Lecture Notes:Week 5

From JOCWiki

Contents

[hide]

Dictionary[edit]

Intro[edit]

A dictionary is a data structure that consists of a "key" and its associated "value". In python, a dictionary is defined with curly braces {}.

Similarities between List and Dictionary[edit]

Differences between List and Dictionary[edit]

1)Accessing the elements

In a list the list items are accessed with help of an "index" value.

For example:

id=[142,234,311,444,456]

id[1] #It can be use to access the element 234.

In a dictionary ,in order to access the "values" we use "keys" instead of an index value.

For example:

stud_1={"name" : "Ankit" , "rollno" : 142 }

To access the value "Ankit" we use the key "name" in the following manner:

stud_1["name"] #It will return the value "Ankit"

How to define a Dictionary?[edit]

Syntax:

dict_name[key]=value

There are two ways in which a dictionary can be defined which are as follows:

1)Creating an empty dictionary:

stud_1={}

Then adding data to it.

stud_1["name"]="Ankit"

#Here "name" is the key and "Ankit" is the associated value of the key.

stud_1["rollno"]=142

2)Defining and initialising the dictionary at the time of declaration:

stud_1={"name" : "Ankit" ,"rollno" : 142 }

Functions[edit]

1)get(): *: This function returns the value of the specified key.*

Syntax:

dict_name.get(key_name)

2)copy() : *This function is use to copy the existing dictionary to another dictionary.*

Syntax:

new_dict=existing_dict.copy()

3)pop() : *This function removes the item from the dictionary associated with specified key.*

Syntax:

dict_name.pop(key)

4)keys() : *This function returns all the keys present in the dictionary.*

Syntax:

dict_name.keys()

5)values(): *This function returns all the values of the keys present in the dictionary.*

Syntax:

dict_name.values()

6)clear() : *This functions deletes all the elements from the dictionary and returns an empty dictionary {}.*

Syntax:

dict_name.clear()

Speech Recognition[edit]

Need[edit]

What is Speech Recognition? *It is the process of converting the spoken words to text. Python supports many speech recognition engines and APIs, including Google Speech Engine, Google Cloud Speech API, Microsoft Bing Voice Recognition, IBM Speech to text. If you have ever interacted with Alexa or have ever ordered Siri to complete a task, you have already experienced the power of speech recognition.*

What is the need for Speech Recognition? *Speech recognition has various applications ranging from the automatic transcription of speech data (like voicemails) to interacting with robots via speech.*

Prerequisites[edit]

=*WAV FILE CONVERSION[edit]

Any audio file that you would like to use should be saved with .wav extension.

*INSTALLATION OF SPEECH RECOGNITION[edit]

##########################

1.If using windows, go to prompt and type "pip install SpeechRecognition"

2.In iconsole, type same thing and it gets installed. (But it appears that the console becomes slow and sometimes unstable occasionally

3.The audio file you saved and the python file must be in the same location.

##########################

Syntax[edit]

The first step, as always, is to import the required libraries. In this case, we only need to import the speech_recognition library that we just downloaded.

import speech_recognition as sr

To convert speech to text the one and only class we need is the Recognizer class from the speech_recognition module. Depending upon the underlying API used to convert speech to text, the Recognizer class has the following methods:

'recognize_bing(): Uses Microsoft Bing Speech API

'recognize_google()': Uses Google Speech API

'recognize_google_cloud()': Uses Google Cloud Speech API

'recognize_houndify()': Uses Houndify API by SoundHound

'recognize_ibm()': Uses IBM Speech to Text API

'recognize_sphinx()': Uses PocketSphinx API

Among all of the above methods, the recognize_sphinx() method can be used offline to translate speech to text...

Monte Hall[edit]

The Monty Hall problem is a probability puzzle named after Monty Hall, the original host of the TV show. Let's Make a Deal. It's a famous paradox that has a solution that is so absurd, most people refuse to believe it's true.

Intro[edit]

Game[edit]

There are 3 doors, behind which are two goats and a car. You pick a door (call it door A). You're hoping for the car of course. Monty Hall, the game show host, examines the other doors (B & C) and opens one with a goat. (If both doors have goats, he picks randomly.) Here's the game: Do you stick with door A (original guess) or swap to the unopened door? Does it matter? Surprisingly, the odds aren't 50-50. If you swap doors you'll win 2/3 of the time!

Should you Swap?

Believe it or not, it's actually to your benefit to swap: If you swap, you have roughly a 2/3 chance of winning the car. If you stick to your original choice you have roughly a 1/3 chance of winning the car. The answer sounds unlikely. After door 3 is opened, you would think that you then have two doors to choose from…both with the same odds. However,

you are actually much more likely to win if you swap. Those who swapped doors won about 2/3 of the time Those who didn't swap won about 1/3 of the time This fact has been proved over and over again with a plethora of mathematical simulations. If you're stumped and still don't believe it — don't worry, even mathematicians scratch their head on this one. One genius mathematician, Paul Erdős didn't believe the answer was right until he was shown simulations of the winning, "swap", strategy.

A lot of people have trouble with the better odds of swappimg doors including me, until I realized a simple fact: the odds are better if you swap because Monty curates the remaining choices. Let's say you played the game where Monty doesn't know the location of the car. It wouldn't make any difference if you swap or not (your odds would be 50% no matter what). But this isn't what happens. The Monty Hall problem has a very specific clause: Monty knows where the car is. He never chooses the door with the car. And by curating the remaining doors for you, he raises the odds that swapping is always a good bet.

Vision(Behind Theory)[edit]

```
I'm going to assume that you are familiar with Bayes' Theorem, which is a way to figure out conditional probability (if
event A happens, what's the probability event B will happen?).
```

The basis for the solution is the same as in the above scenario. There are three doors, one has the car behind it. You pick a door, then Monty opens one of the other doors to reveal a goat.

Let's assume you pick door 1 and then Monty shows you the goat behind door 2. In order to use Bayes' Theorem we need to first assign an event to A and B.

Let event A be that the car is behind door number 1. Let event B be that Monty opens up door 2 to show the goat.

Here is the **Baye's Theorem**'

*Pr(A|B)=(Pr(B|A)*Pr(A))/Pr(B)*

```
      =(1/2*1/3)/(1/3*1/2+1/3*0+1/3*1)
```

Pr(A) is pretty simple to figure out. There is a 1/3 chance that the car is behind door 1. There are two doors left, and each has a 1/2 chance of being chosen — which gives us Pr(B|A), or the probability of event B, given A. Pr(B), in the denominator, is a little trickier to figure out. Consider that:

You choose door 1. Monty shows you a goat behind door 2. If the car is behind door 1, Monty will not choose it. He'll open door 2 and show a goat 1/2 of the time. If the car is behind door 2, Monty will always open door 3, as he never reveals the car. If the car is behind door 3, Monty will open door 2 100% of the time. As Monty has opened door 2, you know the car is either behind door 1 (your choice) or door 3. The probability of the car being behind door 1 is 1/3. This means that the probability of the car being behind door 3 is 1 – (1/3) = 2/3. And that is why you switch.

Syntax[edit]

*import random doors=[0]*3 goatdoor=[0]*2 swap=0 #No. of winnings occured with swapping dont_swap=0 #No. of winnings occured without swapping j=0 while(j<10):*

```
   x=random.randint(0,2) #xth door will comprise of car(BMW)
   doors[x]='BMW'
```

```
    for i in range(0,3):
        if(i==x):
            continue
        else:
            doors[i]='goat'
            goatdoor.append(i)
    choice=int(input("Enter your choice: "))
    door_open=random.choice(goatdoor)
    while(door_open==choice):
        door_open=random.choice(goatdoor)
    ch= input("Do you want to swap? If yes type 'y' else type 'n' ")
    if(ch=='y'):
        if(doors[choice]=='goat'):
            print("You won")
            swap=swap+1
        else:
            print("You lost")
    else:
        if(doors[choice]=='goat'):
            print("You lost")
        else:
            print("You won")
            dont_swap=dont_swap+1
    j=j+1
print("Total number of winnings without swapping are:",dont_swap) print("Total number of winnings with swapping are:",swap)
```

Output[edit]

Enter your choice: 1

Do you want to swap? If yes type 'y' else type 'n' y You won

Enter your choice: 2

Do you want to swap? If yes type 'y' else type 'n' y You won

Enter your choice: 1

Do you want to swap? If yes type 'y' else type 'n' y You lost

Enter your choice: 2

Do you want to swap? If yes type 'y' else type 'n' y You won

Enter your choice: 0

Do you want to swap? If yes type 'y' else type 'n' n You won

Enter your choice: 0

Do you want to swap? If yes type 'y' else type 'n' n You lost

Enter your choice: 1

Do you want to swap? If yes type 'y' else type 'n' n You lost

Enter your choice: 2

Do you want to swap? If yes type 'y' else type 'n' n You lost

Enter your choice: 0

Do you want to swap? If yes type 'y' else type 'n' y You won

Enter your choice: 0

Do you want to swap? If yes type 'y' else type 'n' y You won Total number of winnings without swapping are: 1 Total number of winnings with swapping are: 5

ROCK PAPER SCISSOR[edit]

Rock Paper Scissors is the classic playground game converted into a fun online multiplayer game. Have you ever had a dispute or disagreement where you and a friend cannot decide who is right and who is wrong? If so then you have probably settled it by playing rock paper scissors.

Introduction & Definition[edit]

Rock paper scissors (also known as scissors rock paper, paper rock scissors and scissors paper stone) is a hand game usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are "rock" (a closed fist), "paper" (a flat hand), and "scissors" (a fist with the index finger and middle finger extended, forming a V). "Scissors" is identical to the two-fingered V sign (also indicating "victory" or "peace") except that it is pointed horizontally instead of being held upright in the air. A simultaneous, zero-sum game, it has only two possible outcomes: a draw, or a win for one player and a loss for the other.

A player who decides to play rock will beat another player who has chosen scissors ("rock crushes scissors" or sometimes "blunts scissors"), but will lose to one who has played paper ("paper covers rock"); a play of paper will lose to a play of scissors ("scissors cuts paper"). If both players choose the same shape, the game is tied and is usually immediately replayed to break the tie. The type of game originated in China and spread with increased contact with East Asia, while developing different variants in signs over time. Other names for the game in the English-speaking world include roshambo and other orderings of the three items, with "rock" sometimes being called "stone".

Interesting facts about RPS game *Did you know that this game first appeared in China in the 17th century? Yes, it was not invented in Europe or America but in Asia. Europe started to play this game only in 19th century.*

Statistics say that people usually choose Scissors in the first round and Rock in the second.

There is a robot developed in Japan which wins with 100% chance. It analyzes movement of your hand muscles to predict what choice you'll show

Good advice

If you want to win, you need to be a good psychologist to predict the next choice of your opponent. There are many strategies and different people have different patterns of behavior.

But if you follow this advice you will win in most cases. If your last choice was... ... Rock, then choose Scissors in the next round ... Scissors, then choose Paper in the next round ... Paper, then choose Rock in the next round It will work only with not experienced players. The strategy is based on experiments of Zhejiang University.

Program for Rock Paper Scissors[edit]

def rock_paper_scissor(num1,num2,bit1,bit2):

```
    p1=int(num1[bit1])%3
    p2=int(num2[bit2])%3
    if(player1[p1]==player2[p2]):
        print("Draw")
```

```
    elif(player1[p1]=='Rock' and player2[p2]=='Scissor'):
        print("Player1 Wins")
    elif(player1[p1]=='Rock' and player2[p2]=='Paper'):
        print("Player2 wins")
    elif(player1[p1]=='Paper' and player2[p2]=='Rock'):
        print("Player1 wins")
    elif(player1[p1]=='Paper' and player2[p2]=='Scissor'):
        print("Player2 wins")
    elif(player1[p1]=='Scissor' and player2[p2]=='Rock'):
        print("Player2 wins")
    else:
        print("Player1 wins")
```

player1={0:'Rock',1:'Paper',2:'Scissor'} player2={0:'Paper',1:'Rock',2:'Scissor'} while(1):

```
    num1=input("Player1!Enter Your choice:")
    num2=input("Player2!Enter Your choice:")
    bit1=int(input("Player1!Enter the secret bit position"))
    bit2=int(input("Player2!Enter the secret bit position"))
    rock_paper_scissor(num1,num2,bit1,bit2)
    ch=input("Do you wish to continue?y/n")
    if(ch==n):
        break
```

Output[edit]

Player1!Enter Your choice:12425

Player2!Enter Your choice:324334

Player1!Enter the secret bit position2

Player2!Enter the secret bit position4 Draw

Do you wish to continue?y/n

Searching and Sorting-20 Questions Game:[edit]

Introduction:[edit]

Definition:[edit]

Example for 20 Questions game:[edit]

Binary Search:[edit]

def binary(list1,n):

```
pos = -1
l =0
u = len(list1)-1
found = False
while l<=u and not found:
  mid = (l+u) //2
  if list1[mid] == n:
    found=True
  else:
    if list1[mid]<n:
      l=mid+1
    else:
```

```
       u=mid-1
 return found
```

print(binary([1,2,3,5,8], 8))

Bubble Sort:[edit]

Bubble sort is one of the sorting algorithm.It is easy to understand and simple to implement.In bubble sort,there are n-1 iterations where n is the number of elements in the list.This algorithms works by comparing an element at position i with element at positon i+1 ,if the ith element is greater than i+1 then swapping/exchange of these elements takes place.At the end of each iteration we get largest element at the end of the unsorted list.

Algorithm:

N=Length of the list

iterations=N-1

for i < iterations:

```
     for j < length-1-i:
          if a[j]>a[j+1]:
                temp=a[j+1]
                a[j+1]=a[j]
                a[j]=temp
```

Program:

l=[]

length=int(input("Enter the size of array"))

print("Enter the elements")

for i in range(length):

```
    uin=int(input())


    l.append(uin)
```
for i in range(length):

```
    r=length-1-i
    for j in range(r):
        if(l[j]>l[j+1]):
            temp=l[j]
            l[j]=l[j+1]
            l[j+1]=temp
```
print(l)

Output:

Enter the size of array5

Enter the elements

45 21 78 65 -8

[-8, 21, 45, 65, 78]

Example for Bubble sort:[edit]

Linear search[edit]

list1 = [2,3,4,5,7,2,2,5,76,8,6,5,3]

item_to_search = int(input())

found = False

for i in range(len(list1)):

```
if list1[i] == item_to_search:
found = True
print(item_to_search, 'found at index ', i)
break
```

if found == False:

```
print(item_to_search, 'Not found!')
```

Assignment Doubts[edit]

End Sort[edit]

Week 5 Assignment 2 - End Sort Related FAQs[edit]

This is in regard to assignment[edit]

Hello all

Sample case 3 public

Test Case 5 1 3 2 7 Ouput 3

When i follow the procedure i am ending up with 5 trails. Can any one helping in getting the answer in 3 trails with the description given in webpage

As per me Trial 1

```
5 is moved to the end.
list becomes   1 3 2  7  5
```

Trial 2

```
3 is moved to the end
```

```
 list become  1 2 7 5 3
```
Trail 3

```
  7 is moved to the end
  list becomes 1 2 5 3 7
```
Trail 4

```
   5 is moved the end
   list become 1 2 3 7 5
```
Trail 5

```
  7 is moved to the end
  and the list is sorted. hence we require 5 trails.
```
Can some one help me in getting what are the trails to be followed so that we can get in 3 trails as requested in the program.

I am not clear of how to program it is 3 trails with the available description

If some one kindly explains the trails, i will program accordingly.

Solution

In first move, rather than moving 5, move 3 to the end.

In second step, move 5.

And in the third iteration, move 7 :-)

Semi-Prime[edit]

Isn't 62 a sum of semi primes? In Week 5 assignment 3, a test case contains 62 as input, it is a sum of the semi primes, 4 and 58. So the expected output should be "Yes" instead of the "no".

Solution: A semiprime number is an integer which can be expressed as a product of two distinct primes. 4 is not a semiprime