Khoi Duong

Prof. Leinhos

CE450

11/3/2022

HW#4



Source code:

def mk\_wd(*balance*):

def f(*x*):

nonlocal *balance*

if *x* > *balance*:

return 'Insufficient funds'

*balance* -= *x*

return *balance*

return f

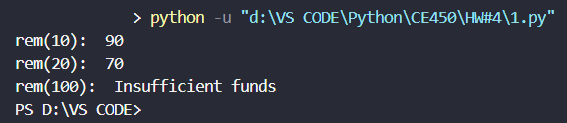
rem = mk\_wd(100)

print("rem(10): ", rem(10))

print("rem(20): ", rem(20))

print("rem(100): ", rem(100))

Run program & result:





Source code:

def rm\_all(*elem*, *lst*):

# remove all instances of element from a list

while *elem* in *lst*:

*lst*.remove(*elem*)

return *lst*

x = [3,1,2,1,5,1,1,7]

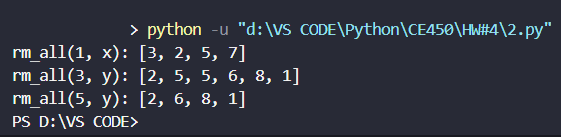
print("rm\_all(1, x):", rm\_all(1, x))

y = [3,2,3,5,5,6,8,1,3]

print("rm\_all(3, y):", rm\_all(3, y))

print("rm\_all(5, y):", rm\_all(5, y))

Run program & result:



Source code:

def add\_many(*x*, *elem*, *lst*):

a = *lst*.count(*x*)

for i in range(a):

*lst*.append(*elem*)

return *lst*

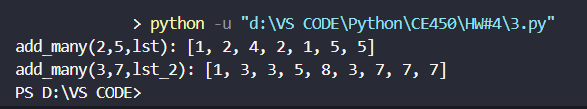
lst = [1,2,4,2,1]

print("add\_many(2,5,lst):", add\_many(2,5,lst))

lst\_2 = [1,3,3,5,8,3]

print("add\_many(3,7,lst\_2):", add\_many(3,7,lst\_2))

Run code and program:





Source code:

def f(*suits*, *numbers*):

if len(*suits*) == 0 or len(*numbers*) == 0:

return []

else:

ans = []

for i in range(len(*suits*)):

for j in range(len(*numbers*)):

ans.append((*suits*[i], *numbers*[j]))

return ans

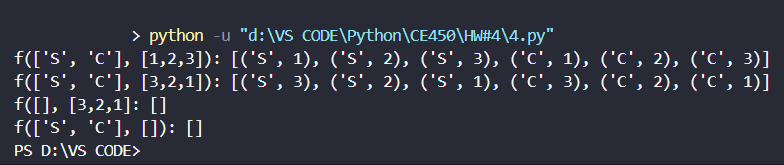
print("f(['S', 'C'], [1,2,3]):", f(['S', 'C'], [1,2,3]))

print("f(['S', 'C'], [3,2,1]):", f(['S', 'C'], [3,2,1]))

print("f([], [3,2,1]:", f([], [3,2,1]))

print("f(['S', 'C'], []):", f(['S', 'C'], []))

Run program & result:





Source code:

def mrg(*list1*, *list2*):

# merges 2 sorted list recursively

if len(*list1*) == 0:

return *list2*

elif len(*list2*) == 0:

return *list1*

elif *list1*[0] < *list2*[0]:

return [*list1*[0]] + mrg(*list1*[1:], *list2*)

else:

return [*list2*[0]] + mrg(*list1*, *list2*[1:])

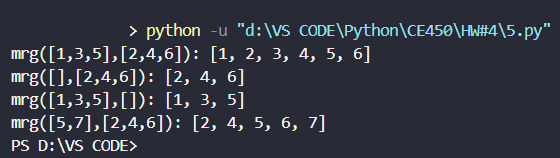
print("mrg([1,3,5],[2,4,6]):", mrg([1,3,5],[2,4,6]))

print("mrg([],[2,4,6]):", mrg([],[2,4,6]))

print("mrg([1,3,5],[]):", mrg([1,3,5],[]))

print("mrg([5,7],[2,4,6]):", mrg([5,7],[2,4,6]))

Run code & program:





Source code:

def fltn(*ls*):

# flatten deep list

if len(*ls*) == 0:

return []

elif type(*ls*[0]) == list:

return fltn(*ls*[0]) + fltn(*ls*[1:])

else:

return [*ls*[0]] + fltn(*ls*[1:])

print("fltn([1,2,[3,4,[5,6]]]):", fltn([1,2,[3,4,[5,6]]]))

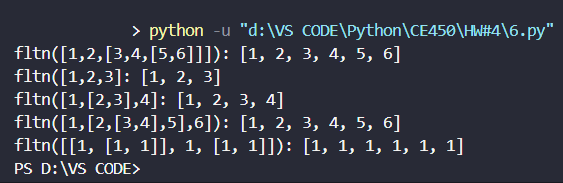
print("fltn([1,2,3]:", fltn([1,2,3]))

print("fltn([1,[2,3],4]:", fltn([1,[2,3],4]))

print("fltn([1,[2,[3,4],5],6]):", fltn([1,[2,[3,4],5],6]))

print("fltn([[1, [1, 1]], 1, [1, 1]]):", fltn([[1, [1, 1]], 1, [1, 1]]))

Run program & result:





Source code:

def chk\_elm(*lst*, *n*):

# check if element exists in the deep list or not

if len(*lst*) == 0:

return False

elif type(*lst*[0]) == list:

return chk\_elm(*lst*[0], *n*) or chk\_elm(*lst*[1:], *n*)

else:

return *lst*[0] == *n* or chk\_elm(*lst*[1:], *n*)

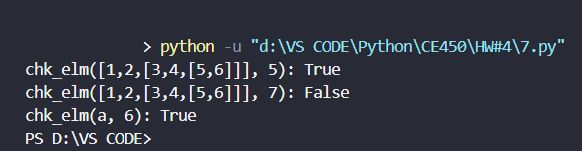
print("chk\_elm([1,2,[3,4,[5,6]]], 5):", chk\_elm([1,2,[3,4,[5,6]]], 5))

print("chk\_elm([1,2,[3,4,[5,6]]], 7):", chk\_elm([1,2,[3,4,[5,6]]], 7))

a = [[1,[2]],3,[[4],[5,[6]]]]

print("chk\_elm(a, 6):", chk\_elm(a, 6))

Run program & result:





Source code:

def sym(*l*):

# check whether the list is symmetric recursively

if len(*l*) == 0 or len(*l*) == 1:

return True

elif len(*l*) == 2:

return *l*[0] == *l*[1]

else:

return *l*[0] == *l*[-1] and sym(*l*[1:-1])

print("sym([1,2,3,2,1]):", sym([1,2,3,2,1]))

print("sym([1,2,3,3,2,1]):", sym([1,2,3,3,2,1]))

print("sym([]):", sym([]))

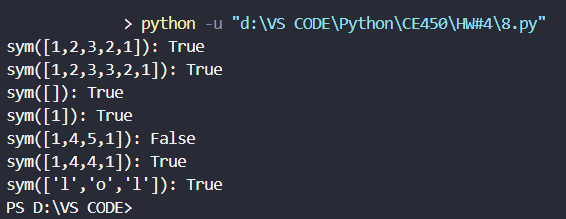
print("sym([1]):", sym([1]))

print("sym([1,4,5,1]):", sym([1,4,5,1]))

print("sym([1,4,4,1]):", sym([1,4,4,1]))

print("sym(['l','o','l']):", sym(['l','o','l']))

Run program & result:





Source code:

from operator import add, sub, mul

def fld(*lst*, *g*, *m*):

if len(*lst*) == 0:

return *m*

a = *g*(*m*, *lst*[0])

for i in range(1, len(*lst*)):

a = *g*(a, *lst*[i])

return a

s = [3,2,1]

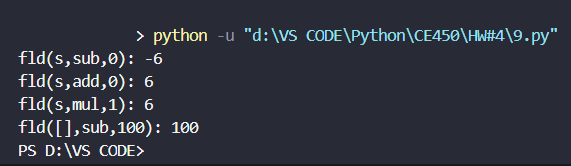
print("fld(s,sub,0):", fld(s,sub,0))

print("fld(s,add,0):", fld(s,add,0))

print("fld(s,mul,1):", fld(s,mul,1))

print("fld([],sub,100):", fld([],sub,100))

Run program & result:





Source code:

def crte\_2d\_arr(*rows*, *columns*):

# create a 2d array

arr = []

for i in range(*rows*):

arr.append([])

for j in range(*columns*):

arr[i].append('-')

return arr

print("crte\_2d\_arr(3,5):", crte\_2d\_arr(3,5))

print("crte\_2d\_arr(2,4):", crte\_2d\_arr(2,4))

print("crte\_2d\_arr(5,2):", crte\_2d\_arr(5,2))

Run program & result:

