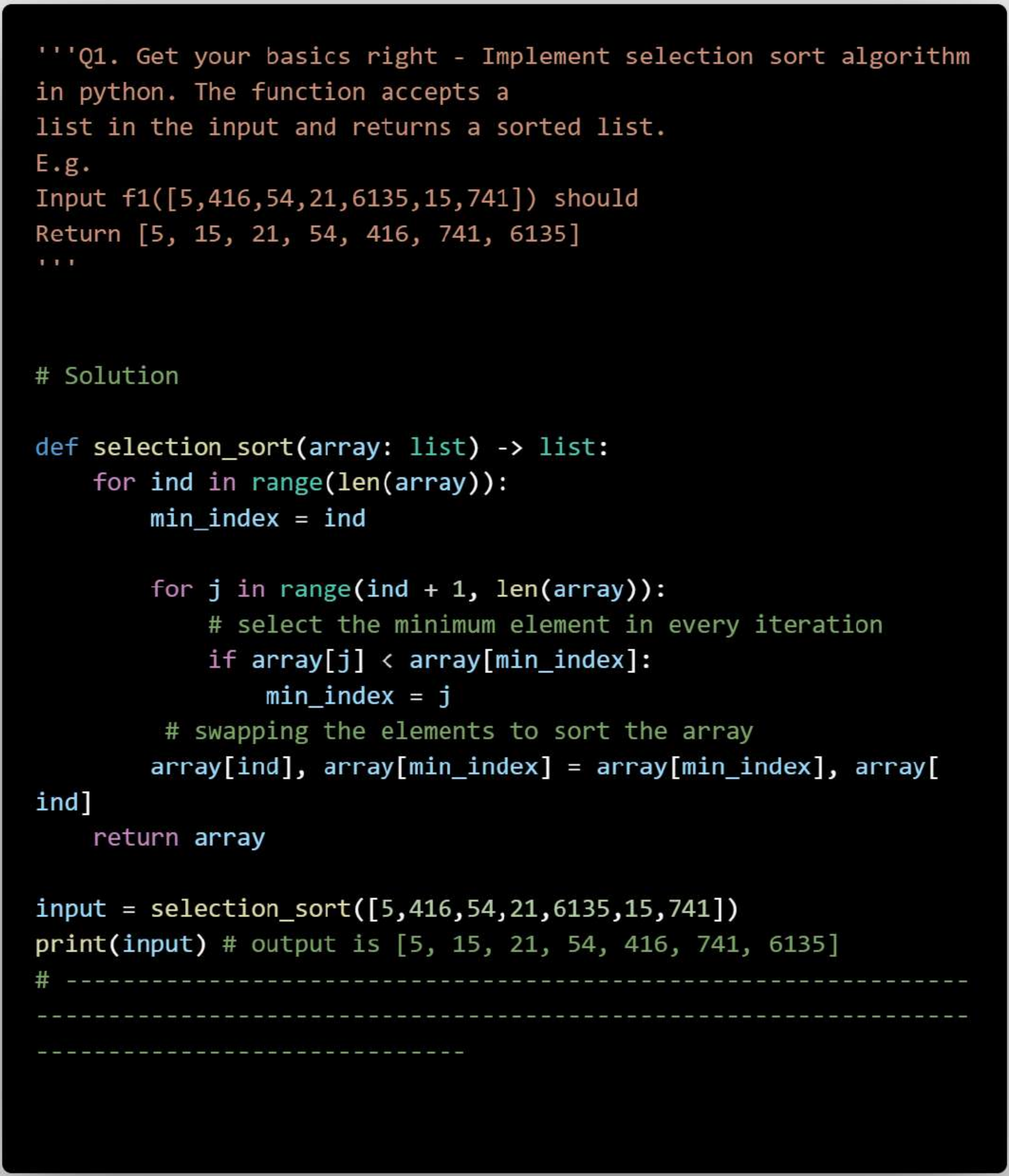
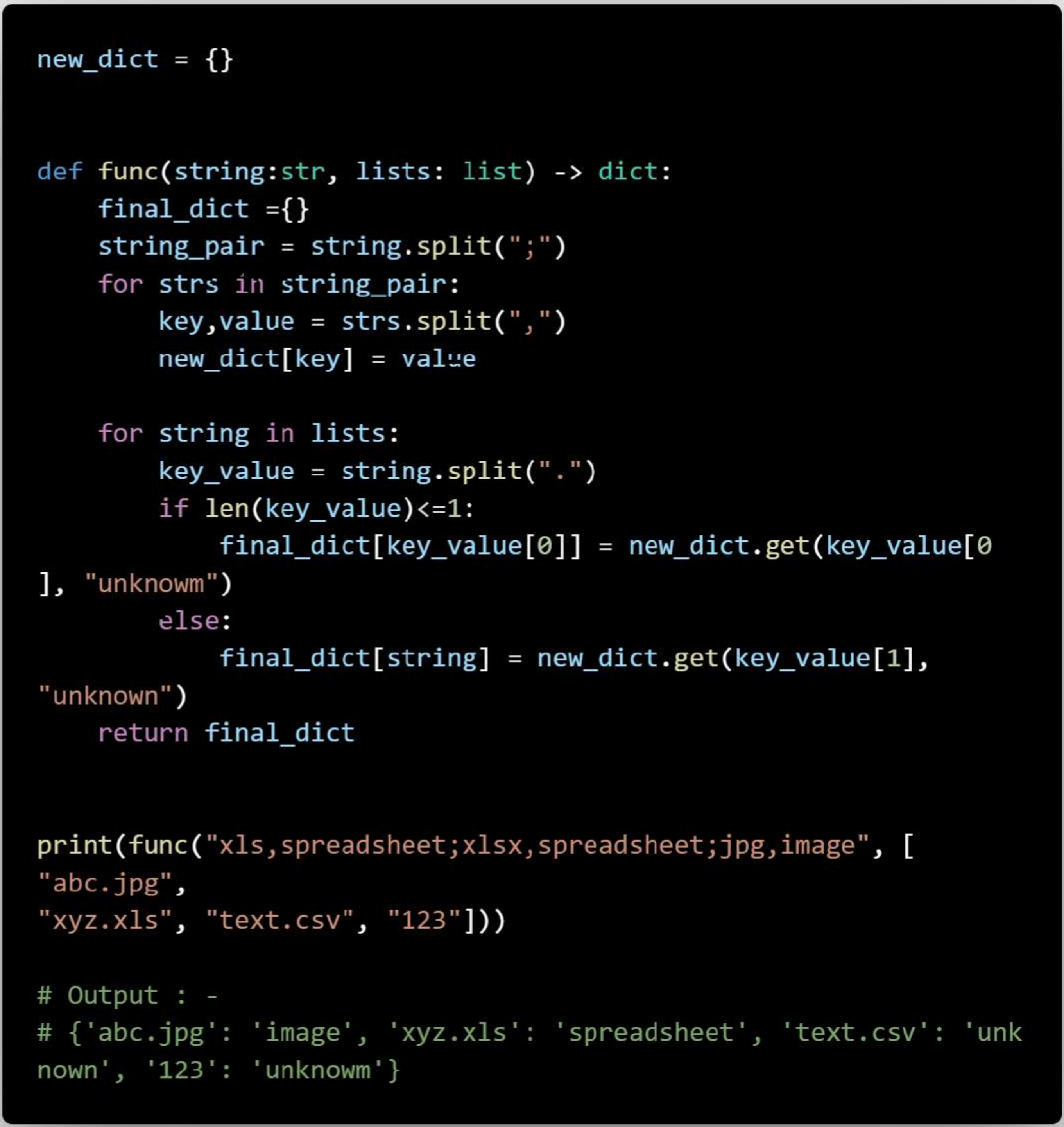
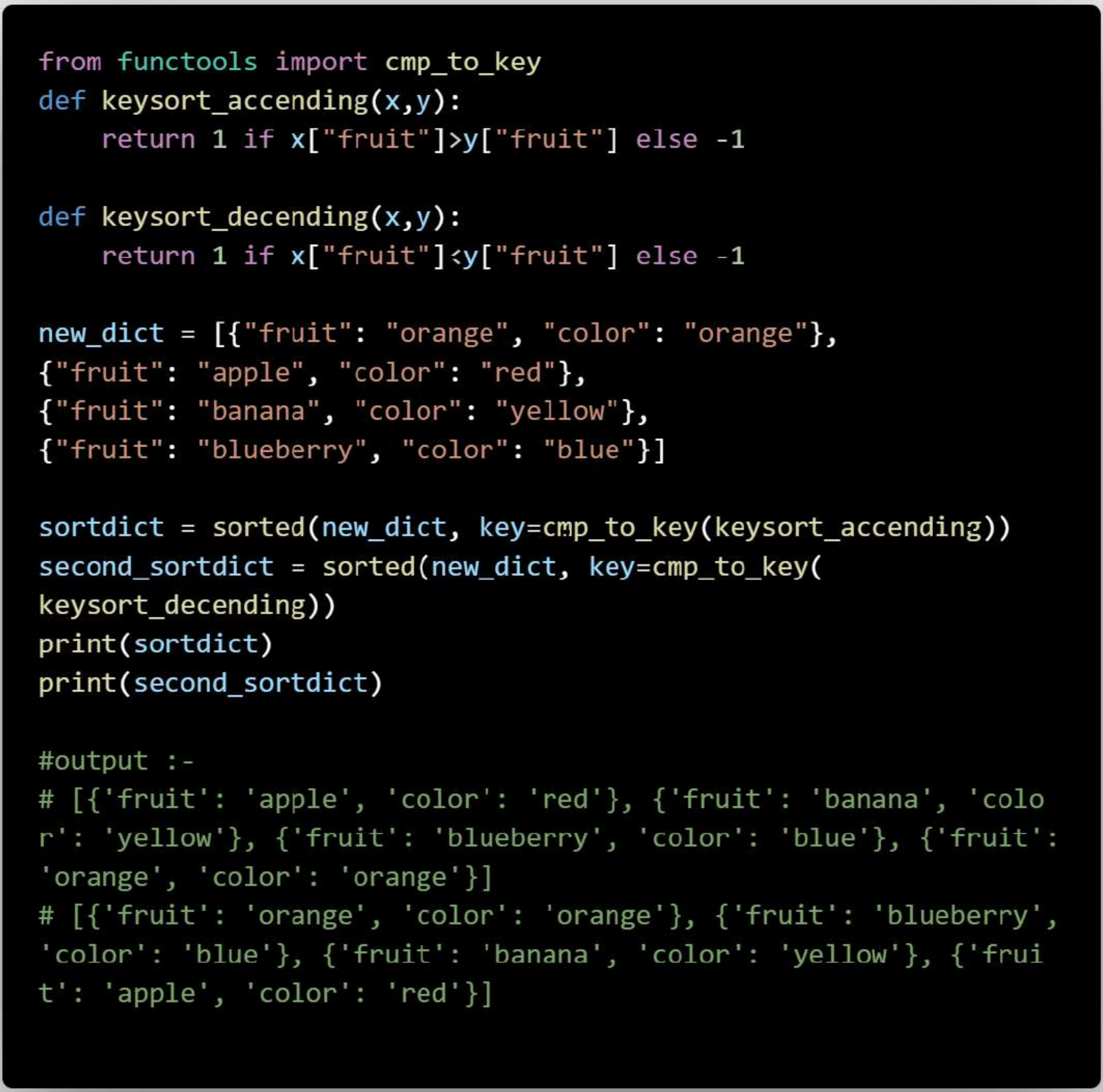


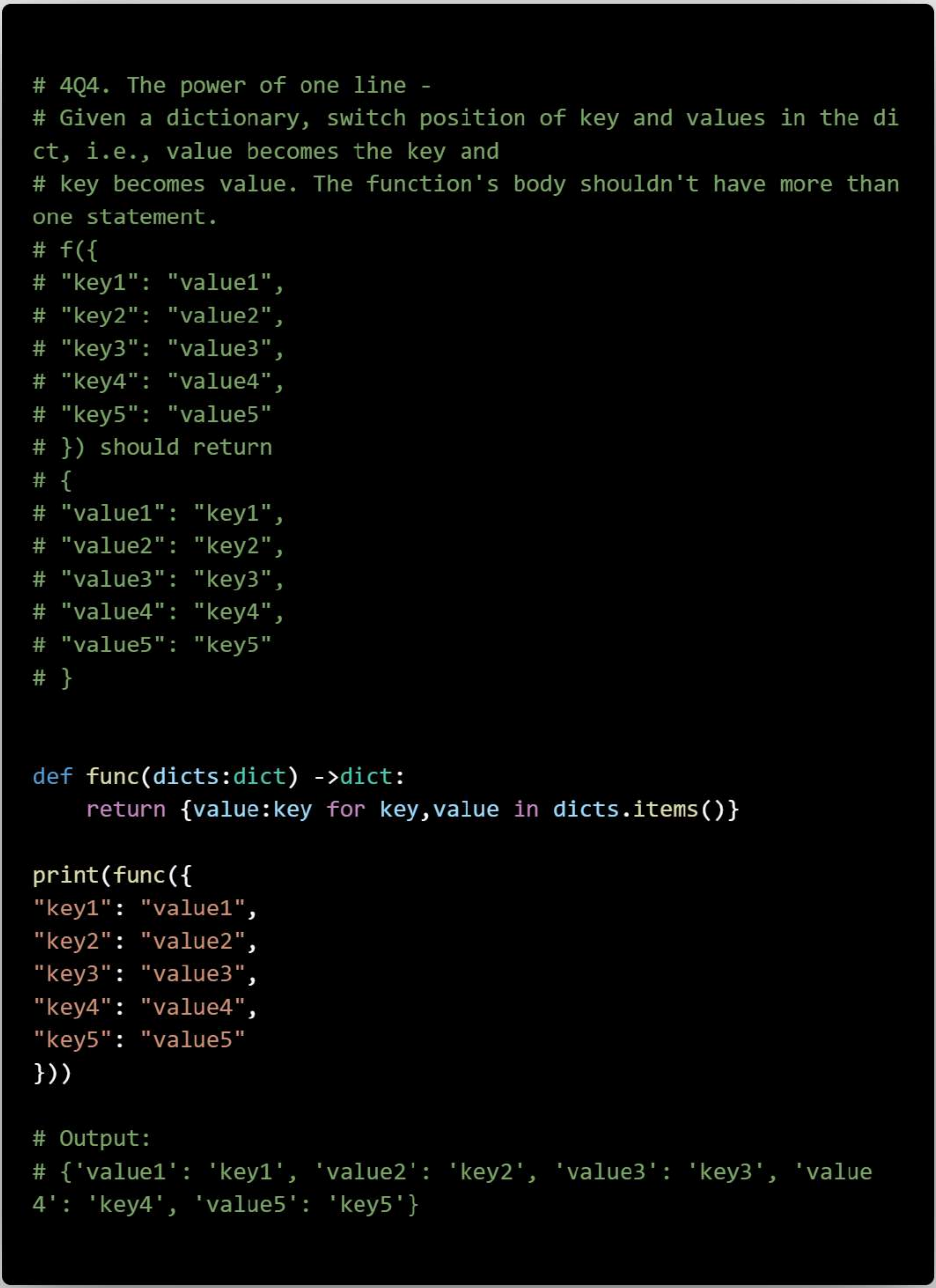
```
'''Q1. Get your basics right - Implement selection sort algorithm
in python. The function accepts a
list in the input and returns a sorted list.
E.g.
Input f1([5,416,54,21,6135,15,741]) should
Return [5, 15, 21, 54, 416, 741, 6135]
'''


# Solution

def selection_sort(array: list) -> list:
    for ind in range(len(array)):
        min_index = ind

        for j in range(ind + 1, len(array)):
            # select the minimum element in every iteration
            if array[j] < array[min_index]:
                min_index = j
         # swapping the elements to sort the array
        array[ind], array[min_index] = array[min_index], array[
ind]
    return array

input = selection_sort([5,416,54,21,6135,15,741])
print(input) # output is [5, 15, 21, 54, 416, 741, 6135]
# -----------------------------------------------------------------
-----------------------------------------------------------------
------------------------------
```



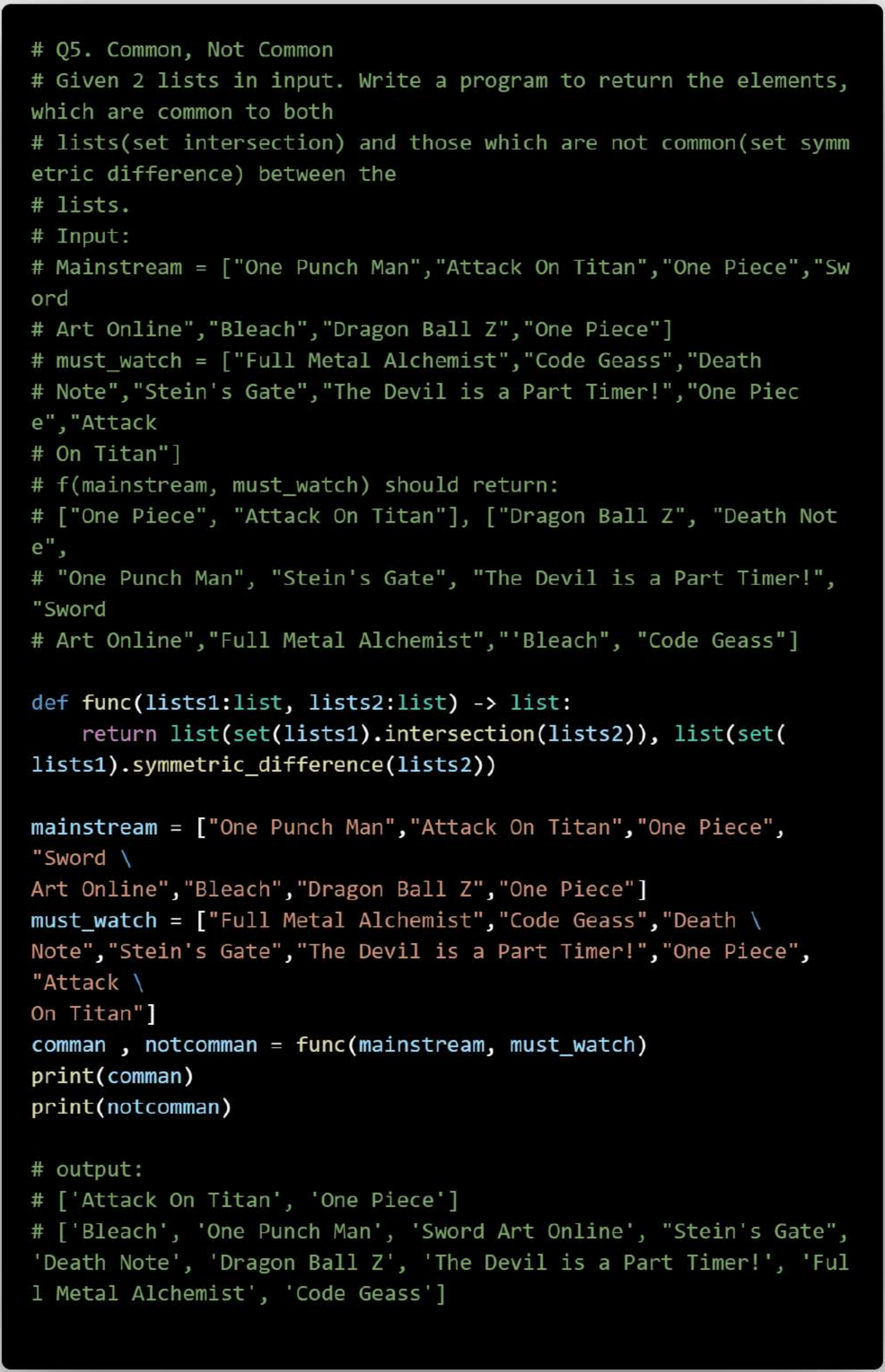
```python
new_dict = {}


def func(string:str, lists: list) -> dict:
    final_dict ={}
    string_pair = string.split(";")
    for strs in string_pair:
        key,value = strs.split(",")
        new_dict[key] = value

    for string in lists:
        key_value = string.split(".")
        if len(key_value)<=1:
            final_dict[key_value[0]] = new_dict.get(key_value[0
], "unknowm")
        else:
            final_dict[string] = new_dict.get(key_value[1],
"unknown")
    return final_dict


print(func("xls,spreadsheet;xlsx,spreadsheet;jpg,image", [
"abc.jpg",
"xyz.xls", "text.csv", "123"]))

# Output : -
# {'abc.jpg': 'image', 'xyz.xls': 'spreadsheet', 'text.csv': 'unk
nown', '123': 'unknowm'}
```



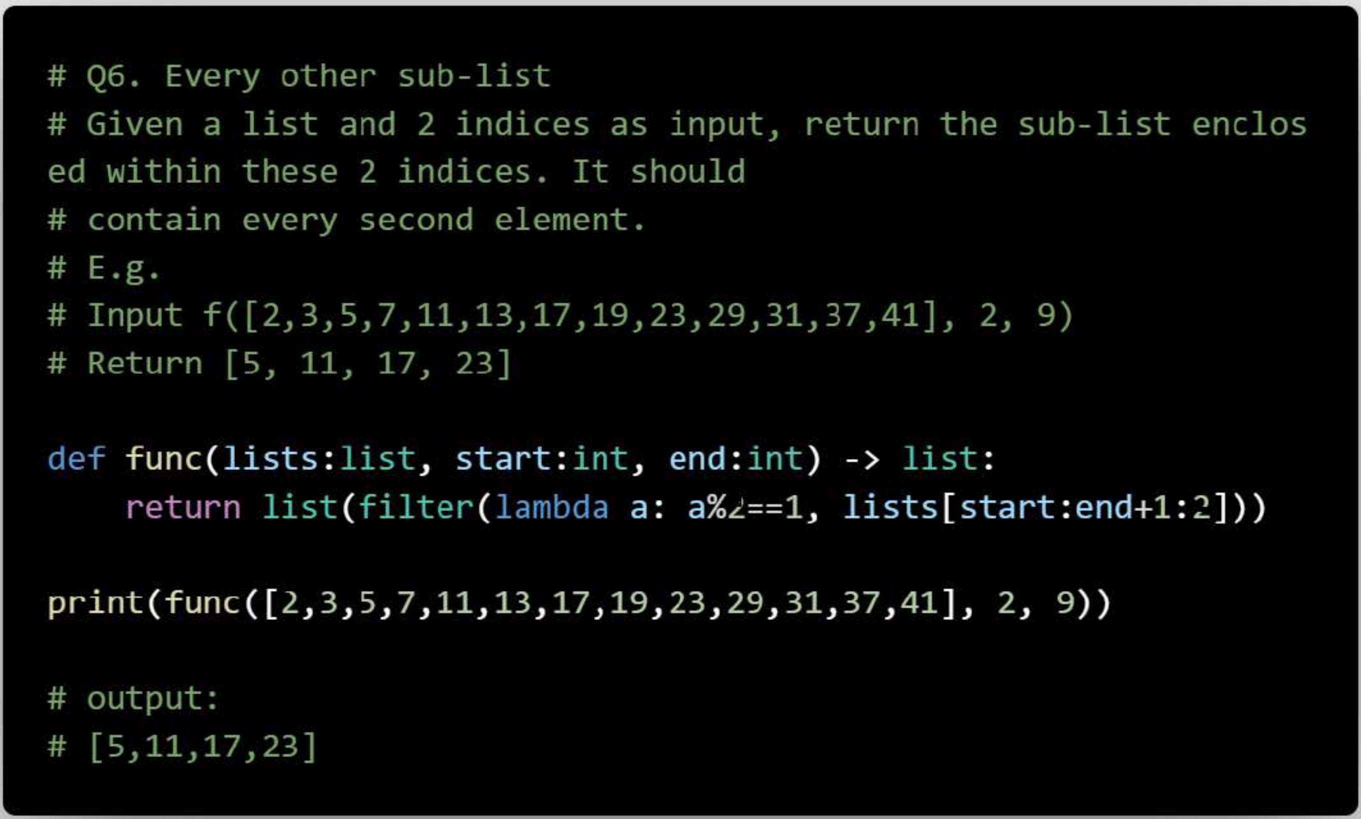
```
from functools import cmp_to_key
def keysort_accending(x,y):
    return 1 if x["fruit"]>y["fruit"] else -1

def keysort_decending(x,y):
    return 1 if x["fruit"]<y["fruit"] else -1

new_dict = [{"fruit": "orange", "color": "orange"},
{"fruit": "apple", "color": "red"},
{"fruit": "banana", "color": "yellow"},
{"fruit": "blueberry", "color": "blue"}]

sortdict = sorted(new_dict, key=cmp_to_key(keysort_accending))
second_sortdict = sorted(new_dict, key=cmp_to_key(
keysort_decending))
print(sortdict)
print(second_sortdict)

#output :-
# [{'fruit': 'apple', 'color': 'red'}, {'fruit': 'banana', 'colo
r': 'yellow'}, {'fruit': 'blueberry', 'color': 'blue'}, {'fruit':
'orange', 'color': 'orange'}]
# [{'fruit': 'orange', 'color': 'orange'}, {'fruit': 'blueberry',
'color': 'blue'}, {'fruit': 'banana', 'color': 'yellow'}, {'frui
t': 'apple', 'color': 'red'}]
```



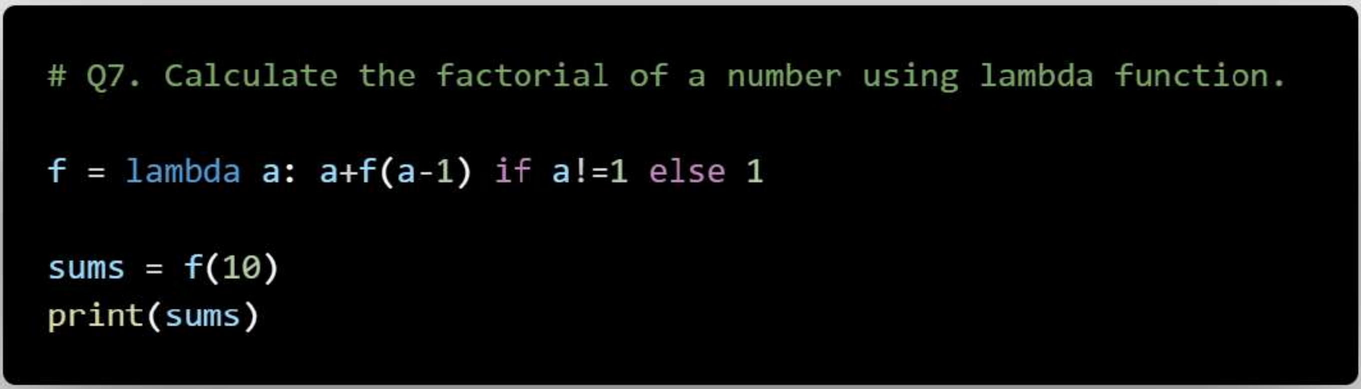
```python
# 4Q4. The power of one line -
# Given a dictionary, switch position of key and values in the di
ct, i.e., value becomes the key and
# key becomes value. The function's body shouldn't have more than
one statement.
# f({
# "key1": "value1",
# "key2": "value2",
# "key3": "value3",
# "key4": "value4",
# "key5": "value5"
# }) should return
# {
# "value1": "key1",
# "value2": "key2",
# "value3": "key3",
# "value4": "key4",
# "value5": "key5"
# }


def func(dicts:dict) ->dict:
    return {value:key for key,value in dicts.items()}

print(func({
"key1": "value1",
"key2": "value2",
"key3": "value3",
"key4": "value4",
"key5": "value5"
}))

# Output:
# {'value1': 'key1', 'value2': 'key2', 'value3': 'key3', 'value
4': 'key4', 'value5': 'key5'}
```

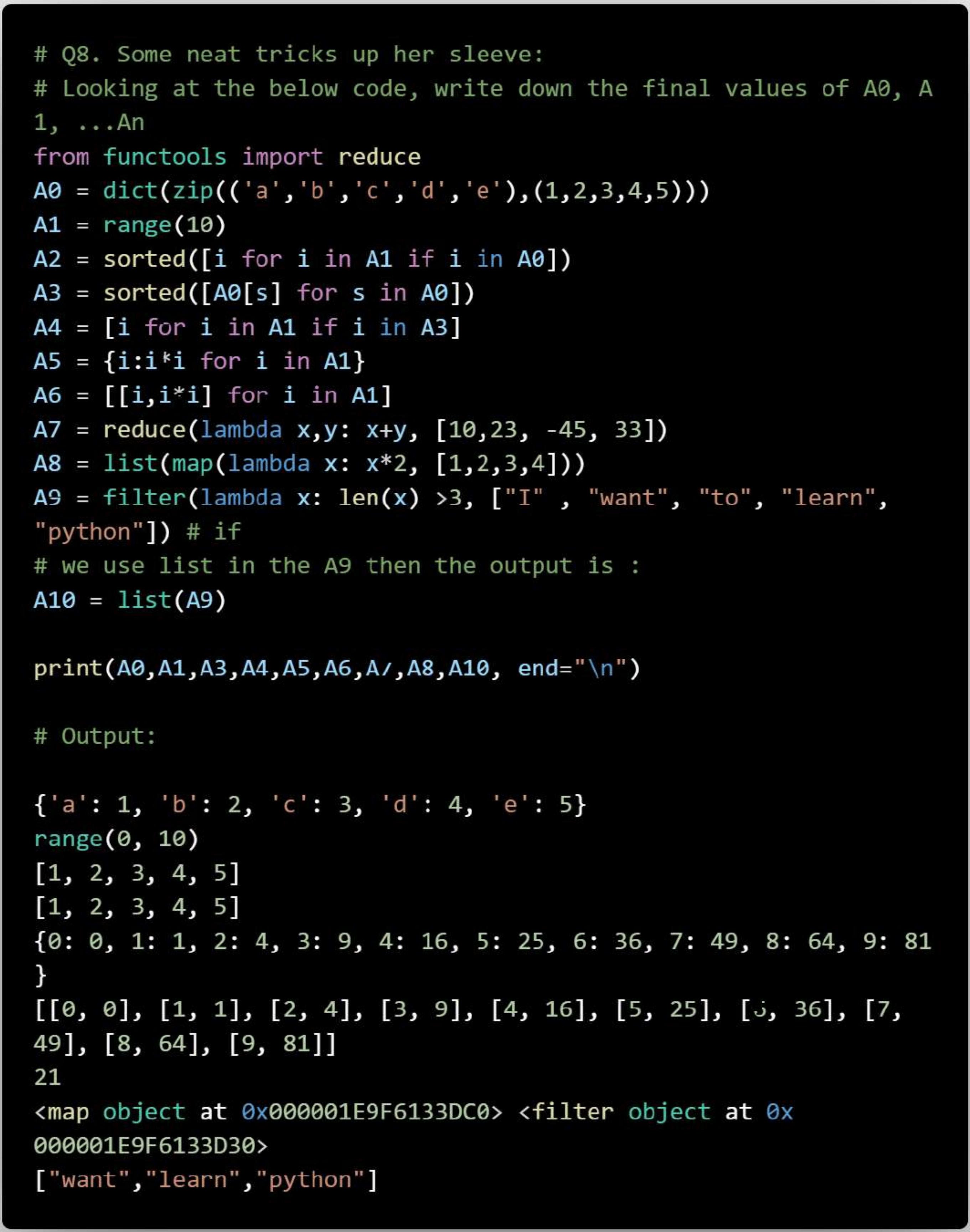


```
# Q5. Common, Not Common
# Given 2 lists in input. Write a program to return the elements,
which are common to both
# lists(set intersection) and those which are not common(set symm
etric difference) between the
# lists.
# Input:
# Mainstream = ["One Punch Man","Attack On Titan","One Piece","Sw
ord
# Art Online","Bleach","Dragon Ball Z","One Piece"]
# must_watch = ["Full Metal Alchemist","Code Geass","Death
# Note","Stein's Gate","The Devil is a Part Timer!","One Piec
e","Attack
# On Titan"]
# f(mainstream, must_watch) should return:
# ["One Piece", "Attack On Titan"], ["Dragon Ball Z", "Death Not
e",
# "One Punch Man", "Stein's Gate", "The Devil is a Part Timer!",
"Sword
# Art Online","Full Metal Alchemist","'Bleach", "Code Geass"]

def func(lists1:list, lists2:list) -> list:
    return list(set(lists1).intersection(lists2)), list(set(
lists1).symmetric_difference(lists2))

mainstream = ["One Punch Man","Attack On Titan","One Piece",
"Sword \
Art Online","Bleach","Dragon Ball Z","One Piece"]
must_watch = ["Full Metal Alchemist","Code Geass","Death \
Note","Stein's Gate","The Devil is a Part Timer!","One Piece",
"Attack \
On Titan"]
comman , notcomman = func(mainstream, must_watch)
print(comman)
print(notcomman)

# output:
# ['Attack On Titan', 'One Piece']
# ['Bleach', 'One Punch Man', 'Sword Art Online', "Stein's Gate",
'Death Note', 'Dragon Ball Z', 'The Devil is a Part Timer!', 'Ful
l Metal Alchemist', 'Code Geass']
```

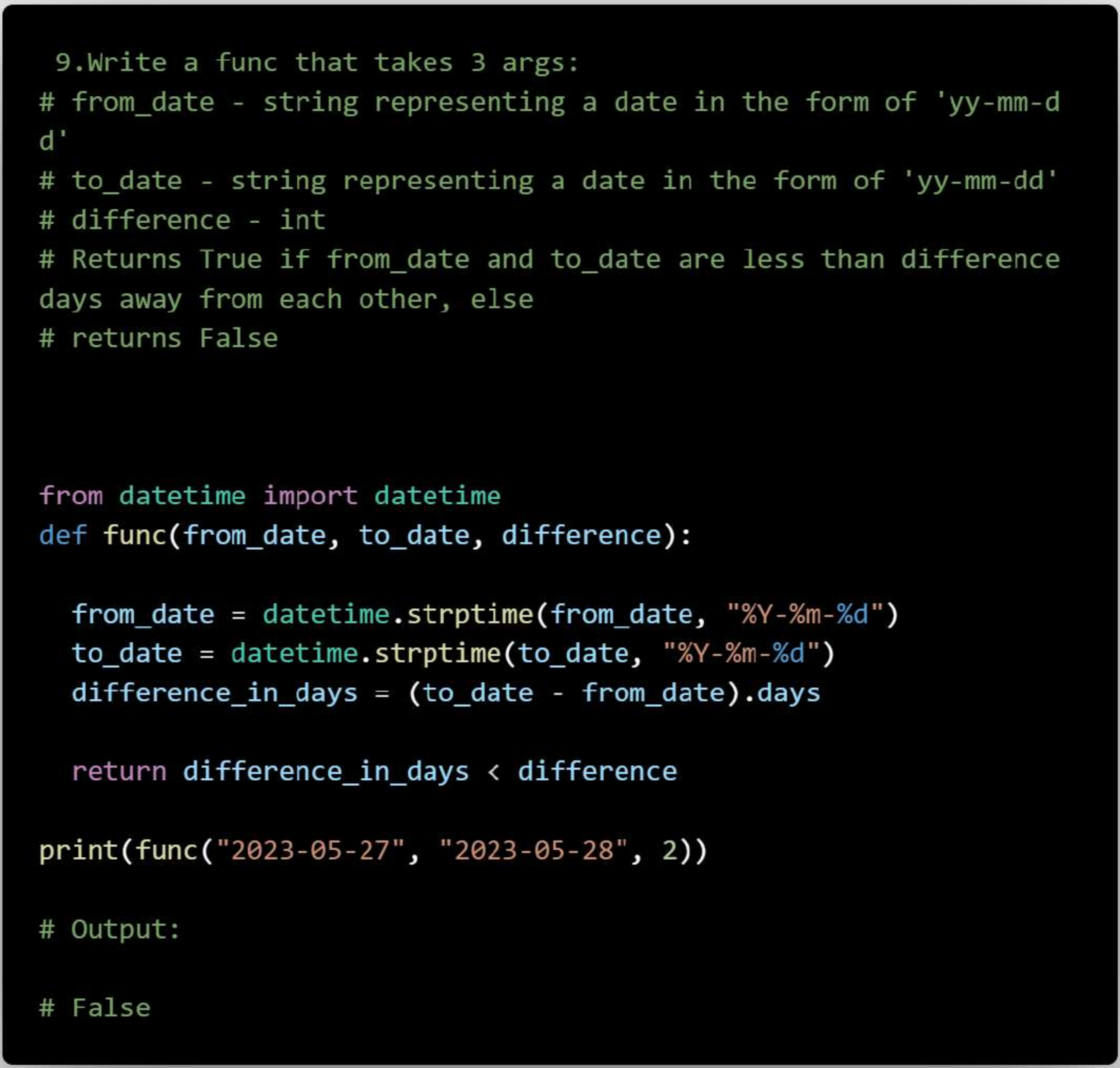


```python
# Q6. Every other sub-list
# Given a list and 2 indices as input, return the sub-list enclos
ed within these 2 indices. It should
# contain every second element.
# E.g.
# Input f([2,3,5,7,11,13,17,19,23,29,31,37,41], 2, 9)
# Return [5, 11, 17, 23]

def func(lists:list, start:int, end:int) -> list:
    return list(filter(lambda a: a%2==1, lists[start:end+1:2]))

print(func([2,3,5,7,11,13,17,19,23,29,31,37,41], 2, 9))

# output:
# [5,11,17,23]
```

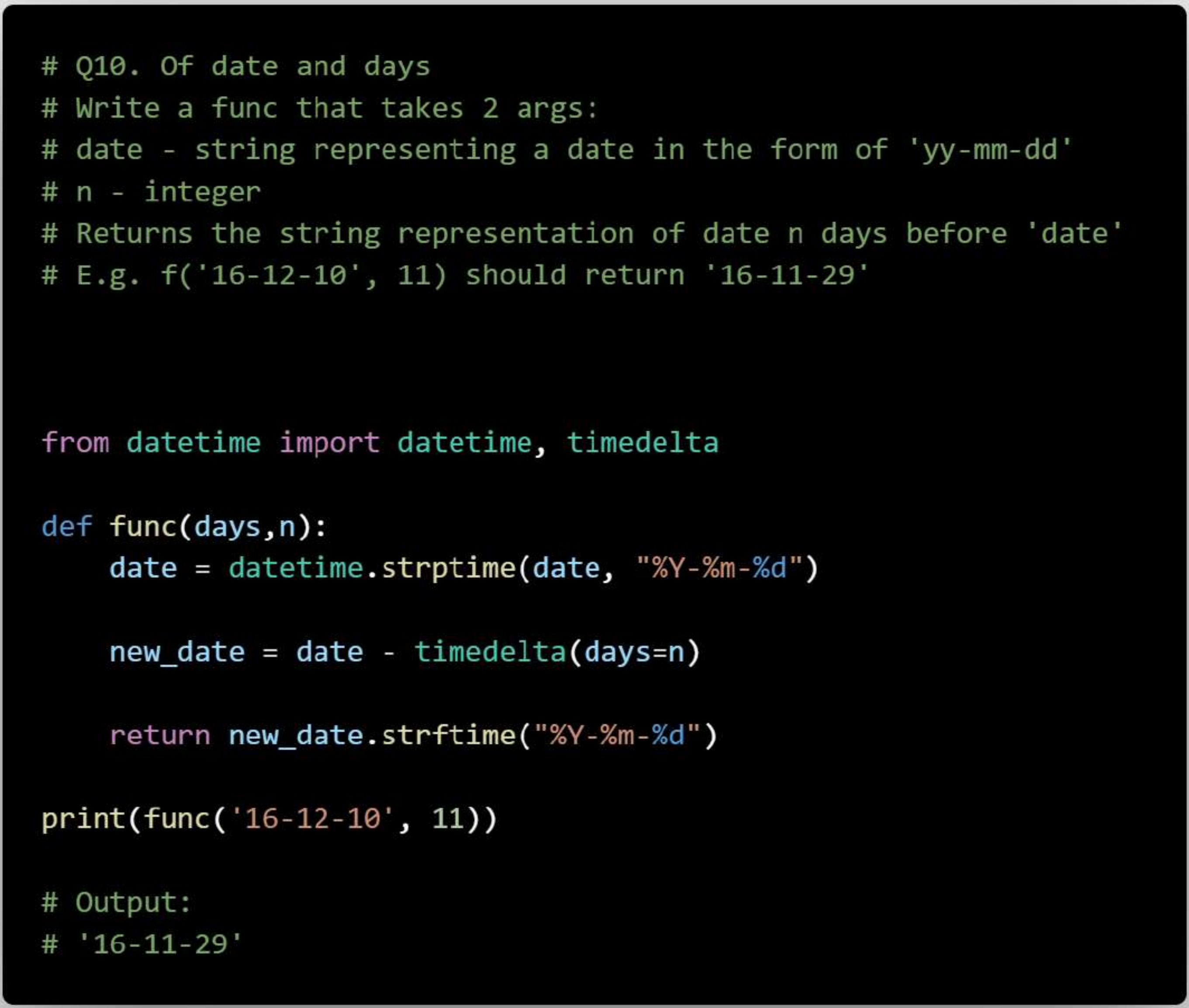

```
# Q7. Calculate the factorial of a number using lambda function.

f = lambda a: a+f(a-1) if a!=1 else 1

sums = f(10)
print(sums)
```



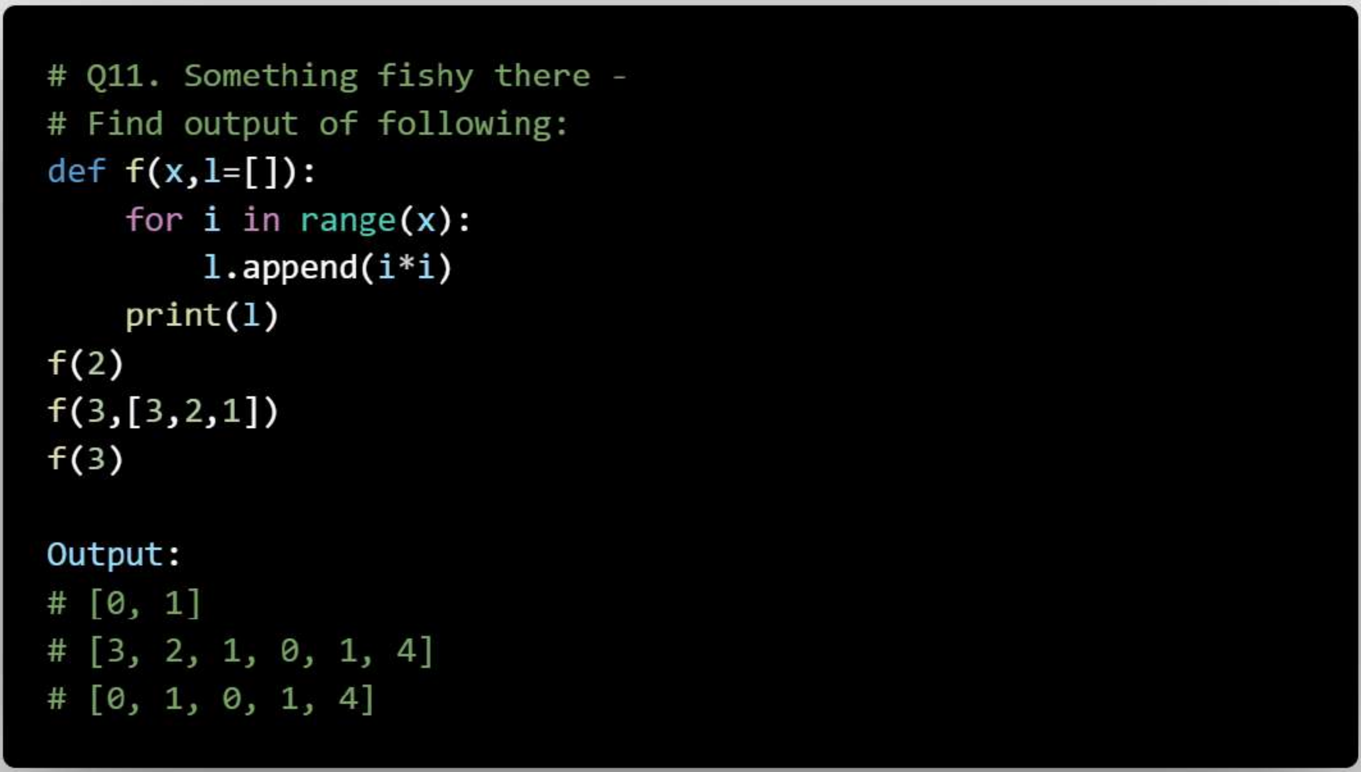
```
# Q8. Some neat tricks up her sleeve:
# Looking at the below code, write down the final values of A0, A
1, ...An
from functools import reduce
A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
A1 = range(10)
A2 = sorted([i for i in A1 if i in A0])
A3 = sorted([A0[s] for s in A0])
A4 = [i for i in A1 if i in A3]
A5 = {i:i*i for i in A1}
A6 = [[i,i*i] for i in A1]
A7 = reduce(lambda x,y: x+y, [10,23, -45, 33])
A8 = list(map(lambda x: x*2, [1,2,3,4]))
A9 = filter(lambda x: len(x) >3, ["I" , "want", "to", "learn",
"python"]) # if
# we use list in the A9 then the output is :
A10 = list(A9)

print(A0,A1,A3,A4,A5,A6,A7,A8,A10, end="\n")

# Output:

{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
range(0, 10)
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81
}
[[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7,
49], [8, 64], [9, 81]]
21
<map object at 0x000001E9F6133DC0> <filter object at 0x
000001E9F6133D30>
["want","learn","python"]
```



```
 9.Write a func that takes 3 args:
# from_date - string representing a date in the form of 'yy-mm-d
d'
# to_date - string representing a date in the form of 'yy-mm-dd'
# difference - int
# Returns True if from_date and to_date are less than difference
days away from each other, else
# returns False



from datetime import datetime
def func(from_date, to_date, difference):

  from_date = datetime.strptime(from_date, "%Y-%m-%d")
  to_date = datetime.strptime(to_date, "%Y-%m-%d")
  difference_in_days = (to_date - from_date).days

  return difference_in_days < difference

print(func("2023-05-27", "2023-05-28", 2))

# Output:

# False
```



```python
# Q10. Of date and days
# Write a func that takes 2 args:
# date - string representing a date in the form of 'yy-mm-dd'
# n - integer
# Returns the string representation of date n days before 'date'
# E.g. f('16-12-10', 11) should return '16-11-29'



from datetime import datetime, timedelta

def func(days,n):
    date = datetime.strptime(date, "%Y-%m-%d")

    new_date = date - timedelta(days=n)

    return new_date.strftime("%Y-%m-%d")

print(func('16-12-10', 11))

# Output:
# '16-11-29'
```



```
# Q11. Something fishy there -
# Find output of following:
def f(x,l=[]):
    for i in range(x):
        l.append(i*i)
    print(l)
f(2)
f(3,[3,2,1])
f(3)

Output:
# [0, 1]
# [3, 2, 1, 0, 1, 4]
# [0, 1, 0, 1, 4]
```