1. Decrypting the code:

def decrypt(text, key):

decrypted\_text = ""

for char in text:

if char.isalpha(): # Check if character is alphabetic

shifted = ord(char) - key # Reverse shift by subtracting the key

if char.islower(): # Handle lowercase letters

if shifted < ord('a'):

shifted += 26

elif char.isupper(): # Handle uppercase letters

if shifted < ord('A'):

shifted += 26

decrypted\_text += chr(shifted)

else:

decrypted\_text += char # Non-alphabet characters remain unchanged

return decrypted\_text

# Encrypted code (from the image)

encrypted\_text = """

tybony\_inevnoyr = 100

zl\_qvpg = {'xr11': 'inyhr1', 'xr12': 'inyhr2', 'xr13': 'inyhr3']

qrs cebprff\_ahzoref():

tybony tybony\_inevnoyr

ybpny\_inevnoyr = 5

ahzoref= [1, 2, 3, 4, 5]

juvyr ybpny\_inevnoyr > 0:

vs ybpny\_inevnoyr % 2 == 0:

ahzoref.erzbir(ybpny\_inevnoyr)

ybpny\_inevnoyr -= 1

erghea ahzoref

zl\_frg = {1, 2, 3, 4, 5, 5, 4, 3, 2, 1}

erfhyg = cebprff\_ahzoref (ahzoref=zl\_frg)

qrs zbqvsl\_qvpg():

ybpny\_inevnoyr = 10

zl\_qvpg['xr14'] = ybpny\_inevnoyr

zbqvsl\_qvpg(5)

qrs hcqngr\_tybony():

tybony tybony\_inevnoyr

tybony\_inevnoyr += 10

sbe v va enatr(5):

cevag(v)

V += 1

vs zl\_frg vf abg Abar naq z1\_qvpg['xr14'] == 10:

cevag("Pbaqvgvba zrg!")

vs 5 abg va zl\_qvpg:

cevag("5 abg sbhaq va gur qvpgvbane1!")

cevag(tybony\_inevnoyr)

cevag(zl\_qvpg)

cevag(zl\_frg)

"""

# Finding the key: From the looks of the text, it appears to be encoded using ROT13, which means the key is 13.

key = 13

decrypted\_text = decrypt(encrypted\_text, key)

print(decrypted\_text)

1. Decrypted Code:

global\_variable = 100

my\_dict = {'ke11': 'value1', 'ke12': 'value2', 'ke13': 'value3']

def process\_numbers():

global global\_variable

local\_variable = 5

numbers= [1, 2, 3, 4, 5]

while local\_variable > 0:

if local\_variable % 2 == 0:

numbers.remove(local\_variable)

local\_variable -= 1

return numbers

my\_set = {1, 2, 3, 4, 5, 5, 4, 3, 2, 1}

result = process\_numbers (numbers=my\_set)

def modify\_dict():

local\_variable = 10

my\_dict['ke14'] = local\_variable

modify\_dict(5)

def update\_global():

global global\_variable

global\_variable += 10

for i in range(5):

print(i)

I += 1

if my\_set is not None and m1\_dict['ke14'] == 10:

print("Condition met!")

if 5 not in my\_dict:

print("5 not found in the dictionar1!")

print(global\_variable)

print(my\_dict)

print(my\_set)

1. Final Correct Code:

global\_variable = 100

my\_dict = {'ke11': 'value1', 'ke12': 'value2', 'ke13': 'value3'}

def process\_numbers():

global global\_variable

local\_variable = 5

numbers = [1, 2, 3, 4, 5]

while local\_variable > 0:

if local\_variable % 2 == 0:

numbers.remove(local\_variable)

local\_variable -= 1

return numbers

my\_set = {1, 2, 3, 4, 5} # Set removes duplicates automatically

result = process\_numbers() # Removed argument

def modify\_dict():

local\_variable = 10

my\_dict['ke14'] = local\_variable

modify\_dict() # Removed argument

def update\_global():

global global\_variable

global\_variable += 10

for i in range(5):

print(i) # Removed invalid "I += 1"

if my\_set is not None and my\_dict['ke14'] == 10:

print("Condition met!")

if 5 not in my\_dict:

print("5 not found in the dictionary!") # Corrected typo

print(global\_variable)

print(my\_dict)

print(my\_set)

Issues and Corrections:

1. Error in calling process\_numbers function:
   * Error: process\_numbers(numbers=my\_set) is incorrect since process\_numbers() doesn't take any parameters.
   * Fix: Simply call process\_numbers() without any arguments.

Correction:

“result = process\_numbers()”

1. Error in modify\_dict function call:
   * Error: modify\_dict(5) is incorrect because the function doesn't accept any arguments.
   * Fix: Call modify\_dict() without arguments.

Correction:

“modify\_dict()”

1. Unnecessary assignment inside for loop:
   * Error: In for i in range(5), there is an invalid statement I += 1 (capital "I"). This doesn't match the loop variable i.
   * Fix: Remove the I += 1 line, as it's unnecessary here.

Correction:

“for i in range(5):

print(i)”

1. Typo in dictionary name:
   * Error: m1\_dict is referenced incorrectly. It should be my\_dict.
   * Fix: Change m1\_dict to my\_dict.

Correction:

“if my\_set is not None and my\_dict['ke14'] == 10:

print("Condition met!")”

1. Typo in string "dictionar1":
   * Error: The word "dictionar1" in the second print statement should be "dictionary".
   * Fix: Correct the typo.

Correction:

“if 5 not in my\_dict:

print("5 not found in the dictionary!")”

1. Duplicate values in my\_set:
   * Error: my\_set = {1, 2, 3, 4, 5, 5, 4, 3, 2, 1} has duplicate values. Sets automatically remove duplicates in Python.
   * Fix: No need to include duplicates in sets; Python handles that automatically.

Correction:

“my\_set = {1, 2, 3, 4, 5}”

Changes Explained (With Comments):  
  
  
1. Fixes for Function Calls: Made sure the function calls matched the function definitions by fixing them.   
  
2. Set Handling: Python manages the removal of duplicate values from the set automatically.   
  
3. Typo Corrections: We corrected a few little typos, including dictionar1 and m1\_dict.   
  
4. Unnecessary Code: The statement I += 1 in the loop was eliminated.