UNIT III
CONTROL FLOW, FUNCTIONS

Conditionals: Boolean values and operators, conditional (if), alternative (if-else), chained conditional (if-elif-else); Iteration: state, while, for, break, continue, pass; Fruitful functions: return values, parameters, scope: local and global, composition, recursion; Strings: string slices, immutability, string functions and methods, string module; Lists as arrays. Illustrative programs: square root, gcd, exponentiation, sum the array of numbers, linear search, binary search.

BOOLEAN VALUES:

Boolean:

- Boolean data type have two values. They are 0 and 1.
- 0 represents False
- 1 represents True
- True and False are keyword.

Example:

```
>>> 3==5
False
>>> 6==6
True
>>> True+True
2
>>> False+True
1
>>> False*True
0
```

Boolean expressions

A **boolean expression** is an expression that is either true or false. The following examples use the operator ==, which compares two operands and produces True if they are equal and False otherwise:

```
>>> 5 == 5
True
>>> 5 == 6
False
```

True and False are special values that belong to the type bool; they are not strings:

```
>>> type(True)
<type 'bool'>
>>> type(False)
<type 'bool'>
```

OPERATORS:

- Operators are the constructs which can manipulate the value of operands.
- Consider the expression *4 + 5 = 9.* Here, 4 and 5 are called operands and + is called operator.

Types of Operators:

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

Arithmetic operators:

They are used to perform mathematical operations like addition, subtraction, multiplication etc.

Operator	Description	Example a=10,b=20
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed	5//2=2

Comparison (Relational) Operators:

- Comparison operators are used to compare values.
- It either returns True or False according to the condition.

Operator	Description	Example a=10,b=20
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a!=b) is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Assignment Operators:

Assignment operators are used in Python to assign values to variables.

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a

*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Logical Operators:

Logical operators are and, or, not operators.

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Bitwise Operators:

Let x = 10 (0000 1010 in binary) and y = 4 (0000 0100 in binary)

Operator	Meaning	Example
&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

Membership Operators:

- Evaluates to find a value or a variable is in the specified sequence of string, list, tuple, dictionary or not.
- To check particular element is available in the list or not.
- Operators are in and not in.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Example:

```
x=[5,3,6,4,1]
>>> 5 in x
True
>>> 5 not in x
False
```

Identity Operators:

They are used to check if two values (or variables) are located on the same part of the memory.

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

Example

```
x = 5
y = 5
a = 'Hello'
b = 'Hello'
print(x is not y) // False
print(a is b)//True
```

CONDITIONALS

- Conditional if
- Alternative execution- if... else
- Chained if...elif...else
- Nested if....else
- Inline if

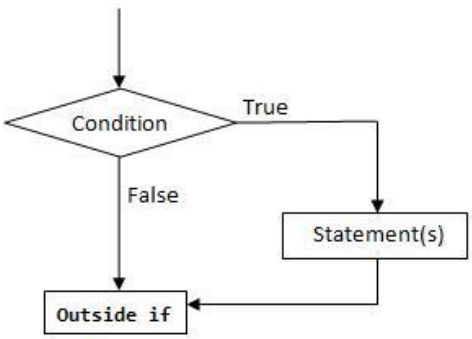
Conditional (if):

Conditional (if) is used to test a condition, if the condition is true the statements inside if will be executed.

syntax:

```
if(condition 1):
    Statement 1
```

Flowchart:



Example:

1. Program to provide flat rs 500, if the purchase amount is greater than 2000.
2. Program to provide bonus mark if the category is sports.

Program to provide flat rs 500, if the purchase amount is greater than 2000.	**output**
purchase=eval(input("enter your purchase amount")) if(purchase>=2000): purchase=purchase-500 print("amount to pay",purchase)	enter your purchase amount 2500 amount to pay 2000
Program to provide bonus mark if the category is sports	**output**
m=eval(input("enter ur mark out of 100")) c=input("enter ur categery G/S") if(c=="S"): m=m+5 print("mark is",m)	enter ur mark out of 100 85 enter ur categery G/S S mark is 90

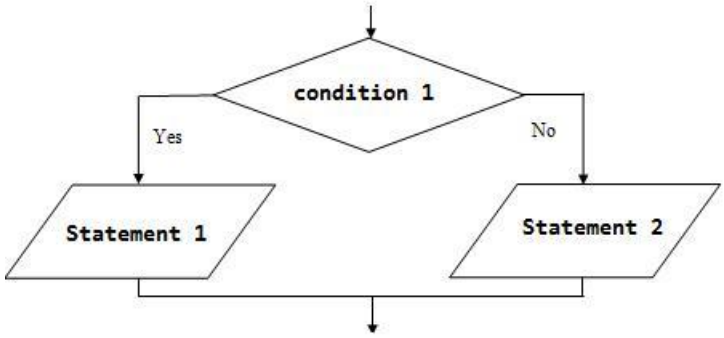
Alternative execution (if-else)

In the alternative the condition must be true or false. In this **else** statement can be combined with **if** statement. The **else** statement contains the block of code that executes when the condition is false. If the condition is true statements inside the if get executed otherwise else part gets executed. The alternatives are called branches, because they are branches in the flow of execution.

syntax:

```
if(condition 1):
   Statement 1
else:
   Statement 2
```

Flowchart:



Examples:

1. odd or even number
2. positive or negative number
3. leap year or not
4. greatest of two numbers
5. eligibility for voting

Odd or even number	**Output**

n=eval(input("enter a number"))
if(n%2==0):
  print("even number")
else:
  print("odd number")
	enter a number4
	even number

positive or negative number	**Output**

n=eval(input("enter a number"))
if(n>=0):
  print("positive number")
else:
  print("negative number")
	enter a number8
	positive number

leap year or not	**Output**

y=eval(input("enter a yaer"))
if(y%4==0):
  print("leap year")
else:
  print("not leap year")
	enter a yaer2000
	leap year

greatest of two numbers	**Output**

a=eval(input("enter a value:"))
b=eval(input("enter b value:"))
if(a>b):
  print("greatest:",a)
else:
  print("greatest:",b)
	enter a value:4
	enter b value:7
	greatest: 7

eligibility for voting	**Output**

age=eval(input("enter ur age:"))
if(age>=18):
  print("you are eligible for vote")
else:
  print("you are eligible for vote")
	enter ur age:78
	you are eligible for vote

Chained conditionals(if-elif-else)

- The elif is short for else if.
- This is used to check more than one condition.
- If the condition1 is False, it checks the condition2 of the elif block. If all the conditions are False, then the else part is executed.

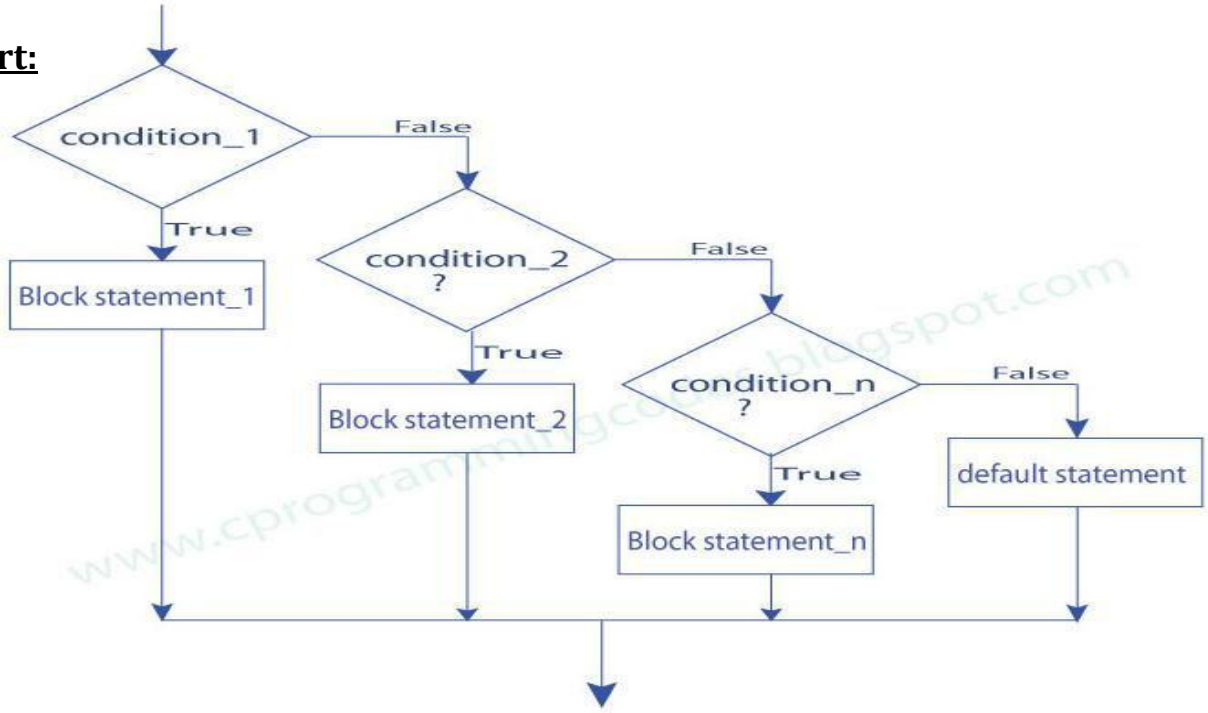
- Among the several if...elif...else part, only one part is executed according to the condition.

- The if block can have only one else block. But it can have multiple elif blocks.
- The way to express a computation like that is a chained conditional.

syntax:

```
if(condition 1):
     statement 1
elif(condition 2):
     statement 2
elif(condition 3):
     statement 3
else:
    default statement
```

Flowchart:



Example:

1. student mark system
2. traffic light system
3. compare two numbers
4. roots of quadratic equation

student mark system	**Output**

mark=eval(input("enter ur mark:"))
if(mark>=90):
  print("grade:S")
elif(mark>=80):
  print("grade:A")
elif(mark>=70):
  print("grade:B")
elif(mark>=50):
  print("grade:C")
else:
  print("fail")
	enter ur mark:78
	grade:B

traffic light system	**Output**

colour=input("enter colour of light:")
if(colour=="green"):
  print("GO")
elif(colour=="yellow"):
  print("GET READY")
else:
  print("STOP")
	enter colour of light:green
	GO

compare two numbers	**Output**

x=eval(input("enter x value:"))
y=eval(input("enter y value:"))
if(x == y):
  print("x and y are equal")
elif(x < y):
  print("x is less than y")
else:
  print("x is greater than y")
	enter x value:5
	enter y value:7
	x is less than y

Roots of quadratic equation	**output**

a=eval(input("enter a value:"))
b=eval(input("enter b value:"))
c=eval(input("enter c value:"))
d=(b*b-4*a*c)
if(d==0):
  print("same and real roots")
elif(d>0):
  print("diffrent real roots")
else:
  print("imaginagry roots")
	enter a value:1
	enter b value:0
	enter c value:0
	same and real roots

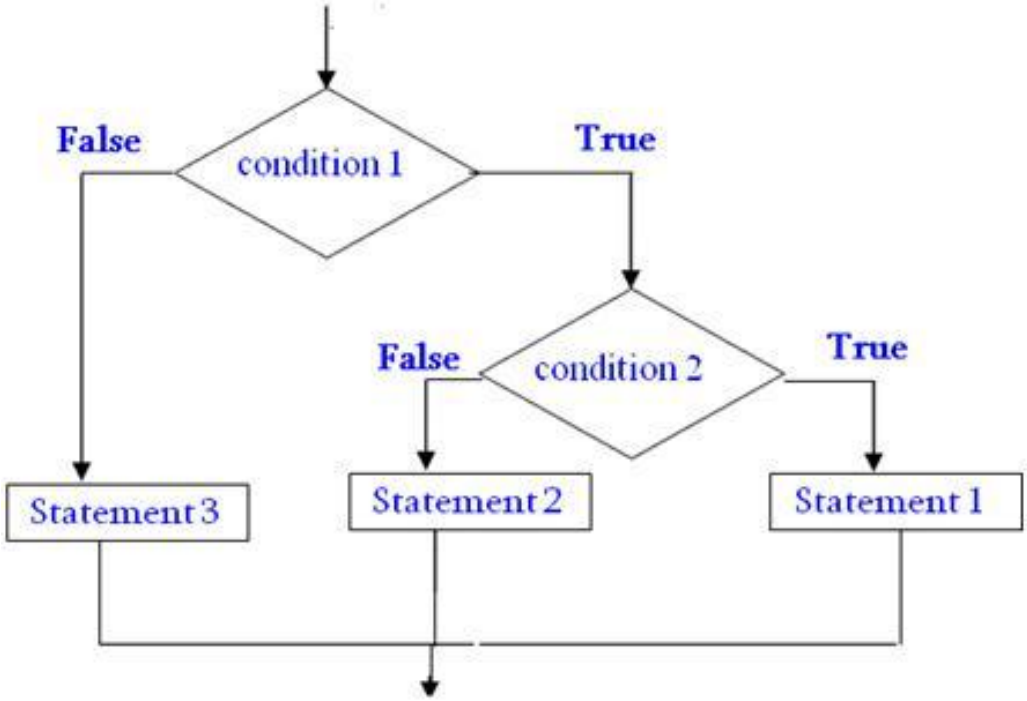
Nested conditionals

One conditional can also be nested within another. Any number of condition can be nested inside one another. In this, if the condition is true it checks another if condition1. If both the conditions are true statement1 get executed otherwise statement2 get execute. if the condition is false statement3 gets executed

Syntax:

```
if (condition):
        if(condition 1):
            statement 1
        else:
            statement 2
else:
          statement 3
```

Flowchart:



Example:

1. greatest of three numbers
2. positive negative or zero

greatest of three numbers	**output**

a=eval(input("enter the value of a"))
b=eval(input("enter the value of b"))
c=eval(input("enter the value of c"))
if(a>b):
  if(a>c):
  print("the greatest no is",a)
  else:
else:
  if(b>c):
  print("the greatest no is",b)
  else:
  print("the greatest no is",c)
	enter the value of a 9
	enter the value of a 1
	enter the value of a 8
	the greatest no is 9

positive negative or zero	**output**

n=eval(input("enter the value of n:"))
if(n==0):
  print("the number is zero")
else:
  if(n>0):
  print("the number is positive")
  else:
  print("the number is negative")
	enter the value of n:-9
	the number is negative

Inline if:

An inline if statement is a simpler form of if statement and is more convenient ,if we need to perform simple task.

Syntax: do **task A** if condition is true else do **task B**

Example:

```
>>> b=True
>>> a=1 if b else None
>>> a
1
>>> b=False
>>> a=1 if b else None
>>> a
#None
```

ITERATION/CONTROL STATEMENTS/LOOPs:

- state
- while
- for
- break
- continue
- pass

State:

Transition from one process to another process under specified condition with in a time is called state.

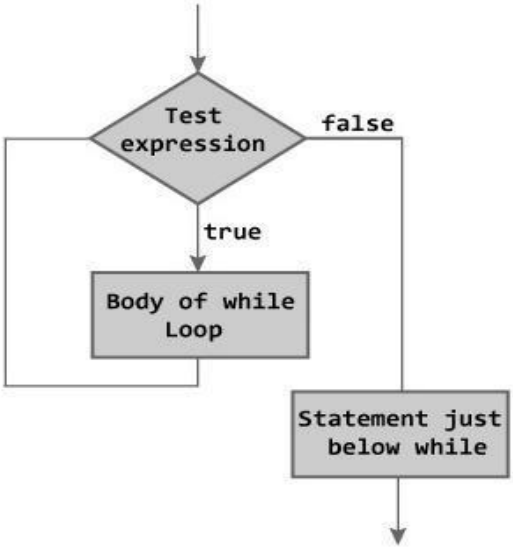
While loop:

- While loop statement in Python is used to repeatedly executes set of statement as long as a given condition is true.
- In while loop, test expression is checked first. The body of the loop is entered only if the test_expression is True. After one iteration, the test expression is checked again. This process continues until the test_expression evaluates to False.
- In Python, the body of the while loop is determined through indentation.
- The statements inside the while starts with indentation and the first unindented line marks the end.

Syntax:

```
inital value
while(condition):
    body of while loop
    increment
```

Flowchart:



Examples:

1. program to find sum of n numbers:
2. program to find factorial of a number
3. program to find sum of digits of a number:
4. Program to Reverse the given number:
5. Program to find number is Armstrong number or not
6. Program to check the number is palindrome or not

Sum of n numbers:	**output**

n=eval(input("enter n"))
i=1
sum=0
while(i<=n):
  sum=sum+i
  i=i+1
print(sum)
	enter n
	10
	55

Factorial of a numbers:	**output**

n=eval(input("enter n"))
i=1
fact=1
while(i<=n):
  fact=fact*i
  i=i+1
print(fact)
	enter n
	5
	120

Sum of digits of a number:	**output**

n=eval(input("enter a number"))
sum=0
while(n>0):
  a=n%10
  sum=sum+a
  n=n//10
print(sum)
	enter a number
	123
	6

Reverse the given number:	**output**

n=eval(input("enter a number"))
sum=0
while(n>0):
  a=n%10
  sum=sum*10+a
  n=n//10
print(sum)
	enter a number
	123
	321

Armstrong number or not	output

n=eval(input("enter a number"))
org=n
sum=0
while(n>0):
  a=n%10
  sum=sum+a*a*a
  n=n//10
if(sum==org):
  print("The given number is Armstrong number")
else:
  print("The given number is not Armstrong number")
	enter a number153
	The given number is Armstrong number

Palindrome or not	output

n=eval(input("enter a number"))
org=n
sum=0
while(n>0):
  a=n%10
  sum=sum*10+a
  n=n//10
if(sum==org):
  print("The given no is palindrome")
else:
  print("The given no is not palindrome")
	enter a number121
	The given no is palindrome

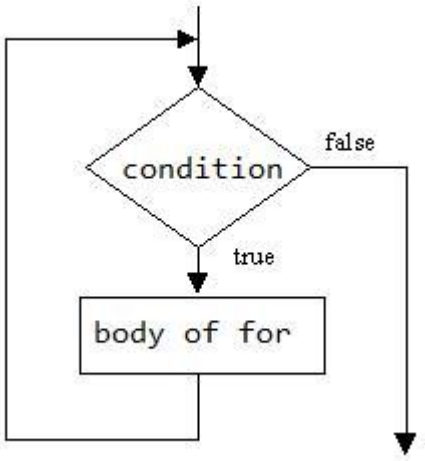
For loop:

- ***for in range:***
  - We can generate a sequence of numbers using range() function. range(10) will generate numbers from 0 to 9 (10 numbers).
  - In range function have to define the start, stop and step size as range(start,stop,step size). step size defaults to 1 if not provided.

syntax

```
for i in range(start,stop,steps):
    body of for loop
```

Flowchart:



For in sequence

- The for loop in Python is used to iterate over a sequence (list, tuple, string). Iterating over a sequence is called traversal. Loop continues until we reach the last element in the sequence.
- The body of for loop is separated from the rest of the code using indentation.

```
for i in sequence:
    print(i)
```

Sequence can be a list, strings or tuples

s.no	sequences	example	output
1.	For loop in string	for i in "Ramu": print(i)	R A M U
2.	For loop in list	for i in [2,3,5,6,9]: print(i)	2 3 5 6 9
3.	For loop in tuple	for i in (2,3,1): print(i)	2 3 1

Examples:

1. print nos divisible by 5 not by 10:
2. Program to print fibonacci series.
3. Program to find factors of a given number
4. check the given number is perfect number or not
5. check the no is prime or not
6. Print first n prime numbers
7. Program to print prime numbers in range

print nos divisible by 5 not by 10	**output**

n=eval(input("enter a"))
for i in range(1,n,1):
  if(i%5==0 and i%10!=0):
  print(i)
	enter a:30
	5
	15
	25

Fibonacci series	**output**

a=0
b=1
n=eval(input("Enter the number of terms: "))
print("Fibonacci Series: ")
print(a,b)
for i in range(1,n,1):
  c=a+b
  print(c)
  a=b
  b=c
	Enter the number of terms: 6
	Fibonacci Series:
	0 1
	1
	2
	3
	5
	8

find factors of a number	**Output**

n=eval(input("enter a number:"))
for i in range(1,n+1,1):
  if(n%i==0):
  print(i)
	enter a number:10
	1
	2
	5
	10

check the no is prime or not	**output**

n=eval(input("enter a number"))
for i in range(2,n):
  if(n%i==0):
  print("The num is not a prime")
  break
else:
  print("The num is a prime number.")
	enter a no:7
	The num is a prime number.

check a number is perfect number or not	**Output**

n=eval(input("enter a number:"))
sum=0
for i in range(1,n,1):
  if(n%i==0):
  sum=sum+i
if(sum==n):
  print("the number is perfect number")
else:
  print("the number is not perfect number")
	enter a number:6
	the number is perfect number

Program to print first n prime numbers	**Output**

number=int(input("enter no of prime numbers to be displayed:"))
count=1
n=2
while(count<=number):
  for i in range(2,n):
  if(n%i==0):
  break
  else:
  print(n)
  count=count+1
  n=n+1
	enter no of prime numbers to be displayed:5
	2
	3
	5
	7
	11

Program to print prime numbers in range	**output:**

lower=eval(input("enter a lower range"))
upper=eval(input("enter a upper range"))
for n in range(lower,upper + 1):
  if n > 1:
  for i in range(2,n):
  if (n % i) == 0:
  break
  else:
  print(n)
	enter a lower range50
	enter a upper range100
	53
	59
	61
	67
	71
	73
	79
	83
	89
	97

Loop Control Structures

BREAK

- Break statements can alter the flow of a loop.
- It terminates the current
- loop and executes the remaining statement outside the loop.
- If the loop has else statement, that will also gets terminated and come out of the loop completely.

Syntax:

break

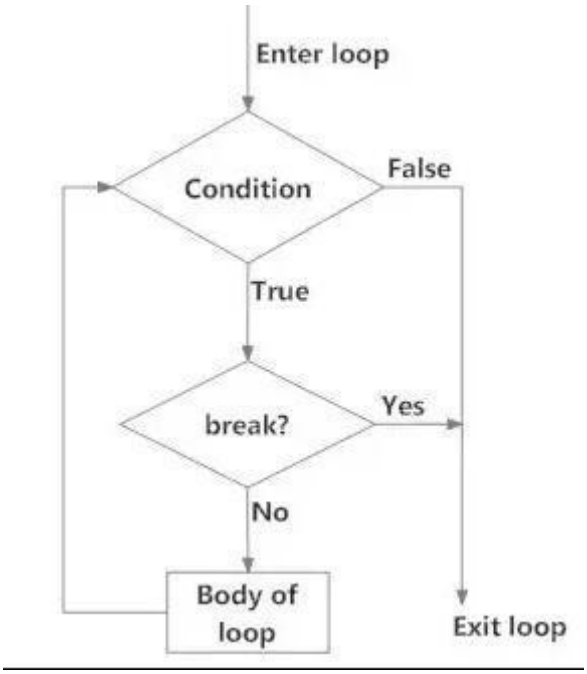
```
while (test Expression):

    // codes
    if (condition for break):

        break

    // codes
```

Flowchart



example	Output

for i in "welcome":
  if(i=="c"):
  break
  print(i)
	w
	e
	l

CONTINUE

It terminates the current iteration and transfer the control to the next iteration in the loop.

Syntax: Continue

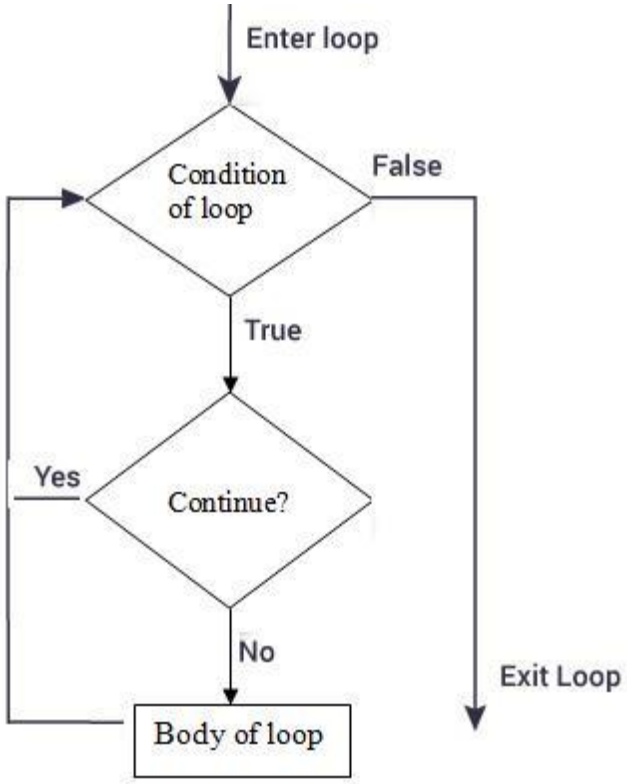
```
while (test Expression):

    // codes
    if (condition for continue):

        continue

    // codes
```

Flowchart



Example:	Output

for i in "welcome":
  if(i=="c"):
  continue
  print(i)
	w
	e
	l
	o
	m
	e

PASS

- It is used when a statement is required syntactically but you don't want any code to execute.
- It is a null statement, nothing happens when it is executed.

Syntax:

pass

break

Example	Output

for i in “welcome”:
  if (i == “c”):
  pass
  print(i)
	w
	e
	l
	c
	o
	m
	e

Difference between break and continue

break	continue
It terminates the current loop and executes the remaining statement outside the loop.	It terminates the current iteration and transfer the control to the next iteration in the loop.
syntax: break	syntax: continue

for i in "welcome":
  if(i=="c"):
  break
  print(i)
	for i in "welcome":
	  if(i=="c"):
	  continue
	  print(i)

w e l	w e l o m e

else statement in loops:

else in for loop:

- If else statement is used in for loop, the else statement is executed when the loop has reached the limit.
- The statements inside for loop and statements inside else will also execute.

example	output

for i in range(1,6):
  print(i)
else:
  print("the number greater than 6")
	1
	2
	3
	4
	5 the number greater than 6

else in while loop:

- If else statement is used within while loop , the else part will be executed when the condition become false.
- The statements inside for loop and statements inside else will also execute.

Program	output

i=1
while(i<=5):
  print(i)
  i=i+1
else:
  print("the number greater than 5")
	1
	2
	3
	4
	5
	the number greater than 5

Fruitful Function

- Fruitful function
- Void function
- Return values
- Parameters
- Local and global scope
- Function composition
- Recursion

Fruitful function:

A function that returns a value is called fruitful function.

Example:

```
Root=sqrt(25)
```

Example:

```
def add():
    a=10
    b=20
    c=a+b
    return c
c=add()
print(c)
```

Void Function

A function that perform action but don't return any value.

Example:

```
print("Hello")
```

Example:

```
def add():
    a=10
    b=20
```

```
    c=a+b
    print(c)
add()
```

Return values:

return keywords are used to return the values from the function.

example:

return a – return 1 variable

return a,b– return 2 variables

return a,b,c– return 3 variables

return a+b– return expression

return 8– return value

PARAMETERS / ARGUMENTS:

- Parameters are the variables which used in the function definition. Parameters are inputs to functions. Parameter receives the input from the function call.
- It is possible to define more than one parameter in the function definition.

Types of parameters/Arguments:

1. Required/Positional parameters
2. Keyword parameters
3. Default parameters
4. Variable length parameters

Required/ Positional Parameter:

The number of parameter in the function definition should match exactly with number of arguments in the function call.

Example	Output:
def student(name, roll): print(name,roll) student("George",98)	George 98

Keyword parameter:

When we call a function with some values, these values get assigned to the parameter according to their position. When we call functions in keyword parameter, the order of the arguments can be changed.

Example	Output:
def student(name,roll,mark): print(name,roll,mark) student(90,102,"bala")	90 102 bala

Default parameter:

Python allows function parameter to have default values; if the function is called without the argument, the argument gets its default value in function definition.

Example	Output:
def student(name, age=17): print (name, age) student(“kumar”): student(“ajay”):	Kumar 17 Ajay 17

Variable length parameter

- Sometimes, we do not know in advance the number of arguments that will be passed into a function.
- Python allows us to handle this kind of situation through function calls with number of arguments.
- In the function definition we use an asterisk (*) before the parameter name to denote this is variable length of parameter.

Example	Output:
def student(name,*mark): print(name,mark) student (“bala”,102,90)	bala (102 ,90)

Local and Global Scope

Global Scope

- The *scope* of a variable refers to the places that you can see or access a variable.
- A variable with global scope can be used anywhere in the program.
- It can be created by defining a variable outside the function.

Example	output

a=50
def add():
  b=20
  c=a+b
  print©
def sub():
  b=30
  c=a-b
  print©
print(a)
	
	70
	
	20
	
	50

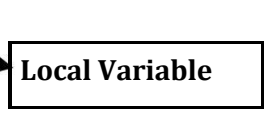
Global Variable

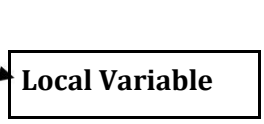
Local Variable

Local Scope A variable with local scope can be used only within the function .

Example	output

def add():
  b=20
  c=a+b
  print©
def sub():
  b=30
  c=a-b
  print©
print(a)
print(b)
	
	70
	
	20
	
	error
	error





Function Composition:

- Function Composition is the ability to call one function from within another function
- It is a way of combining functions such that the result of each function is passed as the argument of the next function.
- In other words the output of one function is given as the input of another function is known as function composition.

Example:	Output:
math.sqrt(math.log(10))	

def add(a,b):
  c=a+b
  return c
def mul(c,d):
  e=c*d
  return e
c=add(10,20)
e=mul(c,30)
print(e)
	900

find sum and average using function composition	**output**

def sum(a,b):
  sum=a+b
  return sum
def avg(sum):
  avg=sum/2
  return avg
a=eval(input("enter a:"))
b=eval(input("enter b:"))
sum=sum(a,b)
avg=avg(sum)
	enter a:4
	enter b:8
	the avg is 6.0

print("the avg is",avg)	

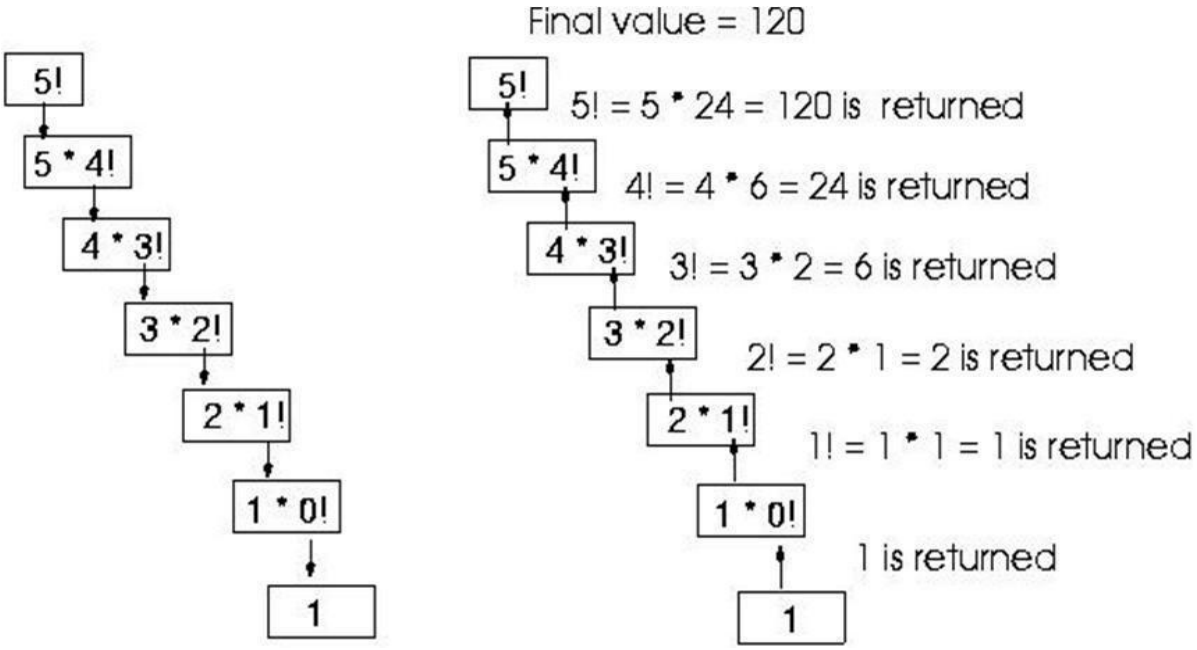
Recursion

A function calling itself till it reaches the base value - stop point of function call. Example: factorial of a given number using recursion

Factorial of n	Output

def fact(n):
  if(n==1):
  return 1
  else:
  return n*fact(n-1)
 	
n=eval(input("enter no. to find fact:"))
fact=fact(n)
print("Fact is",fact)
	enter no. to find fact:5
	Fact is 120

Explanation



Examples:

1. sum of n numbers using recursion
2. exponential of a number using recursion

Sum of n numbers	Output

def sum(n):
  if(n==1):
  return 1
  else:
  return n*sum(n-1)
 	
n=eval(input("enter no. to find sum:"))
sum=sum(n)
print("Fact is",sum)
	enter no. to find sum:10
	Fact is 55

Strings:

- Strings
- String slices
- Immutability
- String functions and methods
- String module

Strings:

- String is defined as sequence of characters represented in quotation marks (either single quotes (‘) or double quotes (“).
- An individual character in a string is accessed using a index.
- The index should always be an integer (positive or negative).
- A index starts from 0 to n-1.
- Strings are immutable i.e. the contents of the string cannot be changed after it is created.
- Python will get the input at run time by default as a string.
- Python does not support character data type. A string of size 1 can be treated as characters.

1. single quotes (' ')
2. double quotes (" ")
3. triple quotes(“”” “”””)

Operations on string:

1. Indexing
2. Slicing
3. Concatenation
4. Repetitions
5. Member ship

String A	H	E	L	L	O
Positive Index	0	1	2	3	4
Negative Index	-5	-4	-3	-2	-1

indexing	>>>a=”HELLO” >>>print(a[0]) >>>H >>>print(a[-1]) >>>O	❖ Positive indexing helps in accessing the string from the beginning ❖ Negative subscript helps in accessing the string from the end.

Slicing:	Print[0:4] – HELL Print[:3] – HEL Print[0:]- HELLO	The Slice[start : stop] operator extracts sub string from the strings. A segment of a string is called a slice.
Concatenation	a=”save” b=”earth” >>>print(a+b) saveearth	The + operator joins the text on both sides of the operator.
Repetitions:	a=”panimalar ” >>>print(3*a) panimalarpanimalar panimalar	The * operator repeats the string on the left hand side times the value on right hand side.

Membership:	>>> s="good morning"
	>>>"m" in s
	True
	>>> "a" not in s
	True
		Using membership operators to check a particular character is in string or not. Returns true if present

String slices:

- A part of a string is called string slices.
- **The process of extracting a sub string from a string is called slicing.**

Slicing: **a=”HELLO”**	Print[0:4] – HELL Print[:3] – HEL Print[0:]- HELLO	The Slice[n : m] operator extracts sub string from the strings. A segment of a string is called a slice.

Immutability:

- Python strings are “immutable” as they cannot be changed after they are created.
- Therefore [ ] operator cannot be used on the left side of an assignment.

operations	Example	output
element assignment	a="PYTHON" a[0]='x'	TypeError: 'str' object does not support element assignment
element deletion	a=”PYTHON” del a[0]	TypeError: 'str' object doesn't support element deletion
delete a string	a=”PYTHON” del a	NameError: name 'my_string' is not defined

	print(a)	

string built in functions and methods:

A **method** is a function that "belongs to" an object.

Syntax to access the method

Stringname.method()

a="happy birthday"
here, a is the string name.

	syntax	example	description
1	a.capitalize()	>>> a.capitalize() ' Happy birthday'	capitalize only the first letter in a string
2	a.upper()	>>> a.upper() 'HAPPY BIRTHDAY'	change string to upper case
3	a.lower()	>>> a.lower() ' happy birthday'	change string to lower case
4	a.title()	>>> a.title() ' Happy Birthday '	change string to title case i.e. first characters of all the words are capitalized.
5	a.swapcase()	>>> a.swapcase() 'HAPPY BIRTHDAY'	change lowercase characters to uppercase and vice versa
6	a.split()	>>> a.split() ['happy', 'birthday']	returns a list of words separated by space
7	a.center(width,"fillchar")	>>>a.center(19,"*") '***happy birthday***'	pads the string with the specified "fillchar" till the length is equal to "width"
8	a.count(substring)	>>> a.count('happy') 1	returns the number of occurences of substring
9	a.replace(old,new)	>>>a.replace('happy', 'wishyou happy') 'wishyou happy birthday'	replace all old substrings with new substrings

10	a.join(b)	>>> b="happy"
		>>> a="-"
		>>> a.join(b)
		'h-a-p-p-y'
			returns a string concatenated with the elements of an iterable. (Here "a" is the iterable)

11	a.isupper()	>>> a.isupper() False	checks whether all the case-based characters (letters) of the string are uppercase.
12	a.islower()	>>> a.islower() True	checks whether all the case-based characters (letters) of the string are lowercase.
13	a.isalpha()	>>> a.isalpha() False	checks whether the string consists of alphabetic characters only.

14	a.isalnum()	>>> a.isalnum() False	checks whether the string consists of alphanumeric characters.
15	a.isdigit()	>>> a.isdigit() False	checks whether the string consists of digits only.
16	a.isspace()	>>> a.isspace() False	checks whether the string consists of whitespace only.
17	a.istitle()	>>> a.istitle() False	checks whether string is title cased.
18	a.startswith(substring)	>>> a.startswith("h") True	checks whether string starts with substring
19	a.endswith(substring)	>>> a.endswith("y") True	checks whether the string ends with the substring
20	a.find(substring)	>>> a.find("happy") 0	returns index of substring, if it is found. Otherwise -1 is returned.
21	len(a)	>>>len(a) >>>14	Return the length of the string
22	min(a)	>>>min(a) >>>' '	Return the minimum character in the string
23	max(a)	max(a) >>>'y'	Return the maximum character in the string

String modules:

- A module is a file containing Python definitions, functions, statements.
- Standard library of Python is extended as modules.
- To use these modules in a program, programmer needs to import the module.
- Once we import a module, we can reference or use to any of its functions or variables in our code.
- There is large number of standard modules also available in python.
- Standard modules can be imported the same way as we import our user-defined modules.

Syntax:

import module_name

Example	**output**
import string print(string.punctuation) print(string.digits) print(string.printable) print(string.capwords("happ y birthday")) print(string.hexdigits) print(string.octdigits)	 !"#$%&'()*+,-./:;<=>?@[\]^_`{\|}~ 0123456789 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJ KLMNOPQRSTUVWXYZ!"#$%&'()*+,- ./:;<=>?@[\]^_`{|}~ Happy Birthday 0123456789abcdefABCDEF 01234567

Escape sequences in string

Escape Sequence	Description	example
\n	new line	>>> print("hai \nhello") hai hello
\\	prints Backslash (\)	>>> print("hai\\hello") hai\hello
\'	prints Single quote (')	>>> print("'") '
\"	prints Double quote (")	>>>print("\"") "
\t	prints tab sapace	>>>print("hai\thello") hai hello
\a	ASCII Bell (BEL)	>>>print("\a")

List as array:

Array:

Array is a collection of similar elements. Elements in the array can be accessed by index. Index starts with 0. Array can be handled in python by module named array.

To create array have to import array module in the program.

Syntax :

import array

Syntax to create array:

Array_name = module_name.function_name('datatype',[elements])

example:

a=array.array('i',[1,2,3,4])

a- array name

array- module name

i- integer datatype

Example

Program to find sum of array elements	**Output**
import array sum=0 a=array.array('i',[1,2,3,4]) for i in a: sum=sum+i print(sum)	10

Convert list into array:

fromlist() function is used to append list to array. Here the list is act like a array.

Syntax:

arrayname.fromlist(list_name)

Example

program to convert list into array	**Output**
import array sum=0 l=[6,7,8,9,5] a=array.array('i',[]) a.fromlist(l) for i in a: sum=sum+i print(sum)	35

Methods in array a=[2,3,4,5]

	Syntax	example	Description
1	array(data type, value list)	array('i',[2,3,4,5])	This function is used to create an array with data type and value list specified in its arguments.
2	append()	>>>a.append(6) [2,3,4,5,6]	This method is used to add the at the end of the array.
3	insert(index,element)	>>>a.insert(2,10) [2,3,10,5,6]	This method is used to add the value at the position specified in its argument.
4	pop(index)	>>>a.pop(1) [2,10,5,6]	This function removes the element at the position mentioned in its argument, and returns it.
5	index(element)	>>>a.index(2) 0	This function returns the index of value
6	reverse()	>>>a.reverse() [6,5,10,2]	This function reverses the array.
7	count()	a.count() 4	This is used to count number of elements in an array

ILLUSTRATIVE PROGRAMS:

Square root using newtons method:	**Output:**

def newtonsqrt(n):
  root=n/2
  for i in range(10):
  root=(root+n/root)/2
  print(root)
n=eval(input("enter number to find Sqrt: "))
newtonsqrt(n)
	enter number to find Sqrt: 9
	3.0

GCD of two numbers	**output**

n1=int(input("Enter a number1:"))
n2=int(input("Enter a number2:"))
for i in range(1,n1+1):
  if(n1%i==0 and n2%i==0):
  gcd=i
print(gcd)
	Enter a number1:8
	Enter a number2:24
	8

Exponent of number	**Output:**

def power(base,exp):
  if(exp==1):
  return(base)
  else:
  return(base*power(base,exp-1))
base=int(input("Enter base: "))
exp=int(input("Enter exponential value:"))
result=power(base,exp)
print("Result:",result)
	Enter base: 2
	Enter exponential value:3
	Result: 8

sum of array elements:	**output:**
a=[2,3,4,5,6,7,8] sum=0 for i in a: sum=sum+i print("the sum is",sum)	the sum is 35
Linear search	**output**

a=[20,30,40,50,60,70,89]
print(a)
search=eval(input("enter a element to search:"))
for i in range(0,len(a),1):
  if(search==a[i]):
  print("element found at",i+1)
  break
else:
  print("not found")
	[20, 30, 40, 50, 60, 70, 89]
	enter a element to search:30
	element found at 2

Binary search	**output**

a=[20, 30, 40, 50, 60, 70, 89]
print(a)
search=eval(input("enter a element to search:"))
start=0
stop=len(a)-1
while(start<=stop):
  mid=(start+stop)//2
  if(search==a[mid]):
  print("elemrnt found at",mid+1)
  break
  elif(search<a[mid]):
  stop=mid-1
  else:
  start=mid+1
else:
  print("not found")
	[20, 30, 40, 50, 60, 70, 89]
	enter a element to search:30
	element found at 2

Function:

Lambda function (Anonymous Functions)

A function is said to be anonymous function when it is defined without a name and def keyword.

In python, normal function are defined using **def** keyword and Anonymous function are defined using **lambda** keyword.

Syntax: lambda arguments: expression

Lambda function can have any number of argument but only one expression.The expression are evaluated and returned.

Example:

```
>>> a=lambda b: b*2+b
>>> print(a(3))
```

9

Or

```
def a(b):
    return b*2+b
```

Part A:

1. What are Boolean values?
2. Define operator and operand?
3. Write the syntax for if with example?
4. Write the syntax and flowchart for if else.
5. Write the syntax and flowchart for chained if.
6. define state
7. Write the syntax for while loop with flowchart.
8. Write the syntax for for loopwith flowchart.
9. Differentiate break and continue.
10. mention the use of pass
11. what is fruitful function
12. what is void function
13. mention the different ways of writing return statement
14. What is parameter and list down its type?
15. What is local and global scope?
16. Differentiate local and global variable?
17. What is function composition, give an example?
18. Define recursion.
19. Differentiate iteration and recursion.
20. Define string. How to get a string at run time.

21. What is slicing? Give an example.

22. What is immutability of string?
23. List out some string built in function with example?
24. Define string module?
25. How can list act as array?
26. write a program to check the number is odd or even.
27. write a program to check the number positive or negative
28. write a program to check the year is leap year or not
29. write a program to find greatest of two numbers
30. write a program for checking eligibility for vote
31. write a program to find sum of n numbers
32. write a program to find factorial of given numbers
33. write a program to find sum of digits of a number
34. Write a program to reverse the given number.
35. Write a program to check the given number is palindrome or not.
36. write a program to check the given number is Armstrong or not
37. how can you use for loop in sequence.
38. how can you use else statement if loops.
39. What is the use of map() function?

Part B:

1. Explain conditional statements in detail with example(if, if..else, if..elif..else)
2. explain in detail about operators in detail
3. Explain in detail about iterations with example.(for, while)
4. Explain the usage of else statements in loops
5. Explain in detail about using for loop in sequence.
6. Explain in detail about string built in function with suitable examples?
7. Explain about loop control statement(break, continue, pass)
8. Breifly discuss about fruitful function.
9. Discuss with an example about local and global variable
10. Discuss with an example about function composition
11. Explain in detail about recursion with example.
12. Explain in detail about strings and its operations(slicing,immutablity)
13. Program to find square root of a given number using newtons method
14. program to find gcd of given nnumber
15. program to find exponentiation of given number using recursion

16. program to find sum of array elements.
17. program to search an element using linear search.
18. program to search an element using binary element.
19. program to find factorial of a given number using recursion