**24-03-2025**

**==============**

**Functions:**

**=======**

What is function?

-------------------

-> it is a named block

-> which we can use to represent the group of statements

-> in python, the function can be defined with "def" keyword.

Syntax:

 def function-name(data):

 function-body

 with some statements

 return value

-> Function can accept the data

-> Function proceed on the data what we have given (manipulation)

-> Function can return the value

the return value from the function can represent with "return" keyword.

-> return is one the transfer statement

can always use within the functions only.

break statement importance:

---------------------------------

# break: can only use in the loop (for or while)

# when we can get the immediate termination from the loop, we can use "break".

i = 1

while i < 11:

 if i == 7:

 break

 print(i,end = "\t")

 i += 1

continue statement importance:

------------------------------------

# continue: transfer statement

# to skip the particular iteration (current iteration) and continue with remaining we can use

# continue

i = 1

while i < 11:

 if i == 7:

 i = i + 1

 continue

 print(i,end = "\t")

 i = i + 1

-> To work with functions, we need two important points:

 1) function definition

 2) function calling/invoking

Note:

----

-> the function cannot perform anything without calling/invoking

-> we can create the function without data sending also by defining parentheses as empty.

Syntax for the function call:

-----------------------------

function-name(values/data)

function without data:

------------------------

# function definition

def Greetings():

 print("Hi")

 print("Good Morning")

 print("Welcome To Ashok IT.")

# function call

# always define outside to the function

Greetings()

function with data:

-------------------

def addition(a,b):

 print("first value = ",a)

 print("Second value = ",b)

 c = a + b

 print("Sum = ",c)

addition(100,200)

function with return statement:

-------------------------------

def multiplication(a,b):

 c = a \* b

 return c

 # print("Hi")

result = multiplication(10,20)

print(result)

print(multiplication(30,40))

Note:

----

We can call the function in any number of times.

Unreachable Statement:

--------------------------

When we can define a statement after the return statement, that statement cannot execute

because the function can be get terminated immediately because of the "return". That statement is called as "Unreachable statement".

Function Parameters:

-----------------------

-> The data which can be send to the function using the function call is called as "Function Parameters".

-> Also called as "Arguments".

-> Arguments are of two types:

 1) Formal Parameters

 2) Actual Parameters

-> when the data can define within the function call is called as "Actual parameters".

-> The data within the function is called as "Actual Arguments".

-> Due the function call, the actual argument values can substitute in format arguments.

Ex:

def greetings(name):# formal parameter

 print("Hi",name)

 print("Good morning")

greetings("Ravi") # actual parameter

-> arguments can define in four different ways:

 1) positional argument representation

 -------------------------------------

def typeOfTriangle(a,b,c):

 print("a = ",a)

 print("b = ",b)

 print("c = ",c)

 if a == b and a == c:

 print("It is Equilateral Triangle.")

 elif a == b or a == c or b == c:

 print("It is Isosceles Triangle.")

 else:

 print("It is Scalene Triangle.")

p = int(input("Enter the side value for the triangle:")) # 13

q = int(input("Enter the side value for the triangle:")) # 12

r = int(input("Enter the side value for the triangle:")) # 13

# typeOfTriangle(p,q,r)

typeOfTriangle(r,p,q) # r = 13 p = 13 q = 12

 2) keyword argument representation

 -----------------------------------

def typeOfTriangle(a,b,c):

 print("a = ",a)

 print("b = ",b)

 print("c = ",c)

 if a == b and a == c:

 print("It is Equilateral Triangle.")

 elif a == b or a == c or b == c:

 print("It is Isosceles Triangle.")

 else:

 print("It is Scalene Triangle.")

p = int(input("Enter the side value for the triangle:")) # 13

q = int(input("Enter the side value for the triangle:")) # 12

r = int(input("Enter the side value for the triangle:")) # 13

# keyword argument

typeOfTriangle(a = q,c = p,b = q)

 3) default argument representation

 -----------------------------------

def additionOfNumber(a = 0,b = 1,c = 0):

 s = a + b + c

 print("The sum = ",s)

additionOfNumber() # without any data

additionOfNumber(10)

additionOfNumber(100,200)

additionOfNumber(1000,2000,3000)

additionOfNumber(b = 200)

 4) variable length argument representation

 ------------------------------------------

# when we want to exceute the function with any number of parameters, we can

# use "variable length arguments".

def addition(\*data):

 s = 0

 for i in data:

 s = s + i

 print("Sum = ",s)

addition()

addition(10)

addition(10,20,30,40)

addition(100,300,500,600,200,400,400,500)

# variable length arguments understood by the PVM as "tuple"

Why functions?

==========

When we need to re-use the same code

when we need to develop any application like modules

Ex: calculator:

 addition

 subtraction

 multiplication

faster development