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.

1) Conditional Statements

- Conditional if
- Alternative if… else
- Chained if…elif…else
- Nested if….else

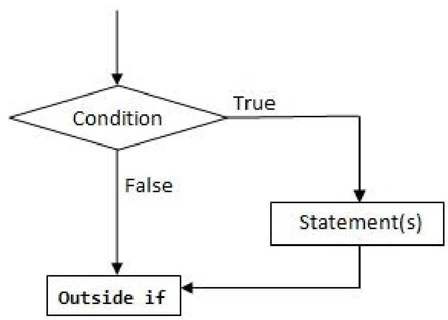
Conditional (if):

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:



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

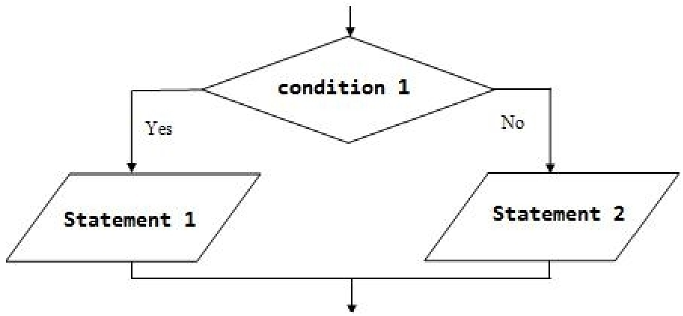
Alternative (if-else):

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

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 year"))
if(y%4==0):
    print("leap year")
else:
    print("not leap year")
	enter a year2000
	leap year

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.

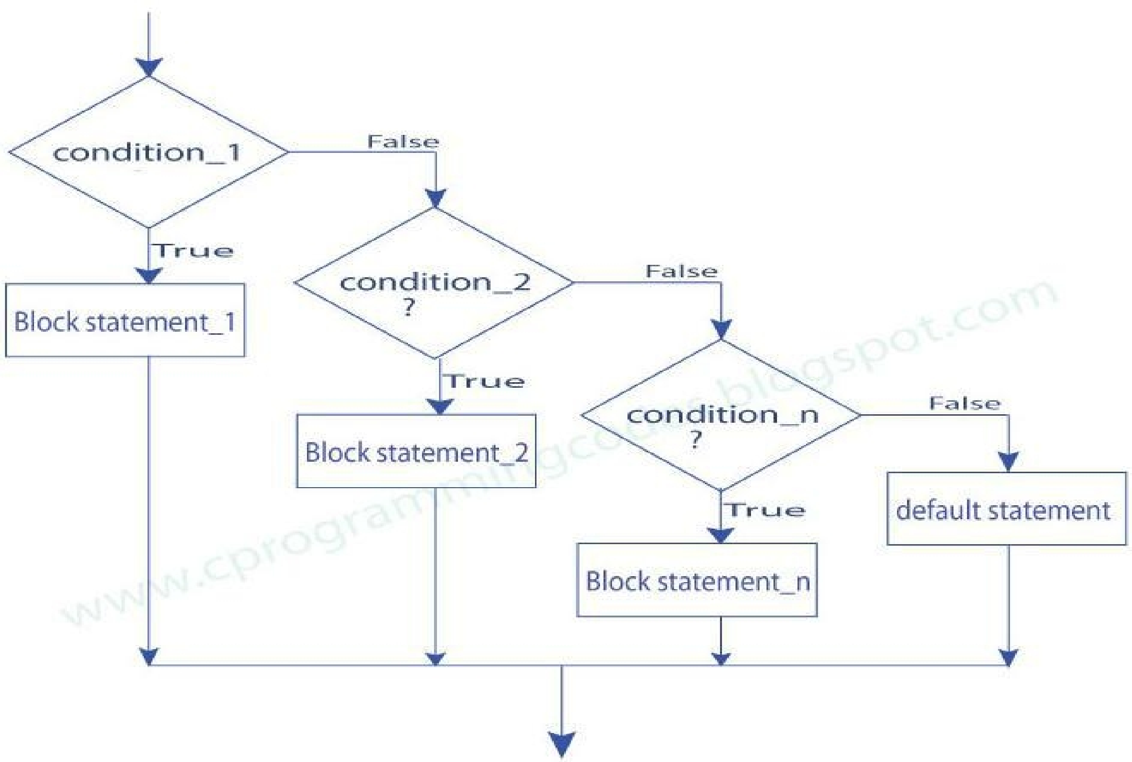
The if block can have only one else block. But it can have multiple elif blocks.

- 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

student mark system	**Output**
mark=eval(input("enter ur mark:"))	enter ur mark:78
if(mark>=90):	grade:B
print("grade:S")	
elif(mark>=80):	
print("grade:A")	
elif(mark>=70):	
print("grade:B")	
elif(mark>=50):	
print("grade:C")	
else:	
print("fail")	
traffic light system	**Output**
colour=input("enter colour of light:")	enter colour of light:green
if(colour=="green"):	GO
print("GO")	
elif(colour=="yellow"):	
print("GET READY")	
else:	
print("STOP")	

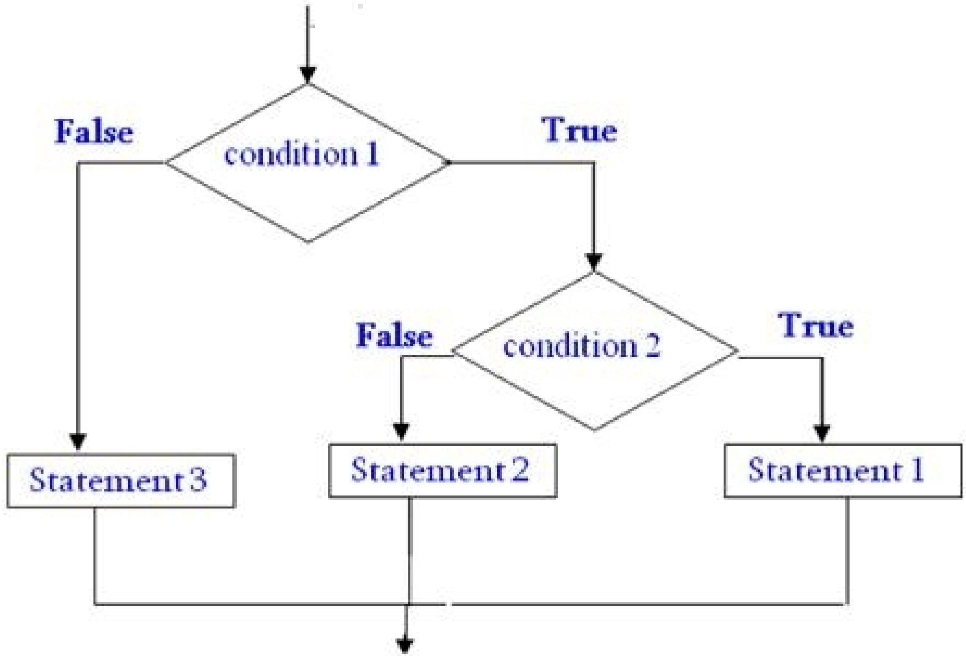
Nested conditionals

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:
        print(“the greatest no is”,c)
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")
	enter the value of n:-9
	the number is negative

else: print("the number is negative")	

2.Iteration Or Control Statements.

- 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.

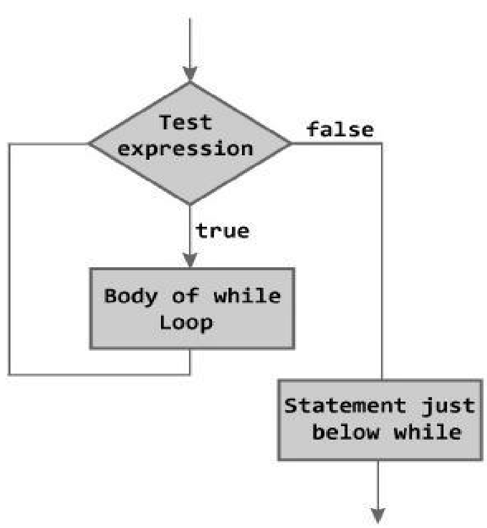
In Python, the body of the while loop is determined through indentation.

The statements inside the while start with indentation and the first unintended line marks the end.

Syntax:

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

Flow chart:



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

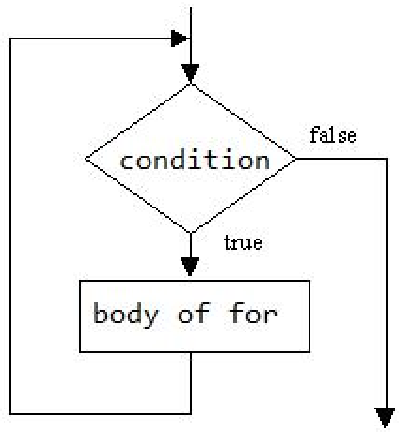
For loop:

- ***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. Program to print Fibonacci series.
2. check the no is prime or not

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

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.

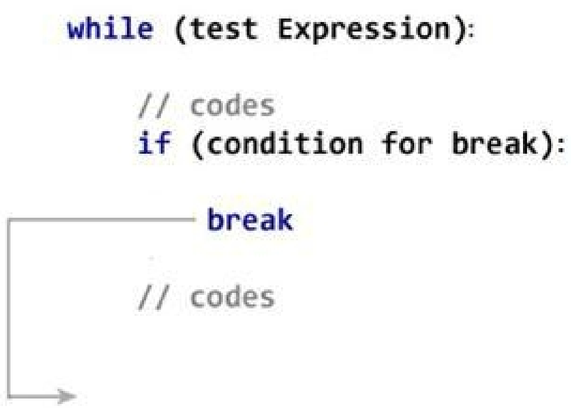
3.Loop Control Structures

BREAK

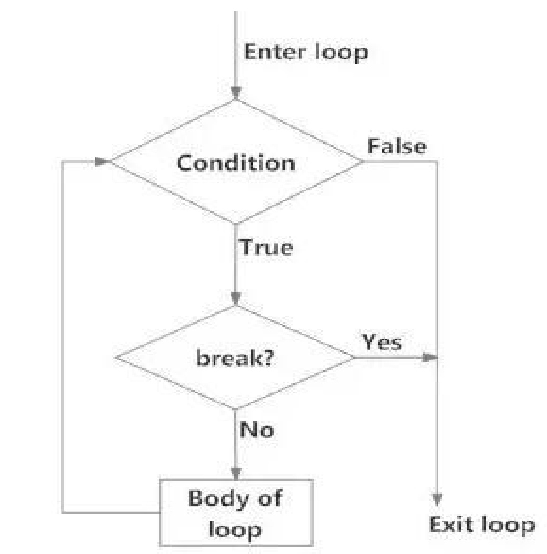
- Break statements can alter the flow of a loop.
- It terminates the current
- loop and executes the remaining statement outside the loop.
- If the loop has else statement, that will also gets terminated and come out of the loop completely.

Syntax:

break



Flowchart



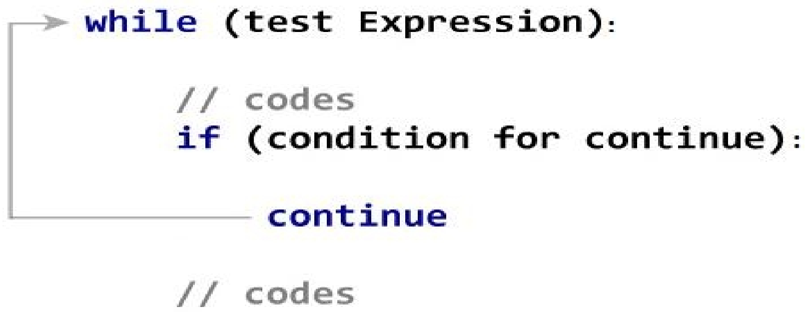
example	Output

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

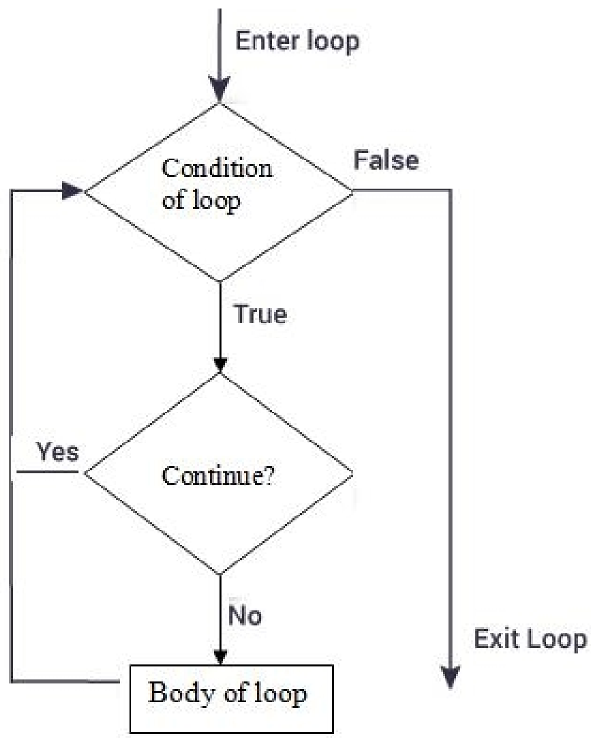
CONTINUE

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

Syntax: Continue



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

4) Fruitful Function

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

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– return expression

return 8– return value

PARAMETERS / ARGUMENTS(refer 2nd unit)

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(): **Global Variable**
  b=20
  c=a+b
  print© **Local Variable**
 	
def sub():
  b=30
  c=a-b
  print©
print(a)
	
	70
	
	20
	
	50

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

Example	output

def add():
    b=20
    c=a+b   **Local Variable**
    print©
def sub():
    b=30
    c=a-b   **Local Variable**
    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.

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)
print("the avg is",avg)
	enter a:4
	enter b:8
	the avg is 6.0

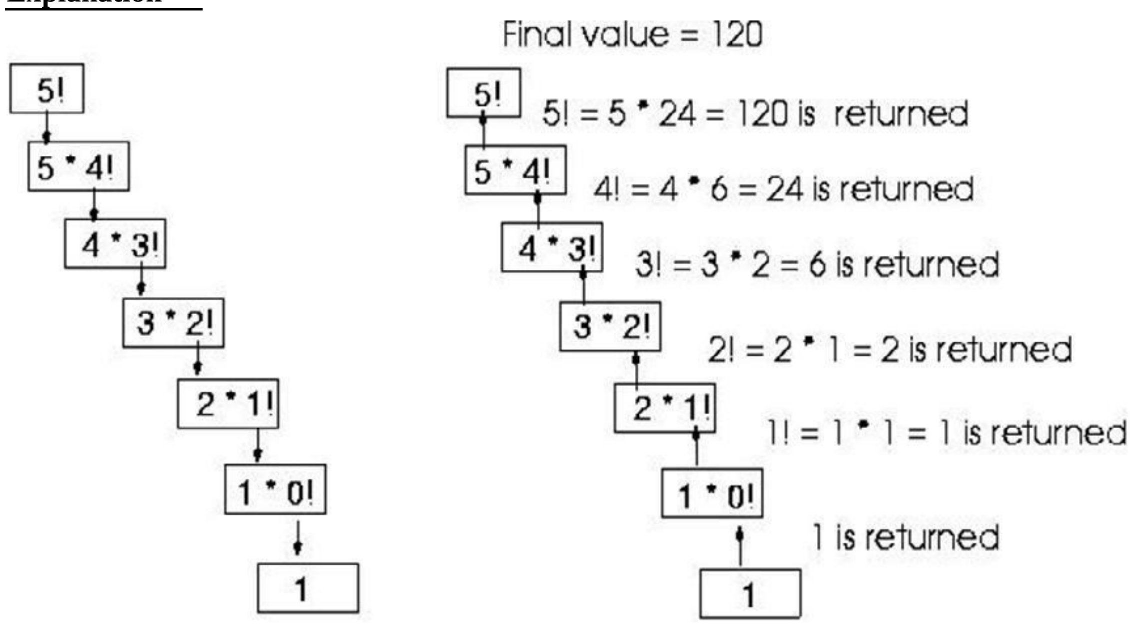
Recursion

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

5)Explain about Strings and its operation:

- 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) Save earth	The + operator joins the text on both sides of the operator.
Repetitions:	a=”panimalar ” >>>print(3*a)	The * operator repeats the string on the left hand side times the value on right

	panimalarpanimalar panimalar	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 print(a)	NameError: name 'my_string' is not defined

<u>string built in functions and methods:</u>

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

<u>Syntax to access the method</u>

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.

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

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

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”)

6) 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 of an 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()	This is used to count number of

7.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

```
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("element found at",mid+1)
        break
    elif(search<a[mid]):
        stop=mid-1
    else:
        start=mid+1
else:
    print("not found")
```

output

```
[20, 30, 40, 50, 60, 70, 89]
enter a element to search:30
element found at 2
```

Two marks:

1. **What is a Boolean value?**
   - 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
```

2. **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

3. **Write a Python program to accept two numbers, multiply them and print the result.**

```
number1 = int(input("Enter first number: "))
number2 = int(input("Enter second number: "))
mul = number1 * number2
print("Multiplication of given two numbers is: ", mul)
```

4. **Write a Python program to accept two numbers, find the greatest and print the result.**

```
number1 = int(input("Enter first number: "))
number2 = int(input("Enter second number: "))
if(number1>number2):
        print('number1 is greater',number1)
else:
        print('number2 is greater',number2)
```

5. Define recursive function.

Recursion is a way of programming or coding a problem, in which a function calls itself one or more times in its body. Usually, it is returning the return value of this function call. If a function definition fulfils the condition of recursion, we call this function a recursive function.

Example:

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

6. Write a program to find sum of n numbers:

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

```
enter n
10
55
```

7. **What is the purpose of pass statement?**

Using a pass statement is an explicit way of telling the interpreter to do nothing.

- 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

8. **Compare string and string slices.**

A string is a sequence of character.

Eg: fruit = 'banana'

String Slices :

A segment of a string is called string slice, selecting a slice is similar to selecting a character.

Eg: >>> s ='Monty Python'

```
>>> print s[0:5]
Monty
>>> print s[6:12]
Python
```

9. **Explain global and local scope.**

The scope of a variable refers to the places that we can see or access a variable. If we define a variable on the top of the script or module, the variable is called global variable. The variables that are defined inside a class or function is called local variable.

Eg:

```
def my_local():
        a=10
        print("This is local variable")
```

Eg:

```
a=10
def my_global():
        print("This is global variable")
```

10. **Mention a few string functions.**

s.captilize() – Capitalizes first character of string

s.count(sub) – Count number of occurrences of string

s.lower() – converts a string to lower case

s.split() – returns a list of words in string