

Edexcel International GCSE O Level Computer Science

Past Paper 2020 – 2024
Practical Paper 2
Python 20 Marks

DR. SAW MYAT SANDAR

20 marks selection and iteration

2017 P2 #6

6 Open the file named **Q06** in the code editor.

In file **Q06**, the names and years of birth of artists are stored in a 2-dimensional data structure.

Labels for their work need to be created by joining the first letter of their last name, the first letter of their first name and their year of birth.

For example, a label for ('Andy', 'Warhol', 1928) would be 'WA1928'.

Write a program to:

- process each artist to create a label
- store all the labels in the data structure named 'theLabels'
- display the labels for all the artists
- find and display the name and year of birth of the youngest artist.

Your program should function correctly, even if 'theArtists' data structure has more, fewer, or different artists.

You **must** use the data structures in file **Q06**.

Save your amended code as **Q06FINISHED** with the correct file extension for the programming language.

```
#-----  
# Name:      Artists  
# Purpose:  
#  
# Author:    CSelby  
#  
# Created:   02/06/2016  
# Copyright: (c) CSelby 2016  
# Licence:   <your licence>  
#-----  
  
theArtists = [ ["Andy", "Warhol", 1928],  
               ["Pablo", "Picasso", 1881],  
               ["Salvador", "Dali", 1904],  
               ["Lavinia", "Fontana", 1552],  
               ["Jackson", "Pollock", 1912],  
               ["Henri", "Matisse", 1869],  
               ["Frida", "Kahlo", 1907],  
               ["Georgia", "O'Keeffe", 1887],  
               ["Kara", "Walker", 1969],  
               ["Yayoi", "Kusama", 1929]  
             ]  
  
theLabels = []    # Put the new user labels into this structure  
  
# Make the artist labels  
for person in theArtists:  
    newRecord = person[1][0] + person[0][0] + str(person[2])  
    theLabels.append (newRecord)  
print ("The new userIDs are: ", theLabels)  
  
# Find and print the youngest person and their birthdate  
maxDate = 0  
for person in theArtists:  
    if person[2] > maxDate:  
        maxDate = person[2]  
        maxPerson = person  
print (maxPerson[0], maxPerson[1], "is youngest", str(maxPerson[2]))
```

2019 P2 #6

5 Ria is a school librarian.

She wants a program to analyse pupil use of the library.

She wants to encourage reading by awarding gold, silver and bronze medals to the three pupils who have read the most books.

Test data has been included in the code.

Open **Q05** in the code editor.

Write a program to calculate and display:

- the total number and average number of books pupils have read
- the IDs of pupils who have read fewer than ten books
- the details of the gold, silver and bronze medal winners.

Your program should function correctly even if the number of pupils in the file is changed.

Save your code as **Q05FINISHED** with the correct file extension for the programming language.

```
#Q05FINISHED

libraryRecord = [
["105MS" , "Marcus" , "Smith" , 25 ],
["103AZ" , "Anthony" , "Zarrent" , 5 ],
["108MW" , "Matt" , "White" , 12 ],
["112DB" , "Denise" , "Bilton" , 58 ],
["124MK" , "Malcolm" , "Kelly" , 26 ],
["116UK" , "Uzere" , "Kevill" , 29 ],
["127AL" , "Abduraheim" , "Leahy" , 94 ],
["124LS" , "Laura" , "Sampras" , 70 ],
["121AP" , "Azra" , "Potter" , 61 ],
["115AC" , "Anthony" , "Calik" , 10 ],
["117PI" , "Pablo" , "Iilyas" , 49 ],
["113MM" , "Mark" , "Montgomerie" , 69 ],
["130FH" , "Felicity" , "Heath" , 11 ],
["132JA" , "Jill" , "Alexander" , 61 ],
["123SG" , "Sara" , "Grimstow" , 9 ],
["134KD" , "Kevin" , "Dawson" , 74 ],
["122AB" , "Andrew" , "Bertwistle" , 42 ],
["125JF" , "Jaide" , "Feehily" , 55 ],
["128JS" , "Justin" , "Slater" , 68 ],
["126CG" , "Colleen" , "Grohl" , 39 ]
]

# -----
# Write your code below this line
gold=0
silver=0
bronze=0
goldID=0
silverID=0
bronzeID=0
```

```
total=0
average=0.0

print("Pupil ID of pupils who have read less than ten books")
for row in range(len(libraryRecord)):
    books=int(libraryRecord[row][3])
    if(books>gold):
        gold=books
        goldID=row

    elif(books>silver):
        silver=books
        silverID=row

    elif(books>bronze):
        bronze=books
        bronzeID=row

    total=total+books
    average=total/len(libraryRecord)

    if(books<10):
        print(libraryRecord[row][0])#pupilID
print("Total",total)
print("Average",average)

print("Gold Winner is: ",gold,libraryRecord[goldID][1])
print("Silver Winner is: ",silver,libraryRecord[silverID][1])
print("Bronze Winner is: ",bronze,libraryRecord[bronzeID][1])
```

2020 P2 #6

6 Farshia is the regional manager for an insurance company.

She manages a team of sales staff.

She wants a program to analyse the performance of her team over a number of months.

Test data has been included in the code.

Open **Q06** in the code editor.

Write a program to:

- calculate and display the total sales made by each member of the team
- calculate and display the total sales made by the whole team
- display the first name, last name and the total sales made by the two members of the team with the highest total sales. (Ignore the possibility of two or more members of the team having the same total sales.)

Your program should function correctly even if the number of months or number of members of the team is changed.

Save your code as **Q06FINISHED** with the correct file extension for the programming language.

(20)

```
# Q6
secondSales=0
highestSales=0
allStaffTotal=0
second=0

# Structure of sales record is
# StaffID, First name, Last name, January sales, February sales,
# March sales, April sales, May sales, June sales

staffSales = [
["101TGY" , "George" , "Taylor" , 6009 , 5262 , 3745 , 7075 , 1943 , 4432],
["103FCY" , "Fehlix" , "Chayne" , 8717 , 2521 , 5777 , 6189 , 5089 , 6957],
["102SBY" , "Sumren" , "Bergen" , 5012 , 1063 , 7937 , 9560 , 1115 , 5499],
["104SBK" , "Samira" , "Beckle" , 1140 , 9206 , 3898 , 8544 , 5937 , 8705],
["105NBT" , "Nellie" , "Bogart" , 3017 , 3342 , 5939 , 2479 , 3374 , 2297],
["106CGT" , "Cheryl" , "Growth" , 9620 , 7160 , 5113 , 4803 , 5492 , 2195],
["107DGT" , "Danuta" , "Graunt" , 1583 , 7450 , 1026 , 7463 , 2390 , 6509],
["108JDN" , "Jaiden" , "Deckle" , 4064 , 4978 , 2984 , 3159 , 1464 , 4858],
["109JCK" , "Jimran" , "Caliks" , 6253 , 7962 , 2732 , 7504 , 2771 , 5193],
["110DDN" , "Deynar" , "Derran" , 6305 , 8817 , 5200 , 3647 , 3365 , 1256]]

#-----
# Write your code below this line

for staff in staffSales:
    staffTotal=0
    sales=3
    while(sales<9):
```

```
        staffTotal=staffTotal+staff[sales]
        sales=sales+1
    print(staff[1],staff[2],staffTotal)
    if(staffTotal>highestSales):
        secondSales=highestSales
        highestSales=staffTotal
        high=staff
    elif(staffTotal>secondSales):
        secondSales=staffTotal
        second=staff
    allStaffTotal=allStaffTotal+staffTotal

print("Total : ",allStaffTotal)
print("Highest Sale: ",high[1],high[2],highestSales)
print("Second Sale: ",second[1],second[2],secondSales)
```

2021 May Jun P2 #5

5 Ann likes to create programs.

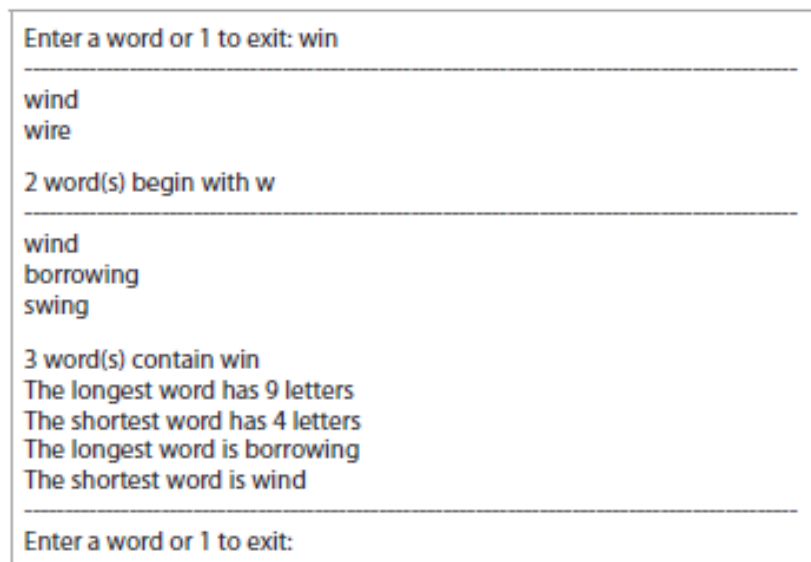
Open **Q05** in the code editor.

It contains an array of words.

Ann wants the program to:

- accept the input of a word, or the number 1 to exit. No validation of the input is required
- display all the words in the array that have the same first letter as the word that has been input
- calculate and display the number of words in the array that begin with the same letter as the word that has been input
- display all the words in the array that contain the word that has been input
- calculate and display
 - the number of words in the array that contain the word that has been input
 - the number of letters in the longest word that contains the word that has been input
 - the number of letters in the shortest word that contains the word that has been input
- display
 - the longest word that contains the word that has been input
 - the shortest word that contains the word that has been input
- repeat until the user inputs the number 1 to exit the program.

Figure 4 shows the display when the word **win** is input.



```
Enter a word or 1 to exit: win
-----
wind
wire

2 word(s) begin with w
-----
wind
borrowing
swing

3 word(s) contain win
The longest word has 9 letters
The shortest word has 4 letters
The longest word is borrowing
The shortest word is wind
-----
Enter a word or 1 to exit:
```

Figure 4

Write the code for the program.

Your program should function correctly even if the number of words in the array is changed.

Save your code as **Q05FINISHED** with the correct file extension for the programming language.

The next page may be used for planning / design work.


```
# Q05
# Array of words
wordArray = ["wind", "leer", "pushy", "lade", "size", "sob", "borrowing", "list",
             "perish", "hoax", "sticks", "seed", "impel", "large", "male", "silent",
             "quilt", "sobbed", "remarkable", "fantastic", "wire", "reflective", "pu
trid", "pushover", "swing"]
inputWord=""
# Add your code here
while inputWord!=1:
    shortest=5000
    longest=0
    longestWord=""
    shortestWord=""
    inputWord=input("Enter a word or 1 to exit: ")#number 1 to exit
    print("-----")
    if inputWord!='1':
        count=0
        for word in wordArray:
            if word[0]==inputWord[0]:#same first letter
                print(word)
                count=count+1
        print("\n",count,"word beginning with",inputWord[0])
        print("-----")
        #find shortest and longest
        count=0
        for word in wordArray:
            if inputWord in word:
                count=count+1
                print(word)
                length=len(word)
                print(length)

                if longest<length:
                    longest=length
                    longestWord=word
                if shortest>length:
                    shortest=length
                    shortestWord=word
        if count>0:
            print("\n",count," word(s) with",inputWord)
            print("The longest word has ",longest,"letters")
            print("The shortest word has ",shortest,"letters")
            print("The longest word is ",longestWord)
            print("The shortest word is ",shortestWord)
        else:
            print("There were 0 word with all the letters from",inputWord,"in
them.")
    print("End of program")
```

2021 Oct Nov P2 #5

5 Bianca has started to write a program.

The program is a guessing game about countries and their capital cities.

Open **Q05** in the code editor.

The program already:

- displays this menu

```
Menu
-----
[1] Add player name
[2] Play guess the capital city
[3] End game
-----
Input your menu choice: |
```

- asks for the user's menu choice.

Write the program code for the menu choices.

Menu choice [1] Add player name

If a player chooses this option, then they must input a player name.

Menu choice [2] Play guess the capital city

If a player chooses this option, then:

- they must answer five questions
- they select a question to answer by choosing a valid question number
- each question can only be selected once (they must not be able to choose a number more than once)
- the question should be displayed
- they must input the name of the capital city
- if they guess correctly their score must be incremented by 1
- if they guess incorrectly the correct answer must be displayed.

Menu choice [3] End game

If the user chooses this option, then the player name and score should be displayed.

Your program should function correctly even if the number of countries and capital cities in the file is increased.

Save your code as **Q05FINISHED** with the correct file extension for the programming language.

(20)

```
# Q05

def displayMenu():
    # Completed subprogram that displays the menu

    print("Menu")
    print("-----")
    print("[1] Add player name")
    print("[2] Play guess the capital city")
    print("[3] End game")
```

```
print("-----")

def getMenuChoice():
    # Completed subprogram that gets and validates the menu choice
    choices = [1,2,3]
    mChoice = 0

    # Menu choice is validated
    while mChoice not in choices:
        mChoice = int(input("Input your menu choice: "))

    # Valid menu option returned to the main menu
    return mChoice

def addPlayerName():
    # Add your code to:
    # ensure a player name is input
    # return the player name to the main menu

    player=""
    while(player==""):
        player=input("Enter your player name: ")
    return player

def guessCapital():
    # Partially completed subprogram to:
    # display questions
    # check guesses
    # return final score

    # Arrays holding question numbers, countries and their capital cities
    questions = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    countries = ["England", "France", "Spain", "Italy", "Germany",
"Scotland", "Wales", "United Arab Emirates", "China"]
    capitals = ["London", "Paris", "Madrid", "Rome", "Berlin", "Edinburgh",
"Cardiff", "Abu Dhabi", "Beijing"]

    questionCount = 1
    questionScore = 0

    # Add your code here
    while questionCount <= 5:
        questionNumbers = ""
        questionChoice = 0

        for question in questions:
            if question !=0:
                questionNumbers+=str(question)+" "
```

```
        #print(questionNumbers) - to produce question number array

    while str(questionChoice) not in questionNumbers:
        questionChoice =int(input("Pick a number from "+
questionNumbers))#by choosing a valid question number
        #questionChoice =int(input("Enter question number from 1 to
9:")) #1
        country=countries[questionChoice-1]#index - 0 (1-1=0)
        capital=capitals[questionChoice-1]

        guess=input("What is the capital of "+country+"? ").lower()

        if guess==capital.lower():
            print("Well done, you guessed correctly")
            questionScore+=1
            print("inner:",questionScore)
        else:
            print("You did not guess correctly. The capital of country
"+country+" is "+capital)

        questionCount+=1

        #set the question number to 0 so that it cannot be guessed again
        questions[questionChoice-1]=0
        print(questions)
        print("outter",questionScore)
        print("que count",questionCount)
    return questionScore
#return

menuChoice = 0
score = 0
playerName = ""

while menuChoice != 3:
    displayMenu()
    menuChoice = getMenuChoice()

    # Add your code to:
    #   call the relevant subprogram if the menu choice is 1 or 2
    #   display the player name and the score if the menu choice is 3
    if menuChoice==1:
        playerName=addPlayerName()
    elif menuChoice==2:
        score=guessCapital()
    else:
        print("Well Done "+playerName+". The score is "+ str(score))
```

2022 P2 #6

6 Carlos wants you to create a **guess the animal** game.

Open **Q06** in the code editor.

The code contains an array of animals.

It also contains a function that randomly selects an animal from the array. This is the secret word the user needs to guess.

Carlos wants the program to:

- generate the number of attempts the user has to guess the secret word. The maximum number of attempts is the length of the secret word +3. For example, the user has 8 attempts to guess when the secret word is **tiger**
- keep track of letters from incorrect attempts that are in the secret word and those that are not. There should be no duplicated letters
- display a message telling the user:
 - the number of letters in the secret word
 - how many attempts they have left
 - force the user to input a word that is the same length as the secret word
 - check whether the input word matches the secret word:
 - if the words match then a message that includes the secret word and the number of attempts taken to guess it is displayed
 - if the words do not match then:
 - letters from the attempt that appear in the secret word should be added to the correct letters store
 - letters from the attempt that do not appear in the secret word should be added to the wrong letters store
 - the contents of the correct and wrong letter stores are displayed
 - allow the user another attempt until they have guessed the word or have run out of attempts
 - display a message telling the user the game is over including the random word if the maximum attempts have been taken and the word has not been guessed.

Figure 4 shows the contents of the correct and wrong stores after two attempts to guess the secret word **cow**.

Secret word cow			
First attempt		Second attempt	
input	dog	input	cat
correct store	o	correct store	o c
wrong store	d g	wrong store	d g a t

Figure 4

Your program should include at least two subprograms that you have written yourself.

You must include comments in the code to explain the logic of your solution.

Save your code as **Q06FINISHED** with the correct file extension for the programming language.

```
# Q06
import random

def generateWord(pWords):
    # Fully completed function that generates and returns a secret word
    randomNum = random.randint(0,len(pWords)-1)
    secretWord = pWords[randomNum]
    return secretWord

# Add your subprograms here
def checkInput(pWordLength):
    check=""
    while len(check)!=pWordLength:
        check=input("Enter your guess. The secret word has "+str(pWordLength)+" letter")
    return check

def checkLetterInWord(pGuess,pWordToGuess,pLetter,ptype):
    for letter in pGuess:
        if ptype==0:
            if letter in pWordToGuess and letter not in pLetter:#correct
                pLetter=pLetter+letter
        else:
            if letter not in pWordToGuess and letter not in pLetter:#wrong
                pLetter=pLetter+letter
    return pLetter

def display(pCorrectLetter,pWrongLetter):
    if len(pCorrectLetter)>0:
        print("Letter in the secret word is: ",pCorrectLetter)
    if len(pWrongLetter)>0:
        print("Letter not in the secret word is: ",pWrongLetter)

#-----
# End of subprograms and start of main program from here

# Array of words
words = ["antelope","ape","badger","bear","beaver","bison","crocodile","elephant",
        "elk","ferret","goat","goose","kangaroo","llama","lion","monkey","moose",
        "orangutan","shark","snake","tiger","whale","wombat"]

# Secret word is generated.
wordToGuess = generateWord(words)

# Add your main program code here
#generate the number of attempts the user
wordLength=len(wordToGuess)
guessed=False
numAttempt=0
maxAttempt=wordLength+3
correctLetter=""
wrongLetter=""
```

```
while guessed==False and numAttempt<maxAttempt:
    print("You have",maxAttempt-numAttempt,"attempt")
    numAttempt=numAttempt+1

    guess=checkInput(wordLength)#
    if guess==wordToGuess:
        guessed=True
        print("Well Done","You can guess",wordToGuess,"in",str(numAttempt),"attempts")
    else:
        correctLetter=checkLetterInWord(guess,wordToGuess,correctLetter,0)
        wrongLetter=checkLetterInWord(guess,wordToGuess,wrongLetter,1)
        display(correctLetter,wrongLetter)
        #print("Wrong letters are:",wrongLetter)

if numAttempt==maxAttempt and guessed==False:
    print("Game Over")
```

2023 P2 #6

6 An agricultural college has a herd of dairy cows.

Data collected for the cows is stored in arrays.

The data stored is:

- the name of the breed
- the ease of care rating for that breed (1 is highest and 3 is lowest)
- the number of cows in the herd of that breed
- the volume of milk per day for a cow of that breed.

The college wants to present the data to farmers and to recommend the best breed of cows for them.

The best breed has the highest care rating and the largest volume of milk per day for a cow of that breed.

Open the file **Q06** in the code editor.

Write a program to:

- calculate the daily volume of milk produced by each breed
- add this daily volume to the data structure `tbl_dailyVolume`
- display a message informing the user what each field holds, such as breed, rating, volume per cow, count and total volume
- display the data for each breed
- calculate and display the total volume of milk produced each day by the herd
- find the recommended breed
- display the recommended breed by name.

Figure 8 shows the output from a functional program.

```
Fields are: Breed, Rating, Volume (cow), Count, Volume (day)
Red Chittagong 1 7.5 6 45.0
Sussex 2 5.7 3 17.1
Dexter 3 11.4 8 91.2
Abundance 2 11.4 7 79.8
Sahiwal 3 22.0 6 132.0
Vorderwald 1 15.2 4 60.8
Ayrshire 2 21.0 3 63.0
Jersey 1 18.3 7 128.1
Randall 2 19.0 3 57.0
Alderney 1 9.0 3 27.0
Carora 3 23.1 4 92.4
Gloucester 2 16.0 7 112.0
Total: 905.4 litres
Recommended breed: Jersey rating: 1 volume 18.3
```

Your program should function correctly even if the number of breeds represented in the data is changed.

Save your amended code as **Q06FINISHED** with the correct file extension for the programming language.

You may use this space for planning/design work.

The content of this space will **not** be assessed.


```

# Q06
tbl_breed = ["Red Chittagong", "Sussex", "Dexter", "Abondance",
             "Sahiwal", "Vorderwald", "Ayrshire", "Jersey",
             "Randall", "Alderney", "Carora", "Gloucester"]
tbl_rating = [1, 2, 3, 2, 3, 1, 2, 1, 2, 1, 3, 2]
tbl_count = [6, 3, 8, 7, 6, 4, 3, 7, 3, 3, 4, 7]
tbl_volume = [7.5, 5.7, 11.4, 11.4,
              22.0, 15.2, 21.0, 18.3,
              19.0, 9.0, 23.1, 16.0]
tbl_dailyVolume = []
# -----
# Write your code below this line
totalVolume=0
maxIndex=0
print("Field are: Breed, Rating, Volume (cow), Count, Volumn (day)")
for index in range(len(tbl_breed)):
    todayVolume=tbl_volume[index]*tbl_count[index]

    tbl_dailyVolume.append(todayVolume)
    print(tbl_breed[index],tbl_rating[index],tbl_volume[index],tbl_count[index],tbl_dailyVolume[index])
    totalVolume=totalVolume+todayVolume
    #print(totalVolume)
    #the best breed
    if (tbl_rating[index]<=tbl_rating[maxIndex]) and
(tbl_volume[index]>=tbl_volume[maxIndex]):
        maxIndex=index
layout="Total : {} liters"
print(layout.format(totalVolume))
print("Recommended breed:"+tbl_breed[maxIndex]+" rating:
"+str(tbl_rating[maxIndex])+" volume: "+str(tbl_volume[maxIndex]))

```

```

Shell x
Field are: Breed, Rating, Volume (cow), Count, Volumn (day)
Red Chittagong 1 7.5 6 45.0
Sussex 2 5.7 3 17.1
Dexter 3 11.4 8 91.2
Abondance 2 11.4 7 79.8
Sahiwal 3 22.0 6 132.0
Vorderwald 1 15.2 4 60.8
Ayrshire 2 21.0 3 63.0
Jersey 1 18.3 7 128.1
Randall 2 19.0 3 57.0
Alderney 1 9.0 3 27.0
Carora 3 23.1 4 92.4
Gloucester 2 16.0 7 112.0
Total : 905.4 liters
Recommended breed:Jersey rating: 1 volume: 18.3

```

2024 P2 #6

6 A program stores pairs of words in a two-dimensional array.

Each word in a pair starts with a different letter.

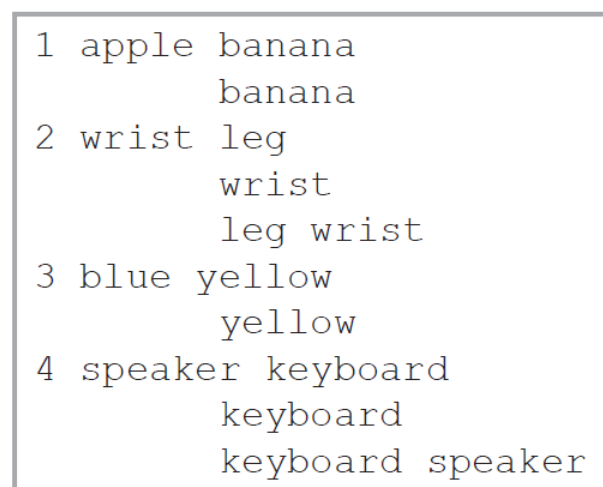
Each pair of words should be in alphabetical order, but some are not.

The last pair in the array is an empty pair.

The program must:

- replace the final blank pair of words with the variables **word1** and **word2**
- display the pair number of each pair, followed by each word in the pair, without punctuation
- display the longer word in the pair, indented
- display any pair found to be not in alphabetical order, in alphabetical order, indented, without punctuation.

Figure 9 shows part of the intended output from a functional program.



```
1 apple banana
    banana
2 wrist leg
    wrist
    leg wrist
3 blue yellow
    yellow
4 speaker keyboard
    keyboard
    keyboard speaker
```

Figure 9

Open the file **Q06** in the code editor.

Write a program to produce the intended output.

You must use the structure and variables given in **Q06** to complete the program.

Your program should function correctly even if the number of pairs in the array is changed.

You should use techniques to make your code easy to read.

Save your amended code as **Q06FINISHED** with the correct file extension for the programming language.

(20)

Q06

```
tblWords = ["apple", "banana",  
            ["wrist", "leg"],  
            ["blue", "yellow"],  
            ["speaker", "keyboard"],  
            ["lavender", "tulip"],  
            ["pencil", "chalk"],  
            ["apartment", "house"],  
            ["bottom", "top"],  
            ["snow", "fog"],  
            ["beach", "mountain"],  
            ["", ""]]
```

```
word1 = "newspaper"  
word2 = "book"  
row=1  
numRows=len(tblWords)
```

```
tblWords[numRows-1][0]=word1  
tblWords[numRows-1][1]=word2
```

Write your code below this line

for words in tblWords:

```
    print(str(row)+" "+words[0]+" "+words[1])  
    row+=1
```

```
    indexMaxLength=0  
    if (len(words[1])>len(words[indexMaxLength])):  
        indexMaxLength=1
```

```
    print(" "*5+words[indexMaxLength])
```

#check order

```
    if (words[0]>words[1]):  
        temp=words[0]  
        words[0]=words[1]  
        words[1]=temp  
    print(" "*5+words[0],words[1])
```