CHAPTER 5

EXERCISES

[1](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1134). What does the code for an empty dictionary look like?

We can use any one of the below methods to create an empty dictionary

* my\_dictionary = {}
* my\_dictionary = dict()

[2](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1135). What does a dictionary value with a key 'foo' and a value 42 look like?

* My\_dictionary = {‘foo’: 42}

[3](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1136). What is the main difference between a dictionary and a list?

* Items in dictionaries are unordered whereas list is ordered.
* We can access the list data values using index whereas in dictionary we can access the value using keys.

[4](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1137). What happens if you try to access spam['foo'] if spam is {'bar': 100}?

* Trying to access a key that does not exist in a dictionary will result in a KeyError error message.

[5](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1138). If a dictionary is stored in spam, what is the difference between the expressions 'cat' in spam and 'cat' in spam.keys()?

* 'cat' in spam is essentially a shorter version of writing 'cat' in spam.keys().
* Both the expressions checks for a key named cat in the dictionary.

[6](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1139). If a dictionary is stored in spam, what is the difference between the expressions 'cat' in spam and 'cat' in spam.values()?

* 'cat' in spam is essentially a shorter version of writing 'cat' in spam.keys().
* The above expression checks whether ‘cat’ is one of the keys of the dictionary or not.
* Whereas 'cat' in spam.values() is used to check whether the value named ‘cat’ is present or not.

[7](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1140). What is a shortcut for the following code?

if 'color' not in spam:  
    spam['color'] = 'black'

* We can use setdefault() method as a short form for the above code like **spam.setdefault(‘color’, black)**

[8](https://automatetheboringstuff.com/2e/chapter5/#calibre_link-1141). What module and function can be used to “pretty print” dictionary values?

* pprint module
* pprint() and pformat() functions

***Chess Dictionary Validator***

In this chapter, we used the dictionary value {'1h': 'bking', '6c': 'wqueen', '2g': 'bbishop', '5h': 'bqueen', '3e': 'wking'} to represent a chess board. Write a function named isValidChessBoard() that takes a dictionary argument and returns True or False depending on if the board is valid.

A valid board will have exactly one black king and exactly one white king. Each player can only have at most 16 pieces, at most 8 pawns, and all pieces must be on a valid space from '1a' to '8h'; that is, a piece can’t be on space '9z'. The piece names begin with either a 'w' or 'b' to represent white or black, followed by 'pawn', 'knight', 'bishop', 'rook', 'queen', or 'king'. This function should detect when a bug has resulted in an improper chess board.

Program:

def isValidChessBoard(board):

piecesCount = {"b": 0, "w": 0}

pawnCount = {"b": 0, "w": 0}

hasKing = {"b": False, "w": False}

letterAxis = ("a", "b", "c", "d", "e", "f", "g", "h")

pieceColour = ("b", "w")

pieceType = ("pawn", "knight", "bishop", "rook", "queen", "king")

# each player has <= 16 pieces

for pos, i in board.items():

# check position value

# all pieces must be on valid space from '1a' to '8h'

if int(pos[0]) >= 9:

print("SpacesError")

return False

if pos[1] not in letterAxis:

print("y AxisError")

return False

# check piece data

if i != "":

# piece names begin with 'w' or 'b'

if i[0] not in pieceColour:

print("WhiteOrBlackError")

return False

thisPieceColour = i[0]

piecesCount[thisPieceColour] += 1

if piecesCount[thisPieceColour] >= 17:

print("TotalPieceError")

return False

# piece names must follow with 'pawn', 'knight', 'bishop', 'rook', 'queen', 'king'

thisPieceType = i[1:]

if thisPieceType not in pieceType:

print("PieceTypeError")

return False

elif thisPieceType == "pawn":

pawnCount[thisPieceColour] += 1

# each player has <= 8 pawns

if pawnCount[thisPieceColour] >= 9:

print("PawnError")

return False

elif thisPieceType == "king":

# one black king and one white king

if hasKing[thisPieceColour] == True:

print("AlreadyHasKingError")

hasKing[thisPieceColour] = True

if list(hasKing.values()) != [True, True]:

print("MissingKingError")

return False

return True

board = {

"1a": "bking",

"2a": "bqueen",

"3a": "brook",

"4a": "brook",

"5a": "bknight",

"6a": "bknight",

"7a": "bbishop",

"8a": "bbishop",

"1b": "bpawn",

"2b": "bpawn",

"3b": "bpawn",

"4b": "bpawn",

"5b": "bpawn",

"6b": "bpawn",

"7b": "bpawn",

"8b": "bpawn",

"1c": "wking",

"2c": "wqueen",

"3c": "wrook",

"4c": "wrook",

"5c": "wbishop",

"6c": "wbishop",

"7c": "wknight",

"8c": "wknight",

"1d": "",

"2d": "",

"3d": "",

"4d": "",

"5d": "",

"6d": "",

"7d": "",

"8d": "",

"1e": "wpawn",

"2e": "wpawn",

"3e": "wpawn",

"4e": "wpawn",

"5e": "wpawn",

"6e": "wpawn",

"7e": "wpawn",

"8e": "wpawn",

"1f": "",

"2f": "",

"3f": "",

"4f": "",

"5f": "",

"6f": "",

"7f": "",

"8f": "",

"1g": "",

"2g": "",

"3g": "",

"4g": "",

"5g": "",

"6g": "",

"7g": "",

"8g": "",

"1h": "",

"2h": "",

"3h": "",

"4h": "",

"5h": "",

"6h": "",

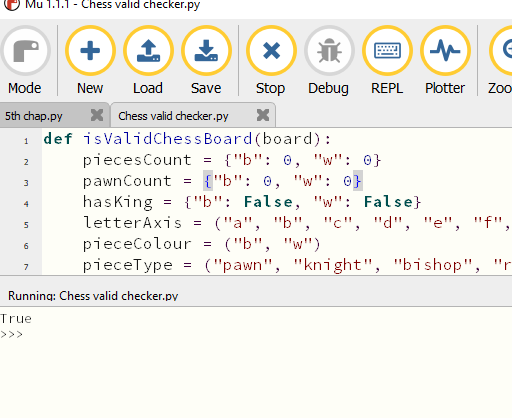
"7h": "",

"8h": ""

}

print(isValidChessBoard(board))

Output:



#### Fantasy Game Inventory

You are creating a fantasy video game. The data structure to model the player’s inventory will be a dictionary where the keys are string values describing the item in the inventory and the value is an integer value detailing how many of that item the player has. For example, the dictionary value {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12} means the player has 1 rope, 6 torches, 42 gold coins, and so on.

Write a function named displayInventory() that would take any possible “inventory” and display it like the following:

Inventory:  
12 arrow  
42 gold coin  
1 rope  
6 torch  
1 dagger  
Total number of items: 62

Hint: You can use a for loop to loop through all the keys in a dictionary.

# inventory.py  
stuff = {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12}  
  
def displayInventory(inventory):  
    print("Inventory:")  
    item\_total = 0  
    for k, v in inventory.items():  
        # FILL THIS PART IN  
    print("Total number of items: " + str(item\_total))  
  
displayInventory(stuff)

Program:

stuff = {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12}

import inflect

s = inflect.engine()

def displayInventory(inventory):

print("Inventory:")

t=0

for key, value in inventory.items():

t=t+value

if value != 1:

key = s.plural(key)

print(value, key)

print('Total number of items ',t)

displayInventory(stuff)

OUTPUT:

Graphical user interface, text, application

Description automatically generated

#### List to Dictionary Function for Fantasy Game Inventory

Imagine that a vanquished dragon’s loot is represented as a list of strings like this:

dragonLoot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']

Write a function named addToInventory(inventory, addedItems), where the inventory parameter is a dictionary representing the player’s inventory (like in the previous project) and the addedItems parameter is a list like dragonLoot. The addToInventory() function should return a dictionary that represents the updated inventory. Note that the addedItems list can contain multiples of the same item. Your code could look something like this:

def addToInventory(inventory, addedItems):  
    # your code goes here  
  
inv = {'gold coin': 42, 'rope': 1}  
dragonLoot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']  
inv = addToInventory(inv, dragonLoot)  
displayInventory(inv)

The previous program (with your displayInventory() function from the previous project) would output the following:

Inventory:  
45 gold coin  
1 rope  
1 ruby  
1 dagger  
  
Total number of items: 48

Program:

stuff = {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12}

dragonLoot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']

import inflect

s = inflect.engine()

def addToInventory(d, l):

for x in l:

if x in d.keys():

d[x]=d[x]+1

else:

d[x]=1

return d

def displayInventory(inventory):

print("Inventory:")

t=0

for key, value in inventory.items():

t=t+value

if value != 1:

key = s.plural(key)

print(value, key)

print('Total number of items ',t)

stuff = addToInventory(stuff, dragonLoot)

displayInventory(stuff)

OUTPUT:

