n=int(input('Enter a number'))
if n%2==0:
    print('EVEN number')
else:
    print('ODD number')
= RESTART: C:/Users/Agni/AppDa
Enter a number8
EVEN number
```

If-else statements are MORE EFFICIENT BECAUSE NO. OF COMPARISONS AND CONDITION CHECKS ARE LESS THAN ONLY if statements.

Both programs will give the same output.

```
File  Edit  Format  Run  Options  Window  Help
n=int(input('Enter a number'))
if n%2==0:
    print('EVEN number')
if n%2!=0:
    print('ODD number')
= RESTART: C:/Users/Agni/AppDa
Enter a number7
ODD number
```

if-elif statements

Sometimes, we need to check another condition in case the test-condition of ***if*** evaluates to ***false***. That is , we want to check a condition when control reaches **else** part i.e. condition test in the form of **else if**.

Python syntax (if-elif statements)

```
if <condition expression>:
        statement
        [statements]
elif <condition expression>:
        statement
        [statements]
```

```
if <condition expression>:
        statement
        [statements]
elif <condition expression>:
        statement
        [statements]
else:
        statement
        [statements]
```

File Edit Format Run Options Window Help

```python
# To check whether a no. is Positive, Negative or Zero

n=int(input("Enter a no."))
if n>0:
    print("No. is Positive")
elif n<0:
    print("No. is Negative")
else:
    print("It is a zero")
```

OUTPUT

```
Enter a no.-12
No. is Negative
```

Python program to check whether a no. taken from the user is Positive,Negative or Zero(EXAMPLE of if-elif statements)

Python program on cricketer's score. Check for century,fifty or less than 50 (EXAMPLE of if-elif statements)

File Edit Format Run Options Window Help

```python
#PROGRAM TO ILLUSTRATE THE USE OF if-elif
runs=int(input("Enter runs scored"))
if runs>=100:
    print("Batsman scored a century")
elif runs>=50:
    print("Batsman scored a fifty")
else:
    print("Batsman has neither scored a century nor fifty")
```

OUTPUT

```
Enter runs scored70
Batsman scored a fifty
```

MORE PROGRAMS using if-elif

```
# ARITHMETIC CALCULATOR
a=int(input("Enter 1st no."))
b=int(input("Enter 2nd no."))
op=input("Enter operator +,-,*,/")
if op=='+':
    c=a+b
elif op=='-':
    c=a-b
elif op=='*':
    c=a*b
elif op=='/':
    c=a/b
else:
    print("Invalid operator")

print(a,op,b,'=',c) # This statement is OUTSIDE if,elif and else
```

OUTPUT

```
Enter 1st no.25
Enter 2nd no.6
Enter operator +,-,*,/*
25 * 6 = 150
```

The nested if statement

A nested if is an if that has another if in its if's body or in elif's body or in its else's body.

The **nested if** can have one of the following forms:

Form1 (if inside if's body)

Form2 (if inside elif's body)

Form3 (if inside else's body

Form 4 (if inside if's as well as else's or elif's body, i.e. elif's body , i.e. multiple ifs inside.)

Form 1

```
if <condition expression>:
    if <condition
    expression>:
        [statements]
    else:
        [statements]
elif <condition
    expression>:
    Statement
    [Statements]
else:
    Statement
    [Statements]
```

Form 2

```
if <condition expression>:
    statement
    [statements]
elif <condition expression>:
    if <condition expression>:
        [statements]
    else:
        [statements]
else:
        statement
        [statements]
```

Form 3

```
if <condition expression>:
    statement
    [statements]
elif <condition expression>:
    statement
    [statements]
else:
    if <condition expression>:
        statement(s)
    else:
        statement(s)
```

Form 4

```
if <condition expression>:
    if <condition expression>:
        statement(s)
    else:
        statement(s)
elif <condition expression>:
    if <condition expression>:
        statement(s)
    else:
        statement(s)
else:
    if <condition expression>:
        statement(s)
    else:
        statement(s)
```

EXAMPLE PROGRAM on nested if : SORT 3 numbers

```
# Program that reads three numbers and print them in ascending order
x=int(input("Enter first number"))
y=int(input("Enter second number"))
z=int(input("Enter third number"))
mini=mid=maxi=None
if x<y and x<z:
    if y<z:
      mini,mid,maxi=x,y,z
    else:
      mini,mid,maxi=x,z,y
elif y<x and y<z:
    if x<z:
      mini,mid,maxi=y,x,z
    else:
      mini,mid,maxi=y,z,x
else:
    if x<y:
      mini,mid,maxi=z,x,y
    else:
      mini,mid,maxi=z,y,x

print("number in ascending order:",mini,mid,maxi)
```

When x is minimum

When y is minimum

When z is minimum

```
Python Programs XI\if-else conditional programs\program that reads t
Enter first number45
Enter second number23
Enter third number90
number in ascending order: 23 45 90
```

format() Function

format() function is used with Strings for presenting the output in a desired format.

In format() function,

< symbol indicates LEFT alignment

> Symbol indicates RIGHT alignment

```
#Displaying format() function
#PROGRAM ON SALES,DISCOUNT,TAX,NET AMOUNT Payable
#-----------------------------------------------
import datetime
sales=float(input("Enter Sales Amount"))
DiscRate=float(input("Enter Discount rate"))
DiscAmt=sales*DiscRate/100
TaxRate=float(input("Enter Sales Tax Rate"))
TaxAmt=(sales-DiscAmt)*TaxRate/100
NetAmt=sales-DiscAmt-TaxAmt
print("------------SALES COUNTER------------")
print("SALES:{0:>25.2f}".format(sales))
print("DISCOUNT RATE:{0:<3.1f}%".format(DiscRate))
print("TAX RATE :    {0:<3.1f}%".format(TaxRate))
print("DISCOUNT AMOUNT:{0:>15.2f}".format(DiscAmt))
print("TAX AMOUNT:{0:>20.2f}".format(TaxAmt))
print("-------------------------------------")
print("NET AMOUNT:{0:>20.2f}".format(NetAmt))
```

Total 25 characters including spaces after printing SALES:

Right alignment for > symbol

.2 denotes no. of characters after decimal.

Here, 2 digits(characters) after decimal .

f denotes float value

d denotes integer value

```
Enter Sales Amount10000
Enter Discount rate7.5
Enter Sales Tax Rate14
------------SALES COUNTER------------
SALES:                  10000.00
DISCOUNT RATE:7.5%
TAX RATE :    14.0%
DISCOUNT AMOUNT:          750.00
TAX AMOUNT:              1295.00
-------------------------------------
NET AMOUNT:              7955.00
```

```
# SIMPLE INTEREST and COMPOUND INTEREST
import math
p=float(input('Enter Principal amount'))
r=float(input('Enter Annual Rate of Interest'))
t=int(input('Enter Time in years'))
print('--------MENU--------')
print('Press 1 for SIMPLE Interest')
print('Press 2 for COMPOUND Interest')
opt=int(input('Enter your OPTION (1 or 2)'))
if opt==1:
    print('You have opted SIMPLE INTEREST')
    interest=(p*r*t)/100
    total=p+interest
else:
    print('You have opted COMPOUND INTEREST')
    n=int(input('Enter no. of times interest is compounded per year'))
    r=r/100
    total=p*pow((1+r/n),t*n)
    interest=total-p
print("INTEREST:{0:31.2f}".format(interest))
print("TOTAL AMOUNT after Interest:{0:12.2f}".format(total))
```

Form 2

PROGRAM: To calculate SIMPLE Interest and COMPOUND Interest .

Simple Interest=Principal*Rate*Time/100

Compound Interest=TotalAmount-Principal

where , TotalAmount=$P*(1+r/n)^{t*n}$

```
Enter Principal amount5000
Enter Annual Rate of Interest2.5
Enter Time in years4
--------MENU--------
Press 1 for SIMPLE Interest
Press 2 for COMPOUND Interest
Enter your OPTION (1 or 2)1
You have opted SIMPLE INTEREST
INTEREST:                              500.00
TOTAL AMOUNT after Interest:          5500.00
```

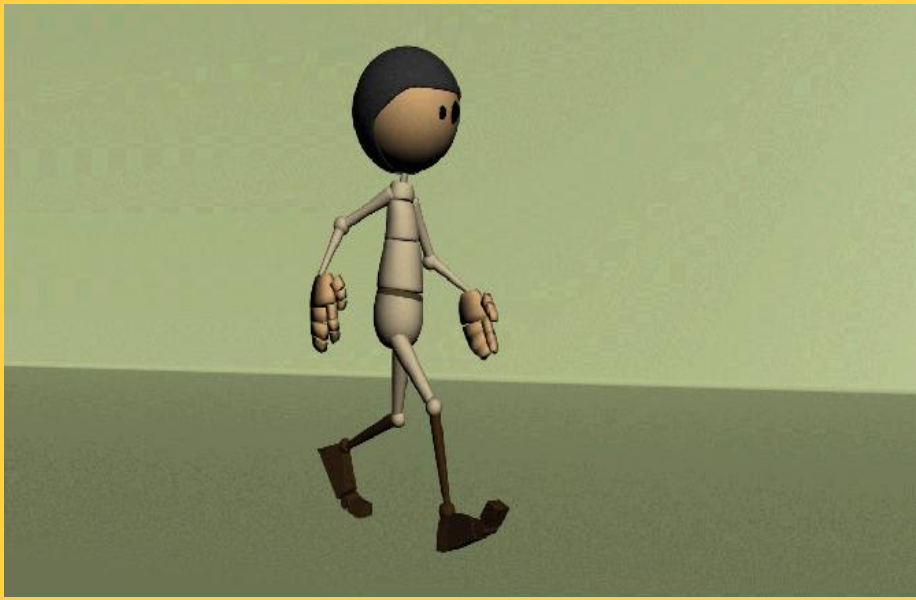
ITERATIVE CONSTRUCT

or

LOOPING CONSTRUCT

Following are the Looping constructs:-

1. For loop
2. While loop



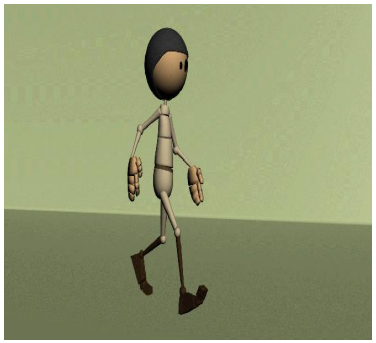
TYPES OF LOOPING STATEMENTS

COUNTING LOOPS:

the loops that repeat a certain number of times

Eg. *for loop*

DUMBO, move 10 times in the loop on the road

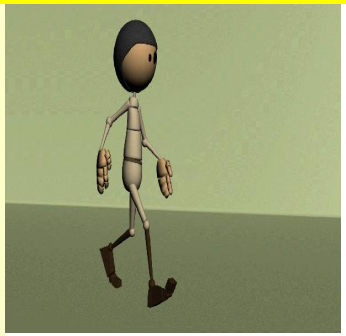


CONDITIONAL LOOPS:

the loops that repeat until a certain thing happens i.e. they keep repeating as long as the condition is *true*

Eg. *while loop*

DUMBO, move in the loop on the road till you meet JERRY



The range() function

- The range() function in Python generates a list which is a special sequence type.
- A sequence in Python is a succession of values bound together by a single name.
- Some Python sequence types/ iterables are:

  strings, lists, tuples etc.

The range() function is used with for loop in Python.

The working of range() function

range(<lower limit>,<upper limit>)

The function in the form range(L,U) will produce a list having values starting from L,L+1,L+2 ……..(U-1)

 # L and U being integers

 # both limits should be integers

Examples of range() function

Command	OUTPUT
range(0,5)	[0,1,2,3,4]
range(5,0)	Empty list [ ]
range(12,18)	[12,13,14,15,16,17]

Default step value in values is +1

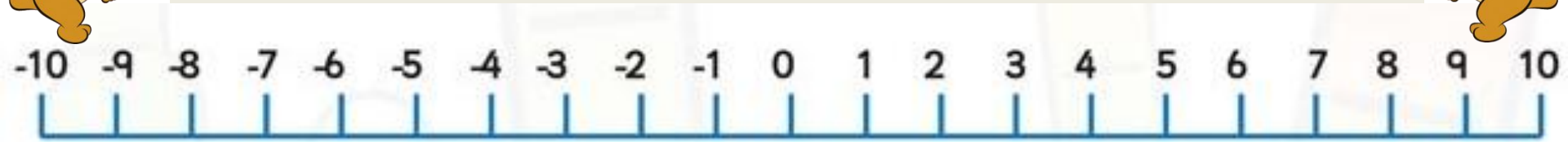
Variation in range() function

range(<lower limit>,<upper limit>,<step value>)

all value should be integers

range(L,U,s)

will produce a list having values as L,L+s,L+2s,….<=U-1



STATEMENT	VALUES GENERATED / OUTPUT
range(10)	0,1,2,3,4,5,6,7,8,9
range(5,10)	5,6,7,8,9
range(3,7)	3,4,5,6
range(5,15,3)	5,8,11,14
range(9,3,-1)	9,8,7,6,5,4
range(10,1,-2)	10,8,6,4,2

The for Loop

This determines how many times the loop will get repeated

for <variable> in <sequence> :

statements to repeat-BODY

Colon : must be placed here

Body of the loop (statements to be repeated)

NOTE on for loop :

The loop variable contains the highest value of the list after the for loop is over.

```
#Printing Series / Range of numbers
#NOTE:(Default start value in range is 0)
for i in range(10):
    print(i)
```

Control enters the loop 10 times till the range gets over and loop terminates

OUTPUT

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

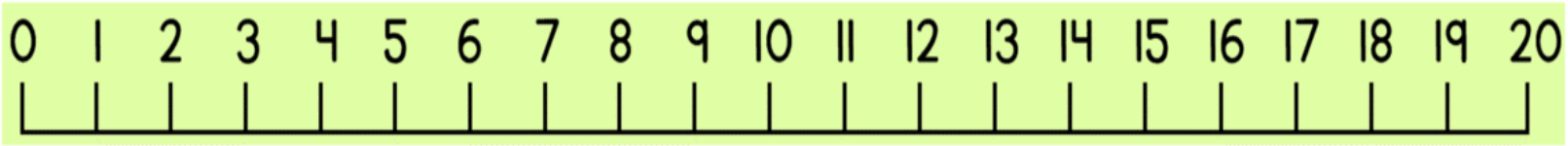
```
for i in range(10):
    print(i,end=' ')
```

OUTPUT

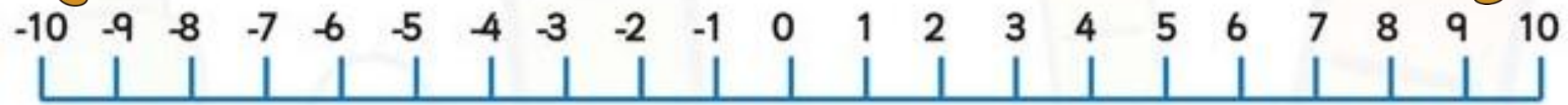
```
0 1 2 3 4 5 6 7 8 9
```

NOTE: If **end=' '** is removed , then the output will be shown in vertical order by default.

PROGRAM CODE	OUTPUT
`for i in range(1,10):` `  print(i,end=' ')`	1 2 3 4 5 6 7 8 9
`for i in range(1,11):` `  print(i,end=' ')`	1 2 3 4 5 6 7 8 9 10
`for i in range(1,11,2):` `  print(i,end=' ')`	1 3 5 7 9
`for i in range(0,11,2):` `  print(i,end=' ')`	0 2 4 6 8 10
`for i in range(25,60,5):` `  print(i,end=' ')`	25 30 35 40 45 50 55
`for i in range(3,12):` `  print(i,end=' ')`	3 4 5 6 7 8 9 10 11



PROGRAM CODE	OUTPUT
`for i in range(10,0,-1):` `  print(i,end=' ')`	10 9 8 7 6 5 4 3 2 1
`for i in range(9,-8,-1):` `  print(i,end=' ')`	9 8 7 6 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7
`for i in range(9,-8,-3):` `  print(i,end=' ')`	9 6 3 0 -3 -6
`for i in range(75,28,-5):` `  print(i,end=' ')`	75 70 65 60 55 50 45 40 35 30
`for i in range(600,250,-50):` `  print(i,end=' ')`	600 550 500 450 400 350 300



PROGRAM CODE	OUTPUT
`for i in [10,25,30]:` `  print(i,i*i)`	10 100 25 625 30 900
`for i in [10,'Hi',25]:` `  print(i*3)`	30 HiHiHi 75

`P=[10,'Hi',25]`
`for i in P:`
`  print(i*3)`
	30
	HiHiHi
	75

`P=(9,'Hi',4,'India')`
`for i in P:`
`  print(i*3)`
	27
	HiHiHi
	12
	IndiaIndiaIndia

`for i in (9,'Hi',4,'India'):` `  print(i*3)`	27 HiHiHi 12 IndiaIndiaIndia

PROGRAM CODE	OUTPUT

`Str='Stay happy'`
`for i in Str:`
`  print(i*4)`
	ssss
	tttt
	aaaa
	yyyy
	
	hhhh
	aaaa
	pppp
	pppp
	yyyy

`for i in "Stay happy":` `  print(i*4)`	ssss tttt aaaa yyyy hhhh aaaa pppp pppp yyyy

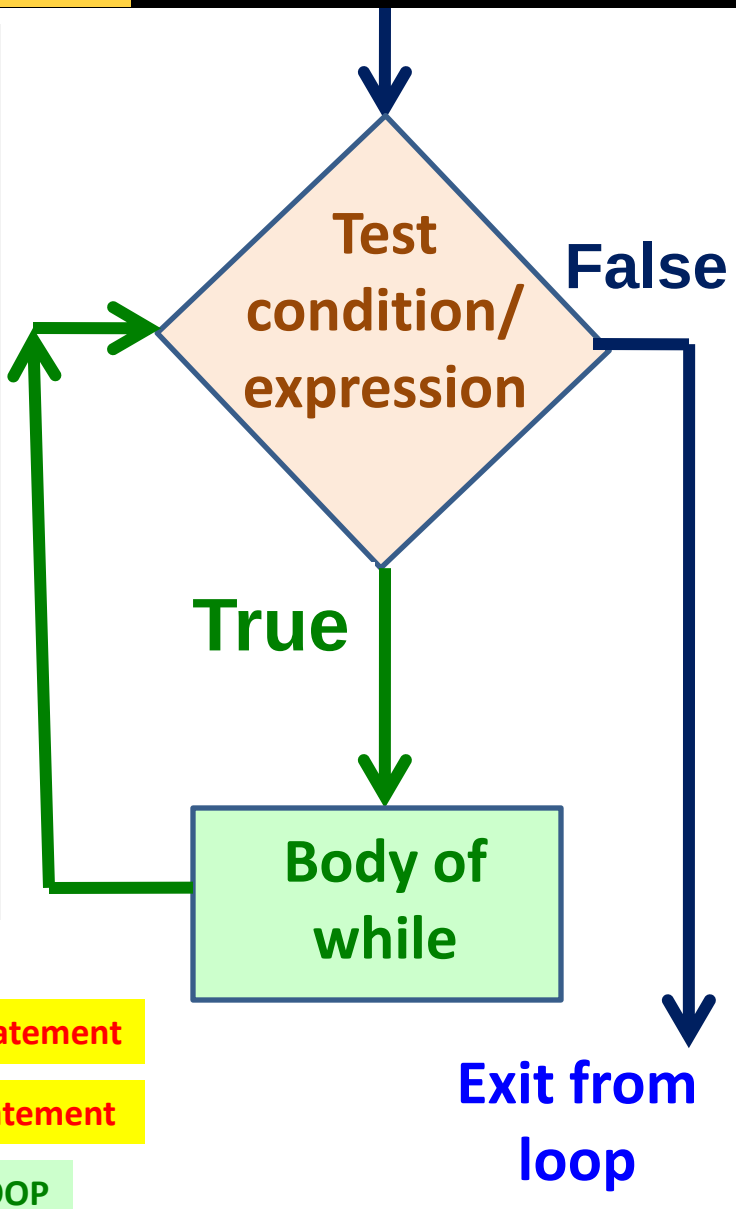
While Loop

A while loop is a conditional loop that will repeat the instructions within itself as long as a condition remains ***true*** .

while <logicalexpression> :

loop body

Where the loop-body may contain a single statement or multiple statements or an empty statement (i.e. pass statement). The loop iterates while the logical expression evaluates to true. When the expression becomes false, the program control passes to the line after the loop-body.



EXAMPLE:

```
i=1
while i<=10:
    print(i,end=' ')
    i=i+1
```

INITIALIZATION Statement

CONDITION Statement

BODY OF THE LOOP

UPDATION Statement

PROGRAM CODE

```
#SERIES 1 2 3 4 5 6 7 8 9 10
i=1
while i<=10:
    print(i,end=' ')
    i=i+1
```

```
#ALTERNATE SERIES 1 3 5 7 9
i=1
while i<=10:
    print(i,end=' ')
    i=i+2
print('Thank you. i=',i)
```

```
#REVERSE SERIES 10 9 8 7 6 5 4 3 2 1
i=10
while i>=1:
    print(i,end=' ')
    i-=1    #augmented assignment i=i-1
```

```
#REVERSE series 10 7 4 1
i=10
while i>=1:
    print(i,end=' ')
    i=i-3
```

OUTPUT

```
1 2 3 4 5 6 7 8 9 10
```

```
1 3 5 7 9 Thank you. i= 11
```

```
10 9 8 7 6 5 4 3 2 1
```

```
10 7 4 1
```

TASK: Write codes for the following series in IDE ??

SERIES :	Using for loop	Using while loop
1 2 3 4 5 6…….n		
-30 -27 -24 ….. 0		
5 10 15 20…..200		
1 -4 7 -10 13 ….. -40		
1 4 7 10 13 … 40		
$S=1+x+x^2+x^3+x^4+\ldots..x^n$		

PROGRAM: Sum of n Natural Numbers

```
''' SUM OF n NATURAL NUMBERS
Eg.
sum=0, n=5(given by user)
--------------
sum=sum+i

____________
sum=0+1=1
sum=1+2=3
sum=3+3=6
sum=6+4=10
sum=10+5=15
'''
#-----------------------------
n=int(input('Enter a number'))
sum=0
for i in range(1,n+1):
    sum=sum+i
print('SUM=',sum)
```

```
Enter a number6
SUM= 21
```

PROGRAM: Factorial of a Number

```
'''FACTORIAL OF A NO. is the same as
PRODUCT OF n NATURAL NUMBERS
f=1
n=5
--------------
f=f*i
--------------
f=1*1=1
f=1*2=2
f=2*3=6
f=6*4=24
f=24*5=120
f=120*6=720
'''
#-----------------------------------
n=int(input('Enter a number'))
fact=1
for i in range(1,n+1):
    fact=fact*i
print('FACTORIAL OF THE NO.',n,'=',fact)
```

```
Enter a number5
FACTORIAL OF THE NO. 5 = 120
```

PROGRAM: Find the REVERSE of a no. and check whether it is a PALINDROME or not.

```
#PROGRAM TO REVERSE A NO. and
#check whether the no. is PALINDROME or not
#Eg.    Number=25852      Reverse=25852
#Eg.     Number=4774      Reverse=4774
#Number==Reverse, then it is a palindrome
# Eg. Number=159          Reverse=951
#Number!=Reverse, then it is NOT a palindrome
#-----------------------------------------
n=int(input('Enter a no.'))
num=n
rev=0
while num!=0:
    d=num%10
    rev=rev*10+d
    num=num//10
print("The Reverse of the number",n," is ",rev)
if rev==n:
    print("The number is a PALINDROME")
else:
    print("The number is NOT a PALINDROME")
```

```
Enter a no.45654
The Reverse of the number 45654  is  45654
The number is a PALINDROME
```

```
Enter a no.345
The Reverse of the number 345  is  543
The number is NOT a PALINDROME
```

PROGRAM: Print FIBONACCI SERIES to n no. of Terms

```
#PROGRAM TO PRINT a Fibonacci series
# 0 1 1 2 3 5 8 13 21 34 .....
a=0
b=1
n=int(input('Enter no. of terms'))
print(a,end=' ')
print(b,end=' ')
for i in range(1,n-1):
    c=a+b
    print(c,end=' ')
    a,b=b,c
```

```
Enter no. of terms8
0 1 1 2 3 5 8 13
```

PROGRAM: MULTIPLICATION TABLE of a Number

```
# MULTIPLICATION TABLE
n=int(input("Enter the no. whose table u want"))
for i in range(1,11):
    print(n,'x',i,'=',n*i)
```

```
Enter the no. whose table u want5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

JUMP statements

Python offers two types of jump statements to be used within loops to jump out of loop iterations.

i. break

Break statements are the jump statements which terminates the very loop it lies within and skips over a part of the code(i.e. rest of the loop) and jumps over to the statement following the loop.

(FORCEFUL termination of the loop)

ii. continue

Continue is the jump statement which forces the next iteration of the loop to take place and skips the rest of the loop statements.

(FORCEFUL re-iteration of the loop)

The working of a break statement

```
while <test-condition> :
    statement 1
    if <condition> :
        break
    statement 2
    statement 3
statement 4
statement 5
```

```
for <var> in <sequence> :
    statement 1
    if <condition> :
        break
    statement 2
    statement 3
statement 4
statement 5
```

The working of a continue statement

```
while <test-condition> :
    statement 1
    if <condition> :
        continue
    statement 2
    statement 3
statement 4
statement 5
```

```
for <var> in <sequence> :
    statement 1
    if <condition> :
        continue
    statement 2
    statement 3
statement 4
statement 5
```

Program to illustrate the difference between break and continue

PROGRAM

```
print('Loop with break')
for i in range(1,13):
    if i%4==0:
        break
    else:
        print(i)
print('------------------')
print('Loop with continue')
for i in range(1,13):
    if i%4==0:
        continue
    else:
        print(i)
```

OUTPUT

```
Loop with break
1
2
3
------------------
Loop with continue
1
2
3
5
6
7
9
10
11
```

HINT:

i) When a smaller no. is divided by a greater no. to calculate its remainder(modulo %) then the smaller is the answer as its remainder. For eg. 1%4 gives 1 as output.

ii) **Break** brings the control out of the loop.

iii) **Continue** takes the control for the next iteration.

Loop *Else* clause

The ***else clause of a Python loop*** executes when the loop terminates normally,i.e., when test condition results into *false* for a **while loop** or **for loop** has executed for the last value in the sequence; and *not when the* ***break*** *statement* terminates the loop.

NOTE:

- ***The else clause of a loop appears at the same indentation as that of loop keyword while or for.***
- ***The loop else suite executes only in the normal termination of loop.***
- ***Loop else clause works when the loop is terminating normally- after the last element of sequence in for loop and when the test-condition becomes false in while loop.***

```
#Program to check whether a number entered is PRIME or not
n=int(input('Enter a positive no.'))
for i in range(2,(n//2)+1):
    if n%i==0:
        print("No. is COMPOSITE")
        break
else:
    print("The no. is PRIME")
```

```
Enter a positive no.34
No. is COMPOSITE
```

```
Enter a positive no.67
The no. is PRIME
```

PROGRAM to show the execution of Loop Else Clause

```
for i in range(1,11):
    print(i,end=' ')
else:
    print("\nThank you")
```

PROGRAM to show the skip of Loop Else Clause due to presence of break

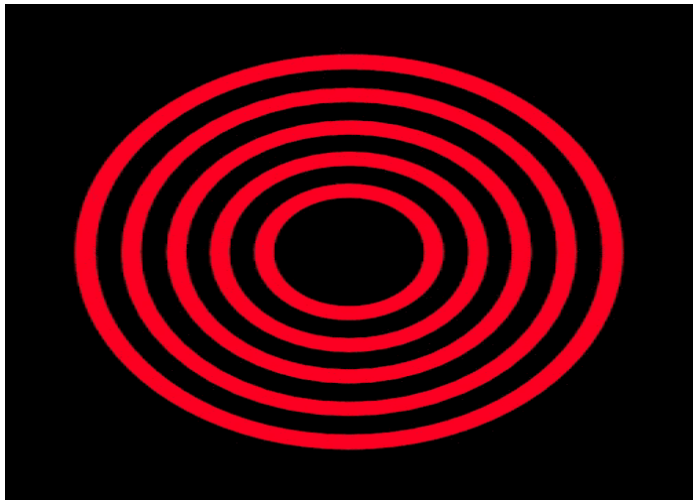
```
for i in range(1,11):
    print(i,end=' ')
    if i==5:
        break
    else:
        print("\nThank you")
```

NESTED LOOPS

A loop containing another loop inside its body is known as **nested loop.**

Note:

In a nested loop, inner loop*(or **nested loop**)* must be terminated before the outer loop*(**enclosing loop**)*.



```
for i in range(1,6):
    for j in range(1,i+1):
        print(i,end=' ')
    print()
```

NESTED LOOP / INNER LOOP

ENCLOSING LOOP/ OUTER LOOP

NOTE: ***First time when the control enters the outer loop, then with that value it enters the inner loop. Until the inner loop doesn't get over, the outer loop does not take the next value from the range of outer loop. Again, when it enters the outer loop with next value from its range, then with that value it starts the inner loop afresh and continues inner loop till its range gets over. This process is repeated until the outer loop gets exhausted with all its values in the sequence/range ,except any accidental termination(i.e. encountering break statement)***

PROGRAM CODE	OUTPUT

`for i in range(1,6):`
`  for j in range(1,6):`
`  print(j,end=' ')`
`  print()`
	1 2 3 4 5
	1 2 3 4 5
	1 2 3 4 5
	1 2 3 4 5
	1 2 3 4 5

`for i in range(1,6):`
`  for j in range(1,6):`
`  print(i,end=' ')`
`  print()`
	1 1 1 1 1
	2 2 2 2 2
	3 3 3 3 3
	4 4 4 4 4
	5 5 5 5 5

`for i in range(5,0,-1):`
`  for j in range(5,0,-1):`
`  print(i,end=' ')`
`  print()`
	5 5 5 5 5
	4 4 4 4 4
	3 3 3 3 3
	2 2 2 2 2
	1 1 1 1 1

PROGRAM CODE	OUTPUT

```
for i in range(1,6):
    for j in range(1,i+1):
        print(i,end=' ')
    print()
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
for i in range(1,6):
    for j in range(1,i+1):
        print(j,end=' ')
    print()
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
for i in range(1,6):
    for j in range(1,6):
        print('*',end=' ')
    print()
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
for i in range(1,6):
    for j in range(1,i+1):
        print('$',end=' ')
    print()
```

```
$
$ $
$ $ $
$ $ $ $
$ $ $ $ $
```

PROGRAM CODE

```
k=65
for i in range(1,7):
    for j in range(1,i+1):
        print(chr(k),end='  ')
    k=k+2
    print()
```

OUTPUT

```
A
C  C
E  E  E
G  G  G  G
I  I  I  I  I
K  K  K  K  K  K
```

```
k=65
for i in range(1,6):
    for j in range(1,i+1):
        print(chr(k),end='  ')
    k=k+1
    print()
k=69
for i in range(1,6):
    for j in range(5,i-1,-1):
        print(chr(k),end='  ')
    k=k-1
    print()
```

```
A
B  B
C  C  C
D  D  D  D
E  E  E  E  E
E  E  E  E  E
D  D  D  D
C  C  C
B  B
A
```

NOTE:

ord() function is used to get the ASCII value of a character

chr() function is used to get the equivalent character of the ASCII no.

NOTE:

ASCII stands for American Standard Code for Information Interchange.

```
>>> ord('a')
97
>>> chr(65)
'A'
```