Khoi Duong

Prof. Yang

CE450

10/11/2022

HW#3

GitHub link: <https://github.com/MynameisKoi/CE450/tree/main/HW%233>



Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%233/1.py>

def abndnt(*n*):

sum = 1

for i in range(1, *n*//2):

if *n* % i == 0:

print(i, "\*", *n*//i)

if i != 1:

sum += i + *n*//i

if sum > *n*:

return True

else:

return False

print("abndnt(12) = ", abndnt(12))

print("abndnt(14) = ", abndnt(14))

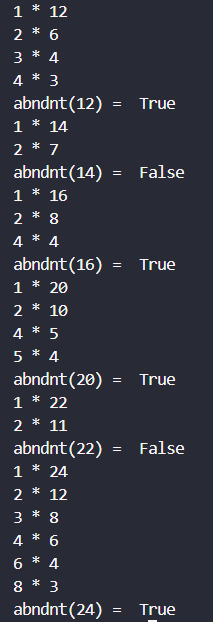
print("abndnt(16) = ", abndnt(16))

print("abndnt(20) = ", abndnt(20))

print("abndnt(22) = ", abndnt(22))

print("abndnt(24) = ", abndnt(24))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%233/2.py>

def adder(*f1*, *f2*):

return lambda *x*: *f1*(*x*) + *f2*(*x*)

def square(*n*): return *n*\*\*2

def identity(*n*): return *n*

a1 = adder(identity, square)

print("a1(4) = ", a1(4))

a2 = adder(a1, identity)

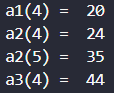
print("a2(4) = ", a2(4))

print("a2(5) = ", a2(5))

a3 = adder(a1, a2)

print("a3(4) = ", a3(4))

Run program & result:





Source code and explanation:

<https://github.com/MynameisKoi/CE450/blob/main/HW%233/3.py>

from operator import add, neg

# import addition and negative transformation functions from operator module

def combine\_funcs(*op*):

def combined(*f*, *g*):

# determine the combination functions from the argument (f and g)

def val(*x*):

# calculate the combined function at the given x

return *op*(*f*(*x*), *g*(*x*))

# return the result of the operation (an integer)

return val

# return the operation function to the combined functions (f and g)

return combined

add\_func = combine\_funcs(add)

# add\_func = combined(f, g) = add(f(x), g(x))

h = add\_func(abs, neg)

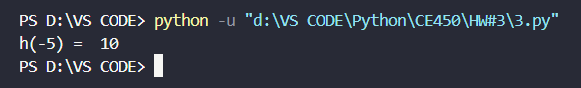
# h = combined(abs, neg) = add(abs(x), neg(x))

print(h(-5))

# h(-5) = add(abs(-5), neg(-5)) = add(5, 5) = 10

This is a high order function. The ‘add\_func’ creates an addition of functions f and g. Then we define h = add\_func(abs, neg) ⇔ this function will add the result of both functions abs (absolute value) and neg (negative value). Therefore, we have:

The printout result is 10.





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%233/4.py>

def intsct(*f*, *x*):

def g(*y*): return *y*(*x*)

return lambda *y*: g(*y*) == *f*(*x*)

def triple(*x*): return *x* \* 3

def square(*x*): return *x* \*\* 2

def identity(*x*): return *x*

def increment(*x*): return *x* + 1

at\_three = intsct(square, 3)

print("at\_three(triple): ", at\_three(triple))

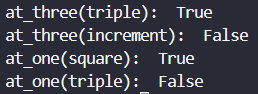
print("at\_three(increment): ", at\_three(increment))

at\_one = intsct(identity, 1)

print("at\_one(square): ", at\_one(square))

print("at\_one(triple): ", at\_one(triple))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%233/5.py>

def f():

def g():

def h(*x*):

def i():

return *x*

return i

return h

return g

print("f()()(3)():", f()()(3)())

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%233/6.py>

def smth(*g*, *dx*):

return lambda *x*: (*g*(*x*-*dx*) + *g*(*x*) + *g*(*x*+*dx*))/3

square = lambda *x*: *x*\*\*2

cube = lambda *x*: *x*\*\*3

triple = lambda *x*: 3\**x*

increment = lambda *x*: *x*+1

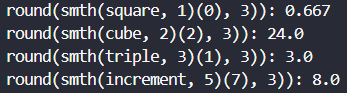
print("round(smth(square, 1)(0), 3)):", round(smth(square, 1)(0), 3))

print("round(smth(cube, 2)(2), 3)):", round(smth(cube, 2)(2), 3))

print("round(smth(triple, 3)(1), 3)):", round(smth(triple, 3)(1), 3))

print("round(smth(increment, 5)(7), 3)):", round(smth(increment, 5)(7), 3))

Run code & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%233/7.py>

def cyc(*g1*,*g2*,*g3*):

def y(*n*):

if *n* == 0:

return lambda *x*: *x*

elif *n* == 1:

return lambda *x*: *g1*(*x*)

elif *n* == 2:

return lambda *x*: *g2*(*g1*(*x*))

elif *n* == 3:

return lambda *x*: *g3*(*g2*(*g1*(*x*)))

else:

return lambda *x*: y(*n*-3)(*g3*(*g2*(*g1*(*x*))))

return y

def add\_one(*x*):

return *x*+1

def times\_two(*x*):

return *x*\*2

def add\_three(*x*):

return *x*+3

my\_cyc = cyc(add\_one, times\_two, add\_three)

h = my\_cyc(0)

print("h(5) = my\_cyc(0)(5) =", h(5))

h = my\_cyc(2)

print("h(1) = my\_cyc(2)(1) =", h(1))

h = my\_cyc(3)

print("h(2) = my\_cyc(3)(2) =", h(2))

h = my\_cyc(4)

print("h(2) = my\_cyc(4)(2) =", h(2))

h = my\_cyc(6)

print("h(1) = my\_cyc(6)(1) =", h(1))

h = my\_cyc(10)

print("h(2) = my\_cyc(10)(2) =", h(2))

Run code & result:

