Functions:

import os

import time

print("Pls wait clearing the screen")

time.sleep(2)

os.system("cls")

print("Fidning list of files")

os.system("dir")

OUTPUT:

Fidning list of files

Volume in drive C is OSc/SEQ/PYTHON\_PR/Functions (main)

Volume Serial Number is 5A8C-A9C3

Directory of C:\SEQ\PYTHON\_PR\Functions

08/05/2025 03:18 PM <DIR> .

08/05/2025 03:05 PM <DIR> ..

08/05/2025 03:21 PM 151 1.py

08/05/2025 03:07 PM 13,440 Function1.docx

2 File(s) 13,591 bytes

2 Dir(s) 756,650,860,544 bytes free

* Same code will not work in Linux

import os

import time

import platform

if ( platform.system() == "Windows"):

    print("Pls wait clearing the screen")

    time.sleep(2)

    os.system("cls")

    print("Fidning list of files")

    os.system("dir")

else:

    print("Pls wait clearing the screen")

    time.sleep(2)

    os.system("clear")

    print("List of files")

    os.system("ls -lrt")

08/05/2025 03:24 PM <DIR> .

08/05/2025 03:05 PM <DIR> ..

08/05/2025 03:27 PM 375 1.pyctions (main)

08/05/2025 03:24 PM 16,113 Function1.docx

2 File(s) 16,488 bytes

2 Dir(s) 756,648,468,480 bytes free

2.

import os

import time

import platform

def my\_code(cmd1,cmd2):

    print("Pls wait clearing the screen")

    time.sleep(2)

    os.system(cmd1)

    print("Fidning list of files")

    os.system(cmd2)

if ( platform.system() == "Windows"):

    my\_code("cls","dir")

else:

    my\_code("clear","ls -l")

A function is a block of code for some specific operation

Function code is re-usable

A function executes only when it is called

Class 2:

How to define a function and use defined function

* A function is a block of code for some specific operation.
* Function code is reusable.
* It will execute only when it is called.

def display():

print("First line")

print("Second line")

display()

display()

print("Now we called display function two lines")

display()

2.

display()

def display():

    print("Welcome to function concept")

    print("Simple way to define your function")

    return None

NameError: name 'display' is not defined

display() –Correct

3. # Rule to define function name

#a-z, A-Z,0-9,\_

#No 0-9 and \_

# no space

# no special char like @,#,$,%,^,&,\*,(,),!,~,`,/,\

4.

Why is the function used:

1. Code Reusability
2. Improve modularity

Types of Functions:

Basically, we can divide functions into the following two types

1. Built in Functions
2. User defined functions

print(len("Hi")) #Built in function

x=(5,6)

print(min(x))

print(max(x))

Class 3 :

Converting simple code into functions:

def welcome\_msg():

    print("Welcme to python scripting")

    print("Python is very easy to learn")

    return None

def known\_concepts():

    print("Now we are good with basics")

    print("We are about to start function concepts in  python")

    return None

def new\_concepts():

    print("Function are very easy in python")

    print("Now we are writing simple functions")

    return None

welcome\_msg()

known\_concepts()

new\_concepts()

Classl 4:

Scope of the variables

def myfunction1():

    print("Welcome to functions")

    myfunction2()

    return None

def myfunction2():

    print("Thanks you!!")

    return None

myfunction1()

Output:

############ output ############

#Welcome to functions

#Thanks you!!

2.

def myfunction1():

    print("x from function1",x)

    return None

def myfunction2():

    print("x from function2",x)

    return None

x=10

myfunction1()

myfunction2()

######## output ##########

# x from function1 10

# x from function2 10

3.

def myfunction1():

    x=60

    print("x from function1",x)

    return None

def myfunction2():

    print("x from function2",x)

    return None

#x=10

myfunction1()

myfunction2()

# NameError: name 'x' is not defined

4.

def myfunction1():

    x=60 # Local variable

    print("x from function1",x)

    return None

def myfunction2():

    print("x from function2",x)

    return None

x=10 # Global variabel

myfunction1()

myfunction2()

Output:

x from function1 60

x from function2 10

5.

def myfunction1():

    x=60 # Local variable

    print("x from function1",x)

    myfunction2()

    return None

def myfunction2():

    print("x from function2",x)

    return None

def  main():

    x=10

    myfunction1()

    return None

main()

Output:

###

#   print("x from function2",x)

#   NameError: name 'x' is not defined

6.

def myfunction1():

    x=60 # Local variable

    print("x from function1",x)

    myfunction2()

    return None

def myfunction2():

    print("x from function2",x)

    return None

def  main():

    global x

    x=10

    myfunction1()

    return None

main()

Output:

x from function1 60

x from function2 10

7.

def myfunction1():

    x=60

    print("x from function1",x)

    return None

def myfunction2(y):

    print("y from function2",y)

    return None

def  main():

    #global x

    x=10

    myfunction1()

    myfunction2(x)

    return None

main()

Output:

x from function1 60

y from function2 10

Simple function with ARGS:

def get\_result():

    result=num\*10

    print(f'Result: {result}')

    return None

num=eval(input("Enter a number: "))

get\_result()

Output:

Enter a number: 5

Result: 50

2.

def get\_result():

    result=num\*10

    print(f'Result: {result}')

    return None

def main():

    num=eval(input("Enter a number: "))

    get\_result()

    return None

main()

Output:

result=num\*10

NameError: name 'num' is not defined

3.

def get\_result(num):

    result=num\*10

    print(f'Result: {result}')

    return None

def main():

    num=eval(input("Enter a number: "))

    get\_result(num)

    return None

main()

Output:

Enter a number: 5

Result: 50

4.

Not same parameter can be used , any parameter will use

def get\_result(value):

    result=value\*10

    print(f'Result: {result}')

    return None

def main():

    num=eval(input("Enter a number: "))

    get\_result(num)

    return None

main()

Output:

Enter a number: 5

Result: 50

5.

def get\_result(value):

    result=value\*10

    print(f'Result: {result}')

    return None

def main():

    num=eval(input("Enter a number: "))

    get\_result(num)

    return None

main()

output:

Enter a number: 5

Result: 50

7.

def add(p,q):

    print(f'Addition: {p+q}')

    return None

def subtract(m,n):

    print(f'Subtraction: {m-n}')

    return None

def main():

    a=eval(input("Enter a number: "))

    b=eval(input("Enter a number: "))

    add(a,b)  # 11

    subtract(a,b) # -1

    subtract(b,a) # 1 --> Positional argument

    subtract(10,5) # 5 --> Positional argument

    return None

main()     #5,6

Output:

$ python 5.py

Enter a number: 5

Enter a number: 6

Addition: 11

Subtraction: -1

Subtraction: 1

Subtraction: 5

Functions with arguments and return values

def add(p,q):

    result=p+q

    print(f'Result: {result}')

    return None

def main():

    a=eval(input("Enter a number: "))

    b=eval(input("Enter a number: "))

    add(a,b)

    return None

main()

Output:

Enter a number: 5

Enter a number: 6

Result: 11

def add(p,q):

    result=p+q

    return result

def main():

    a=eval(input("Enter a number: "))

    b=eval(input("Enter a number: "))

    result=add(a,b)

    print(f'Result: {result}')

    return None

main()

Output:

Enter a number: 5

Enter a number: 6

Result: 11

3.

def multiply(p):

    # result=p\*10

    # return result

    return p \* 10

def main():

    num=eval(input("Enter a number: "))

    result=multiply(num)

    print(f'Result: {result}')

main()

$ python 6.py

Enter a number: 5

Result: 50

FUNCTION with Default arguments:

.