## Unit - III

- of reusable code which can be called whenever required
- · Divide a large portion program into the basic building blocks known as function
- · The function contains the set of programming statements enclosed by if
- The functions is also known as procedure in other programming languages.

  Collection of function creates a program
- · Need of function: can be called multiple times.
- · By using functions we can avoid rewriting same code again in a program.
- · We call call python functions any number of times and from any place in program.
- · We can track a large python program easily when it is divided into multiple functions
- · Defining a function:
  - · Function blocks begin with the Keyword def followed by the function name and parentheses (()).
- · Any input parameters or argument should be placed within these parameters parentheses.
- · The first statement of a function can be an optional statement.

  The code block within every functions starts with a colon
- (:) and is indented.

   A return statement with no argument is the same as return None.

Syntax:

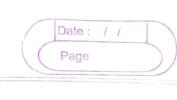
def function-name (parameter):

Instruction to be processed

Return statement

Date:	/	/	
Page			

F	
Example:	
( ) def hello world ():	
print ("hello world")	
	,
2 def func (name);	
print ("Hi" name);	
# calling the function func ("Ayush")	•
Tunc ( nyush )	•
3 def sum (a,b):	
return atb;	
a=int(input ("Enter a:"))	
b=int (input("Enter b:"))	
print ("Sum=",sum(a,b))	
4 def printme (str):	
- print str	
return;	
- printme ("first call to user definied function!")	
- printme ("second call to the same function")	
(5) 1.C C . C	
(5) def func (name, message):	n
print (" printing the message with", name func (name = "John", message = "hello")	be "and", message)
- OIP=	•
- printing the message with John and hello.	
promise massage with ostal and mello	
- * To create a function we use def keyword	1 followed by
- the tunchon_nome()	1
: def fynction-name():	



Enka the name & Mry an

Drequired Arguments: used at time of Function calling.

these are the arguments which are required to be passed at time of function calling with exact match of positions

rame = input("Enter name?")..... arga
print (func(name))

Types of Arguments

def func (name):

message="Hi"+name;

(2) Keyword Arguments:

call function with keywords. argument = keywords

with random order function call.

keywords are matched in function call and replaced in def func (name, message):

def func (name, message):

print ("printing message with "name," and ", message)

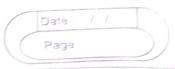
func (name = "John", message = "hello") --- keywords

printing message with John and hello

Default Arguments: if value of any argument is not provided at time of function call, then that argument can be initialized with the value given in the definition.

def printme (name, age = 22): ..... defaut Argument print ("Mynune is" name, "and age is", age) printme (name = "john") age is considered in function

O/P My name is john and age is 22.



(4) Variable length Arguments: We may not know the no of arguments to pais.
in such cases python provides comma separated value (hyples) at function call. def printme ("names): print ("type of passed argument is", type (names)) print ("printing the passed orguments.") for name in names: print (name) print me ("john", "David" "Smith", "nick") musble OIP type of passed arguments is < class 'tuples'> printing the passed arguments. john David Smith · Scope of variables: depend upon the location where the variable is being declared. alobal variables lad variables \* Scope of Variables: depends upon the location where the variable

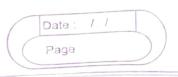
is being declared variable declared in one part of program may

Date: / /
Page

Odeclared inside a function of computer program	GLOBAL VARIABLE  Odeclared outside a function of computer program			
@ Accessible only within the functions program	© Accessible by all Functions in the functions program			
(3) Created -> function starts  executing  destroyed -> execution complete	3 Remain Forever entire time !			
(3) More realiable and secure Value cannot be changed	(4) Value can be changed			
. * Anonymous Function (lambo				
· called anonymous because they are not declared in *				
· Can use lambde Keyword to create small anonymous fun. · can take any no of arguments.				
· Returns just one value in the form of expression .  · Can't contain multiple expressions -				
· can't be a direct coll to point .				
· have their own local namespace Syntax: lambda arguments: expression				
ex				
x = lambda q: a+10 print ("sum =", x(20))				
DITIT ( SUID - , X ( 20))				

0/P => 30

10



\* The Anonymous function (lambda):-

· The anonymous function contains a small piece of code

· You can use the lambda keyword to create small anonymous

functions.

· Lambda forms can take any number of arguments.
· Return just one value in the form of an expression.

· They cannot contain commands or multiple expressions.

· Lambda requires an expression.

· lambda functions have their own local namespace · cannot access variables ·

Syntax:

x=lambda a:a+10 print ("sum="x(20))

O def myfunc (n):

return lambda a: a \*n

x = my func(2)

print (x(11))

· Python Module:

· Eallows you to logically organize your python code. · Grouping code into module makes code easier to

understand and use.

· module is a python object ·

· module is a file consisting of python rode · can define functions, classes, variables

· used to import functionality of one module into other

· Instead of importing the whole module into namespace python provides flexibility to import only the specific attributes of a module.

a using from-import statement



· Python packages:-· facilitate the developer with application development environment by providing a hierarchical dictiony packages contains sub-packages module and

Sub-modules · used to categorize the application level code efficiently

+ class account: def -- init -- (self, accounting balance):

self.account no = account no. self. balance = balance

def withdraw (self, amount) self balance = balance - amount

print ("Amount of", self balance) acci = account (1234,5000) acchaithdraw (500)

\* class yehicle: def\_\_int\_\_ (self, model, rolour)

Self. model = model

self-colour=colour

def show (self) print (self. (obur)

print (Self. model) vehicle ( A(234, "orenge") veh show ()

O/p => orenge

learning fundions of function def greet (): • - print ("Hello") Hew de youde ? - print ("How do you do?") Jefine Forther 1greet () · Function with Arguments OIP Hello Jack : def greet (name): print ("Hello", name) How do you do? →print ("How do you do?") greet ("Jack") Jer nome = Jack + def add-numbers (n1, n2): + -> result = n1 + n2 - print ("The sum is", result) number 1 = 5.4 number 2 = 6.7 add-numbers (number 1, number 2) OIP => The sum is 12.1 \* def add-numbers (n1, n2) result=nitne return & result number 1 = 5.4 number 2=6.7 result = add-numbers (number 1, number 2) print ("The sum is", result)

	Date: / / Page
def greet ():  Print ("Hello")	O/P=>
Print ("Hello")	Hella Tark
print ("How do you do?") greet (Jack")	teminales program
print ("How do you do?")	
greet (Jack")	