Khoi Duong

Prof. Yang

CE450

9/21/2022

HW #1

GitHub link: <https://github.com/MynameisKoi/CE450/tree/main/CE450/HW%231>



Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/1.py>

def if\_function(*condition*, *true\_result*, *false\_result*):

if *condition*:

return *true\_result*

else:

return *false\_result*

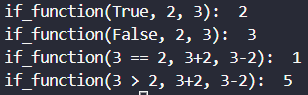
print("if\_function(True, 2, 3): " , if\_function(True, 2, 3))

print("if\_function(False, 2, 3): ", if\_function(False, 2, 3))

print("if\_function(3 == 2, 3+2, 3-2): ", if\_function(3 == 2, 3+2, 3-2))

print("if\_function(3 > 2, 3+2, 3-2): ", if\_function(3 > 2, 3+2, 3-2))

Run program & result:





Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/2.py>

def sum\_odd(*n*):

i = 0

ans = 0

while i <= *n*:

if i % 2 != 0:

ans += i

i += 1

return ans

print("sum\_odd(6): ", sum\_odd(6))

print("sum\_odd(7): ", sum\_odd(7))

Run program & result:



Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/3.py>

def foo(*a*,*b*,*c*,*d*):

num = [*a*,*b*,*c*,*d*]

x = min(num)

num.remove(x)

y = min(num)

return x\*\*2 + y\*\*2

print("foo(1,2,3,4): ", foo(1,2,3,4))

print("foo(-3,1,5,6): ", foo(-3,1,5,6))

Run program & result:



Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/4.py>

def df(*x*,*y*,*z*):

# takes three integers x, y, and z. It returns whether subtracting one of these numbers from another gives the third.

if *x*+*y* == *z*:

return True

elif *x*+*z* == *y*:

return True

elif *y*+*z* == *x*:

return True

else:

return False

print("df(5,3,2):", df(5,3,2))

print("df(2,3,5):", df(2,3,5))

print("df(2,5,3):", df(2,5,3))

print("df(-2,3,5):", df(-2,3,5))

print("df(-5,-3,-2):", df(-5,-3,-2))

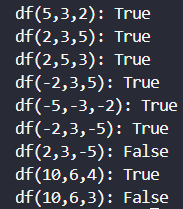
print("df(-2,3,-5):", df(-2,3,-5))

print("df(2,3,-5):", df(2,3,-5))

print("df(10,6,4):", df(10,6,4))

print("df(10,6,3):", df(10,6,3))

Run program & result:





Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/5.py>

def lrgst\_factor(*m*):

if *m* <= 1:

print("Error: m must be greater than 1")

return None

else:

for i in range(2,*m*):

if *m*%i == 0:

return int(*m*/i)

return 1

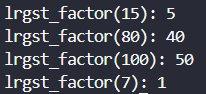
print("lrgst\_factor(15):", lrgst\_factor(15))

print("lrgst\_factor(80):", lrgst\_factor(80))

print("lrgst\_factor(100):", lrgst\_factor(100))

print("lrgst\_factor(7):", lrgst\_factor(7))

Run program & result:





Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/6.py>

def pfct\_num(*m*):

a = 1

while *m* > 0:

*m* -= a

a += 1

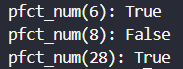
return *m* == 0

print("pfct\_num(6):", pfct\_num(6))

print("pfct\_num(8):", pfct\_num(8))

print("pfct\_num(28):", pfct\_num(28))

Run program & result:





Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/7.py>

def same\_ord(*a*, *b*):

#check if the number of digits from two positive input parameters is the same or not.

#return True if the number of digits is the same, otherwise return False.

if *a* < 0 or *b* < 0:

print("Error: a and b must be positive")

return None

else:

return len(str(*a*)) == len(str(*b*))

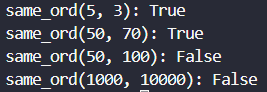
print("same\_ord(5, 3):", same\_ord(5, 3))

print("same\_ord(50, 70):", same\_ord(50, 70))

print("same\_ord(50, 100):", same\_ord(50, 100))

print("same\_ord(1000, 10000):", same\_ord(1000, 10000))

Run program & result:





Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/8.py>

def double\_5(*n*):

# return True if n has two fives in a row

if *n* < 0:

print("Error: n must be positive")

return None

else:

return "55" in str(*n*)

print("double\_5(5):", double\_5(5))

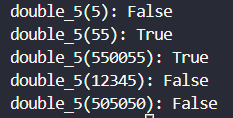
print("double\_5(55):", double\_5(55))

print("double\_5(550055):", double\_5(550055))

print("double\_5(12345):", double\_5(12345))

print("double\_5(505050):", double\_5(505050))

Run program & result:





Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/9.py>

def uniq\_digits(*x*):

# return the number of unique digits in positive integer x

if *x* < 0:

print("Error: x must be positive")

return None

else:

return len(set(str(*x*)))

print("uniq\_digits(8675309):", uniq\_digits(8675309))

print("uniq\_digits(1313131):", uniq\_digits(1313131))

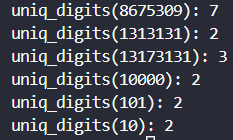
print("uniq\_digits(13173131):", uniq\_digits(13173131))

print("uniq\_digits(10000):", uniq\_digits(10000))

print("uniq\_digits(101):", uniq\_digits(101))

print("uniq\_digits(10):", uniq\_digits(10))

Run program & result:





Source code:

<https://github.com/MynameisKoi/CE450/blob/main/CE450/HW%231/10.py>

def amc(*n*):

# Return the smallest amicable number greater than positive integer n.

# If no amicable number exists, return None.

def sum\_factors(*a*):

# Return a list of all factors of positive integer a.

return sum([i for i in range(1,*a*) if *a*%i == 0])

if *n* < 0:

print("Error: n must be positive")

return None

else:

if *n* < 220:

# since the first amicable number is 220, we can skip all numbers < 220

return 220

else:

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

a = sum\_factors(i)

if sum\_factors(a) == i and a != i:

return i

return None

print("amc(5):", amc(5))

print("amc(220):", amc(220))

print("amc(284):", amc(284))

print("amc(5000):", amc(5000))

Run program & result:

