Function with default arguments:

def display(a):

    print(f'a={a}')

    return None

display(4) # 4

2.

def display(a):

    print(f'a={a}')

    return None

display()

# TypeError: display() missing 1 required positional argument: 'a'

3.

def display(a=1):

    print(f'a={a}') # a=1

    return None

display() #1

display(4) # a=4

display(5.5) # a=5.5

4.

def add\_numbers(p, q):

    result = p + q

    print(f'Result: {result}')

    return None

add\_numbers(5,6)  # Result: 11

#add\_numbers()  #TypeError: add\_numbers() missing 2 required positional arguments: 'p' and 'q'

5.

def add\_numbers(p, q=10): # Default value for q=10

    result = p + q

    print(f'Result: {result}')

    return None

add\_numbers(5,6)  # Result: 11

add\_numbers(10)  # Result: 20

6.

def add\_numbers(p=20, q=10): # Default value for q=10

    result = p + q

    print(f'Result: {result}')

    return None

add\_numbers()  # Result: 30

7.

def add\_numbers(p, q=10): # Default value for q=10

    result = p + q

    print(f'Result: {result}')

    return None

add\_numbers(5)  # Result: 15

8.

def add\_numbers(p=5, q):

    result = p + q

    print(f'Result: {result}') # SyntaxError: non-default argument follows default argument

    return None

add\_numbers(5)

9.

def add\_numbers(p, q=22):   # default values are alwyas right

    result = p + q

    print(f'Result: {result}')

    return None

add\_numbers(5)  # 27

10.

def add\_numbers(p, q=22):

    result = p + q

    print(f'Result: {result}')

    return None

add\_numbers(5)  # 27

add\_numbers(10,20)  # 30

add\_numbers(10) #32

FUNCTIONS with KEYWORD-BASED Args:

def display(a,b):

    print(f'a={a}, b={b}')

    return None

display(3,4) #  a=3, b=4

display(a=3,b=4) # a=3, b=4

display(b=44,a=33) # a=3, b=4

FUNCTIONS with Variable length Arguments:

def display(a):

    print(type(a))

    return None

display(4) #class 'int'

#display(4,5) #TypeError: display() takes 1 positional argument but 2 were given

# No of variables are more

2.

def display(\*arg):

    #print(type(arg)) # <class 'tuple'>

    print(arg)

    return None

display()

display(4)

display(4,5)

display(4,5,6)

#()

# (4,)

#(4, 5)

# (4, 5, 6)

3.

def display(\*arg):

    for each in arg:

        print(each,end=' ')

    return None

display(4,5,6)

print("\n-----------")

display("Hi",4,64)

4 5 6

-----------

Hi 4 64

4.

def display(\*arg):

    for each in arg:

        print(type(each),end=' ')

    return None

display(4,5,6)

print("\n-----------")

display("Hi",4.6,64)

# <class 'int'> <class 'int'> <class 'int'>

# -----------

# <class 'str'> <class 'float'> <class 'int'>

Function with variable keyword arguments:

def display(a,b):

    print(a)

    print(b)

    return None

display(5,6) # 5,6

display(a=5,b=4) # 5,4

display(b=44,a=33) # 33,44

#display(a=8,b=66,c=56) # TypeError: display() got an unexpected keyword argument 'c'

2.

def display(\*\*karg):

    print(type(karg)) # <class 'dict'>

display(a=1,b=2,c=3)

3.

def display(\*\*karg):

    print(karg)

display(a=1,b=2)

display(a=1,b=2,c=3)

# {'a': 1, 'b': 2}

# {'a': 1, 'b': 2, 'c': 3}

4.

def display(p,\*\*karg):

    print(p)

    print(karg)

display(56,q=10,r=20)

Answer:

56

{'q': 10, 'r': 20}

HOW TO USE FUNCTIONS of one script into another script?

Code 1:

Script1:

def add(a,b):

    print(f'The result add: {a+b}')

    return None

def sub(a,b):

    print(f'The result  sub: {a-b}')

    return None

script2:

import script1

print(dir(script1))

def mult(a,b):

    print(f'The result mul: {a\*b}')

    return None

def div(a,b):

    print(f'The result div: {a/b}')

    return None

mult(2,5)

Output:

['\_\_builtins\_\_', '\_\_cached\_\_', '\_\_doc\_\_', '\_\_file\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', 'add', 'sub']

The result mul: 10

Code 2:

mult(2,5)

script1.add(20,10)

script1.sub(20,10)

$ python script2.py

Output:

The result mul: 10

The result add: 30

The result sub: 10

Code 3:

Script1

add(2,3)

sub(2,3)

script2:

mult(2,5)

print("Called from script2")

script1.add(20,10)

script1.sub(20,10)

Output:

The result add: 5

The result sub: -1

The result mul: 10

Called from script2

The result add: 30

The result sub: 10

Code 4:

Script1:

print(\_\_name\_\_)

script2:

import script1

print(\_\_name\_\_)

Output:

script1

\_\_main\_\_

Code 5:

Script 1:

def add(a,b):

    print(f'The result add: {a+b}')

    return None

def sub(a,b):

    print(f'The result  sub: {a-b}')

    return None

def main():

    print("script 1")

    add(2,3)

    sub(2,3)

if \_\_name\_\_=="\_\_main\_\_":

    main()

script2:

import script1

def mult(a,b):

    print(f'The result mul: {a\*b}')

    return None

def div(a,b):

    print(f'The result div: {a/b}')

    return None

def main():

    print("script 2")

    mult(2,3)

    script1.add(4,5)

    script1.sub(4,6)

    return None

if \_\_name\_\_=="\_\_main\_\_":

    main()

Output:

$ python script2.py

script 2

The result mul: 6

The result add: 9

The result sub: -2

New Structure:

import sys

import os

import time

import datetime

def add(x,y):

    print(x+y)

    return None

def main():

    x=6

    y=7

    add(x,y)

    return None

if \_\_name\_\_=="\_\_main\_\_":

    main()

Output: 13

Simple Execution handling to changing the current working directory:

import os

def main():

    req\_path=input("Enter the dir name:::")

    try:

        os.chdir(req\_path)

        print("current workdir:::",os.getcwd())

    except FileNotFoundError:

        print("File not found error:::")

    except NotADirectoryError:

        print("Not a dir error:::::")

    except PermissionError:

        print("Permission error....")

    except Exception as e:

        print(e)

    return None

if \_\_name\_\_=="\_\_main\_\_":

    main()

Output:

$ python chdir.py

Enter the dir name:::/etc1

File not found error:::