Khoi Duong

Prof. Yang

CE450

9/30/2022

HW#2

Github link: <https://github.com/MynameisKoi/CE450/tree/main/HW%232>



Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/1.py>

def fancy\_printing(*n*):

# Assume that the specified range is 0 to 30

for i in range(0, 31):

if i == 0: # Print 0

print(i)

continue

if *n* % i == 0:

print("Buzz!")

else:

print(i)

print("fancy\_printing(3): ")

fancy\_printing(3)

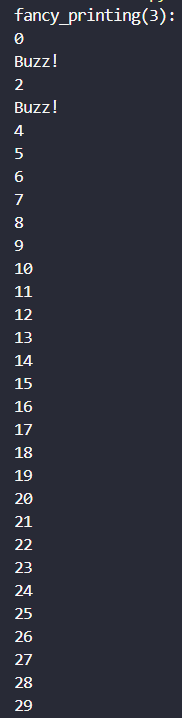
print("fancy\_printing(5): ")

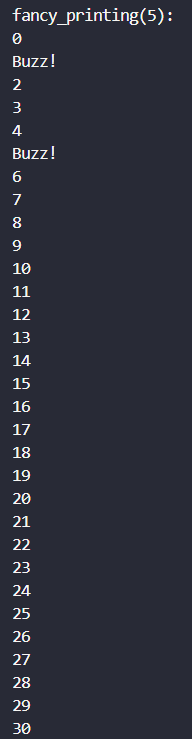
fancy\_printing(5)

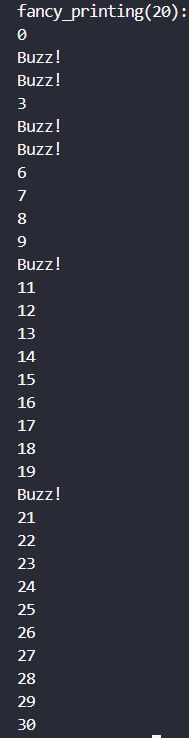
print("fancy\_printing(20): ")

fancy\_printing(20)

Run program & result:









Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/2.py>

def sum\_num(*n*):

if *n* == 0 or *n* == 1:

return *(n)*

return *n* + sum\_num(*n*-2)

print("sum\_num(4): ", sum\_num(4))

print("sum\_num(5): ", sum\_num(5))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/3.py>

def cnt\_primes(*n*):

if *n* == 0: return 0

# check if a number is prime

def is\_prime(*n*):

if *n* == 1: return False

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

if *n* % i == 0: return False

return True

# count the number of primes

if is\_prime(*n*): return 1 + cnt\_primes(*n*-1)

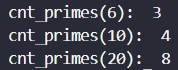
else: return cnt\_primes(*n*-1)

print("cnt\_primes(6): ", cnt\_primes(6))

print("cnt\_primes(10): ", cnt\_primes(10))

print("cnt\_primes(20): ", cnt\_primes(20))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/4.py>

def foo(*f*, *n*):

# return the function that computes the nth application of f

if *f* == "incr":

return lambda *x*: *x* + *n*

if *f* == "triple":

return lambda *x*: *x* \* (3 \*\* *n*)

if *f* == "square":

return lambda *x*: *x* \*\* (2 \*\* *n*)

# The three functions below are for single use application of f

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

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

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

print("incr(5): ", incr(5))

add3 = foo("incr", 3)

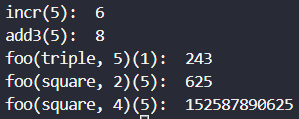
print("add3(5): ", add3(5))

print("foo(triple, 5)(1): ", foo("triple", 5)(1))

print("foo(square, 2)(5): ", foo("square", 2)(5))

print("foo(square, 4)(5): ", foo("square", 4)(5))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/5.py>

def op(*a*,*b*,*c*):

# computes a\*b + c using only recursion

if *a* == 0:

return *c*

if *a* > 0:

return op(*a*-1,*b*,*c*+*b*)

if *a* < 0:

return op(*a*+1,*b*,*c*-*b*)

print("op(2,3,4): ", op(2,3,4))

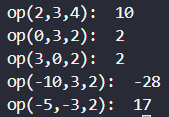
print("op(0,3,2): ", op(0,3,2))

print("op(3,0,2): ", op(3,0,2))

print("op(-10,3,2): ", op(-10,3,2))

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

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/6.py>

def checking(*x*):

# return True if the (base) digits of x > 0 are in non-decreasing order, and False otherwise

if *x* < 10: return True

if *x* % 10 < (*x* // 10) % 10: return False

return checking(*x* // 10)

print("checking(5): ", checking(5))

print("checking(11): ", checking(11))

print("checking(127): ", checking(127))

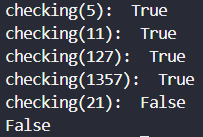
print("checking(1357): ", checking(1357))

print("checking(21): ", checking(21))

result = checking(1375)

print(result)

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/7.py>

def cal(*n*):

if *n* == 0:

return 0

if *n* == 1:

return 1

return *n* \* cal(*n*-2)

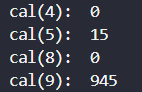
print("cal(4): ", cal(4))

print("cal(5): ", cal(5))

print("cal(8): ", cal(8))

print("cal(9): ", cal(9))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/8.py>

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

# Returns a function that returns whether f intersects function g at x

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

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

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

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

def g(*y*):

if *y* == "square":

return square(*x*)

if *y* == "triple":

return triple(*x*)

if *y* == "identity":

return identity(*x*)

if *y* == "increment":

return increment(*x*)

if *f* == "square":

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

if *f* == "triple":

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

if *f* == "identity":

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

if *f* == "increment":

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

at3 = intscts("square", 3)

print("at3(triple): ", at3("triple"))

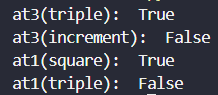
print("at3(increment): ", at3("increment"))

at1 = intscts("identity", 1)

print("at1(square): ", at1("square"))

print("at1(triple): ", at1("triple"))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/9.py>

def A(*n*):

if *n* <= 3: return *n*

else:

return A(*n*-1) + 2\*A(*n*-2) + 3\*A(*n*-3)

print("A(0): ", A(0))

print("A(1): ", A(1))

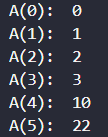
print("A(2): ", A(2))

print("A(3): ", A(3))

print("A(4): ", A(4))

print("A(5): ", A(5))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%232/10.py>

def bnc\_bck\_frth(*k*):

trig = 0

num = 1

def process():

nonlocal num

if trig == 0: num += 1

if trig == 1: num -= 1

def switch():

nonlocal trig

if trig == 0: trig = 1

else: trig = 0

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

process()

if i % 7 == 0 or "7" in list(str(i)):

switch()

return num

print("bnc\_bck\_frth(7) = ", bnc\_bck\_frth(7))

print("bnc\_bck\_frth(8) = ", bnc\_bck\_frth(8))

print("bnc\_bck\_frth(15) = ", bnc\_bck\_frth(15))

print("bnc\_bck\_frth(21) = ", bnc\_bck\_frth(21))

print("bnc\_bck\_frth(22) = ", bnc\_bck\_frth(22))

print("bnc\_bck\_frth(30) = ", bnc\_bck\_frth(30))

print("bnc\_bck\_frth(68) = ", bnc\_bck\_frth(68))

print("bnc\_bck\_frth(69) = ", bnc\_bck\_frth(69))

print("bnc\_bck\_frth(70) = ", bnc\_bck\_frth(70))

print("bnc\_bck\_frth(71) = ", bnc\_bck\_frth(71))

print("bnc\_bck\_frth(72) = ", bnc\_bck\_frth(72))

print("bnc\_bck\_frth(100) = ", bnc\_bck\_frth(100))

Run program & result:

